A high-quality video denoising algorithm based on reliable motion estimation

A High-Quality Video Denoising Algorithm based on Reliable Motion Estimation

Ce Liu[1] William T. Freeman[1,2]

[1]Microsoft Research New England [2] Massachusetts Institute of Technology

Abstract. Although the recent advances in the sparse representations of images have achieved outstanding denosing results, removing real, structured noise in digital videos remains a challenging problem. We show the utility of reliable motion estimation to establish temporal correspondence across frames in order to achieve high-quality video denoising. In this paper, we propose an adaptive video denosing framework that integrates robust optical flow into a non-local means (NLM) framework with noise level estimation. The spatial regularization in optical flow is the key to ensure temporal coherence in removing structured noise. Furthermore, we introduce approximate K-nearest neighbor matching to significantly reduce the complexity of classical NLM methods. Experimental results show that our system is comparable with the state of the art in removing AWGN, and significantly outperforms the state of the art in removing real, structured noise.

Key words: Video denoising, structured noise, approximate K-nearest neighbors, non-local means, optical flow

1 Introduction

Image quality enhancement is a long-standing area of research. As low-end imaging devices, such as web-cams and cell phones, become ubiquitous, there is ever more need for reliable digital image and video enhancement technologies to improve their outputs. Noise is dominant factor that degrades image quality.

We focus on video denoising in this paper. Our goal is to achieve an efficient, adaptive and high-quality video denoising algorithm that can effectively remove real, structured noise introduced by low-end camcorders and digital cameras. Unlike synthetic, additive noise, the noise in real cameras can have strong spatial correlations. This structured noise can have many different causes, including the demosaicing process in CCD camera. We find that computer vision analysis and techniques are useful in addressing these noise problems.

For image and video denoising, a key is to exploit the property of *image sparsity* [1–3]. In the frequency domain, image sparsity can be formulated as high-kurtotic marginal distribution of bandpass filtering, and image coring [4, 5] is a straightforward denoising algorithm that preserves large-magnitude responses while shrinking small-magnitude responses. In the spatial domain, image sparsity arguments imply that for any image patch, there will be similar ones in other locations of the image. The non-local means (NLM) method [6] was introduced to remove noise by averaging pixels in an image

(a) RGB noisy image (b) R channel (c) G channel (d) B channel

Fig. 1. In real video denoising scenarios, images contain structured noise. For this example, the blue channel is heavily contaminated with structured noise that can be mixed with signal. Even the state-of-the-art video denoising algorithm [10] fails to obtain temporally smooth denoising results. On the contrary, the proposed algorithm in this paper is able to remove structured noise and obtain temporally smooth results.

weighted by local patch similarities. Recently, these two forms of sparsity are combined in [7] to produce the state of the art in image denoising.

Sparsity also resides in videos. Most videos are temporally consistent; a new frame can be well predicted from previous frames [8, 9]. Indeed, *temporal coherence* can be vital to achieving high quality. Given two noise-free videos that share the same average peak signal-to-noise ratio (PSNR), we may prefer the one with more temporal coherence.

Although the state of the art video denoising algorithms often satisfy the temporal coherence criterion in removing additive white Gaussian noise (AWGN), many real videos contain structured noise that makes it challenging to ensure temporal coherence. As shown in Figure 1, the blue channel of the image contains structured noise that can be misinterpreted as signal by many denoising algorithms. Confused by the jittering blocky noise, block matching techniques (*e.g.* in [10]) may fail to track the true motion of the objects.

Therefore, in contrast with [11], we argue that high-quality video denoising, especially when structured noise is taken into account, indeed needs reliable motion estimation. In theory, estimating motion and noise suffers from a chicken-and-egg problem, since motion should be estimated from the underlying signals after denoising, and denoising relies on the temporal correspondence from motion estimation. In practice, however, we used robust optical flow with spatial regularization to establish reliable temporal correspondence despite noise. Because of its power, we use non-local means

(NLM) as the backbone of our system. Due to the inherent search complexity of NLM, searching for similar patches is often constrained to a small neighborhood. We introduce approximate K-nearest neighbor patch matching with much lower complexity to allow for searching over the entire image for similar patches. In addition, we estimate the noise level at each frame for noise-adaptive denoising.

We conduct experiments to test our theories. We first show that our system is comparable with the state of the art [10] in removing additive white Gaussian noise (AWGN) on benchmark videos. Then, we show the importance of establishing good temporal correspondence through some real, challenging examples. Our video denoising system produces high-quality and temporal coherent denoising results on these real-world examples, outperforming the state of the art.

2 Related work

Image and video denoising has been studied for decades. As it is beyond the scope of this paper to provide a thorough review, we will focus on reviewing the work closest to ours.

Image sparsity can manifest itself in different forms. When images are decomposed into sub-bands, sparsity leads to image coring algorithms on wavelets coefficients [4, 5]: large-magnitude coefficients that more likely correspond to true image signal should be retained, whereas small-magnitude coefficients that more likely correspond to noise should be shrunk. When the prior of natural images is incorporated in denoising [12–14], image sparsity is reflected by the heavy-tailed robust potential functions associated with band-pass filters: pixels in a neighborhood are encouraged to be similar, but occasional dissimilarity is allowed. Other denoising techniques such as PDE's [15] and region-based denoising [16] also implicitly formulate sparsity in their representation.

Unfortunately, wavelet- and natural image prior-based denoising algorithms can introduce unwanted artifacts to denoised images. Recently, image sparsity was formulated as image self similarity, namely patches in an image are similar to one another, which leads to the non-local means (NLM) methods [6]. In NLM, similar patches are aggregated together with weights based on patch similarities. This surprisingly simple algorithm produces high-quality results. NLM was also extended to video denoising [11] by aggregating patches in a space-temporal volume. Because of this, we choose NLM as the framework of our video denoising system.

The frequency and spatial forms of image sparsity are seamlessly integrated in [7], where similar patches are stacked in a 3D array, and both hard and soft shrinkages are performed on a 3D DCT transformed domain. This idea can be easily extended to video denoising, and state of the art video denoising results were reported in [10].

In [17], it was claimed that under the NLM framework "denoising image sequences does not require motion estimation" because the aperture problem, which often causes motion estimation to fail on textureless regions, is indeed beneficial to denoising as redundant patches are available for better denoising. However, we disagree on this point. As shown in Figure 1, structured noise can mislead the search for similar patches and then breaks the temporal coherence criterion in video denoising. We attempt to resolve this issue in this paper.

Patch matching has been widely used for image synthesis and editing, *e.g.* [18]. Recently, random patch matching was proposed to significantly speed up nearest neighbor searching on images [19]. The key ideas are random initialization and improvement, and spatial propagation. We extend this idea to random K-nearest neighbor matching to speed up patching matching under the NLM framework.

3 A Temporally Coherent Video Denoising Framework

For every patch in a video, we want to find a set of supporting patches from this frame and temporal adjacent frames that are similar to this patch. To ensure the nature of spatial and temporal sparsity of videos, we want spatially neighboring pixels and temporally corresponding pixels to share similar structures of supporting patches. This is ensured by approximate K-nearest neighbor matching for a single frame and establishing temporal correspondence using optical flow.

3.1 Approximate K-nearest neighbors (AKNN) for a single frame

Mathematically, we use notion $\{I_1,\ I_2,\ \cdots,\ I_T\}$ to denote an input noisy sequence that contains T frames. We use $\mathbf{z} = (x, y, t)$ to index the space-time volume, and $P(\mathbf{z})$ (or equivalently $P(x, y, t)$) to denote a patch at location $\mathbf{z}$. In this subsection, we focus on searching for K-nearest neighbors within a single frame, and will extend to multiple frames in next subsection. For notational convenience, we let $\mathbf{q} = (x, y)$ and omit time t from the notation. For each pixel $\mathbf{q}$, we want to obtain a set of *approximate K-nearest neighbors* (AKNN) $\mathcal{N}(\mathbf{q}) = \{P(\mathbf{q}_i)\}_{i=1}^K$. Let $\mathbf{v}_i = \mathbf{q}_i - \mathbf{q}$ be the offset of the found patch from the current patch. Searching for $\mathcal{N}(\mathbf{q})$ is equivalent to searching for $\{\mathbf{v}_i\}$.

For efficiency, we used the *priority queue* data structure to store the K-nearest neighbors such that the following increasing order is always maintained for the elements in the priority queue:

$$D\big(P(\mathbf{q}), P(\mathbf{q}_i)\big) \leqslant D\big(P(\mathbf{q}), P(\mathbf{q}_j)\big),\ \forall 1 \leqslant i < j \leqslant K, \tag{1}$$

where $D(\cdot, \cdot)$ is sum of square distance (SSD) over two patches, defined as

$$D\big(P(\mathbf{q}), P(\mathbf{q}_i)\big) = \sum_{\mathbf{u}\in[-s,s]\times[-s,s]} \Big(I(\mathbf{q}+\mathbf{u}) - I(\mathbf{q}_i+\mathbf{u})\Big)^2. \tag{2}$$

When a new patch is pushed back to this queue, it will be discarded if the distance is greater than the last element of the queue, or will otherwise be added at the appropriate position in the priority queue. A heap implementation of the priority queue has complexity $O(\log K)$.

Suppose there are N pixels in an image, then the complexity of a brute-force K-nearest neighbor search over the entire image is $O(N^2 \log K)$, almost implausible for high-definition (HD) videos. Inspired by the approximate nearest neighbor algorithm in [19], we propose an approximate K-nearest neighbor algorithm that contains three phases, *initialization*, *propagation* and *random search*, which will be explained below.

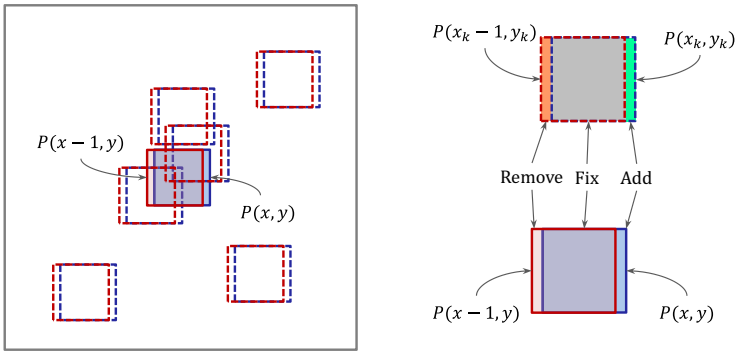


Fig. 2. The approximate K-nearest neighbors (AKNN) of patch $P(x,y)$ (blue) can be improved by propagating AKNN from $P(x-1,y)$ (red). Left: the approximate K-nearest neighbors of $P(x-1,y)$ are shifted one pixel to the right to be pushed to the priority queue of (x,y). Right: we do not need to recompute patch distances with this shift. To compute the distance between $P(x_k,y_k)$ and $P(x,y)$, we can simply take the distance between $P(x_k-1,y_k)$ and $P(x-1,y)$, remove the left column (orange) and add the right column (green).

To ensure the order in Eqn. (1), any new item generated in these phases is pushed back to the priority queue.

Initialization. The K-nearest neighbors are initialized by randomization

$$\mathbf{v}_i = \sigma_s \mathbf{n}_i \tag{3}$$

where $\mathbf{n}_i$ is a standard 2d normal random variable, and σ_s controls the radius. In this paper we set $\sigma_s = w/3$ where w is the width of an image.

Propagation. After initialization, an iterative process that consists of *propagation* and *random search* is performed in an interleaving manner. The idea is to improve the approximate K-nearest neighbor set based on the fact that neighboring pixels tend to have similar AKNN structures (offsets). The propagation procedure intertwines between scanline order and reverse scanline order [19]. In the scanline order, we attempt to improve AKNN $\{\mathbf{v}_i(x,y)\}$ using neighbor $\{\mathbf{v}_i(x-1,y)\}$ and $\{\mathbf{v}_i(x,y-1)\}$. In the reverse scanline order, we attempt to improve $\{\mathbf{v}_i(x,y)\}$ using neighbor $\{\mathbf{v}_i(x+1,y)\}$ and $\{\mathbf{v}_i(x,y+1)\}$.

As an example, we use the AKNN of patch $P(x–1,y)$ (red, filled square) to improve the AKNN of patch $P(x,y)$ (blue, filled square) as shown in Figure 2. The approximate K-nearest neighbors (red, dashed squares) of $P(x–1,y)$ are shifted one pixel to the right to obtain a *proposal set* (blue, dashed squares), which are pushed back to the priority queue of $P(x,y)$. There is no need of recalculating the patch distance as illustrated in Figure 2. We only need to compute the pixel-distance contributed from the non-overlapping region, as in [19].

This propagation is very similar to the sequential update scheme in belief propagation [20]. Although the (implicit) objective function is independent for neighboring pixels, this propagation scheme makes neighboring patches share similar AKNN structures.

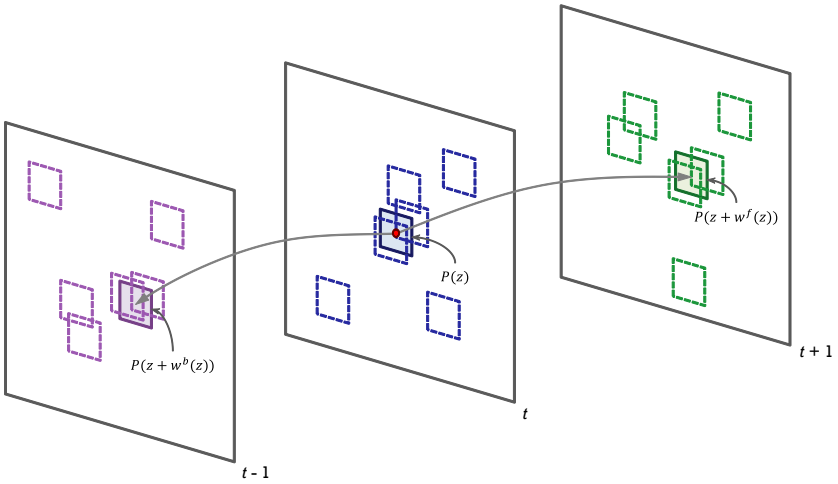


Fig. 3. Illustrations of the supporting patches in spatial-temporal domain for a patch $P(\mathbf{z})$. We use approximate K-nearest neighbor patch matching at frame t to find an initial set of spatially supporting patches $\mathcal{N}(\mathbf{z})$, shown as dashed boxes. Since $\mathbf{z}$ corresponds to $\mathbf{z}+w^f(\mathbf{z})$ in frame $t+1$, the AKNN of $\mathbf{z}+w^f(\mathbf{z})$ is added to the set of supporting patches. Likewise, the AKNN of $\mathbf{z}+w^b(\mathbf{z})$ that $\mathbf{z}$ corresponds to in frame $t-1$ is also added. In fact, we use the AKNN's along the motion path up to $\pm H$ frames to form the entire supporting patches.

Random search. After the propagation step, we allow every patch to randomly match other patches in the image for M times using the following mechanism

$$\mathbf{v}_i = \sigma_s \alpha^i \mathbf{n}_i,\ i = 1, \cdots, M \tag{4}$$

where $\mathbf{n}_i$ is a standard 2d normal random variable, $\alpha = \frac{1}{2}$ and $M = \min(\log_2 \sigma_s, K)$. So the radius of the random search $\sigma_s \alpha^i$ decreases exponentially. Each random guess is again pushed back to the priority queue to maintain the increasing order of the queue.

This approximate K-nearest neighbor patch matching algorithm converges quickly. We found that running more than 4 iterations does not generate more visually pleasing results. Furthermore, in the matching procedure we excluded the patch itself because it has distance zero. In the end, we add patch $P(x, y)$ into $\mathcal{N}(x, y)$.

3.2 Non-local means with temporal coherence

We feel that temporal coherence is vital for denoising. Two algorithms may perform equally well for a single frame, but the one that produces more temporal coherent results is preferred. It was argued in [11] that denoising image sequence does not require motion estimation. But for real sequences, it can be difficult to distinguish high-intensity structured noise from image signal. Therefore, it is important to establish temporal correspondence between adjacent frames and require corresponding pixels to be similar. Instead of formulating this property explicitly, we design a mechanism to satisfy this criterion implicitly.

We implemented a state-of-the-art optical flow algorithm [21] which integrates Lucas-Kanade [22] into the total variation optical flow framework [23] with robust L1 norms for both the data and smoothness terms. Since, in general, optical flow is not invertible, we estimate forward flow $w^f(\mathbf{z}) = [v_x, v_y, 1]$ from frame I_t to I_{t+1}, and backward flow $w^b(\mathbf{z}) = [v_x, v_y, -1]$ from frame I_t to I_{t-1}, in order to establish bidirectional correspondence.

Under this setup, pixel $\mathbf{z}$ corresponds to $\mathbf{z} + w^f(\mathbf{z})$ in next frame and to $z + w^b(\mathbf{z})$ in previous frame, as shown in Figure 3. This process can propagate up to $\pm H$ frames and we set $H = 5$ in our system. We include the AKNN of temporally corresponding pixels to the set of supporting patches, and therefore the motion path results in a series of AKNN's $\{\mathcal{N}_{t-H}, \cdots, \mathcal{N}_{t-1}, \mathcal{N}_t, \mathcal{N}_{t+1}, \cdots, \mathcal{N}_{t+H}\}$, which forms the supporting patches for $P(\mathbf{z})$. $\mathcal{N}_i = \{P(\mathbf{z}_{ij})\}_{j=1}^K$ denotes the patches in the AKNN at the ith frame. Notation $\mathbf{z}_{ij} = (x_{ij}, y_{ij}, i)$ means the jth-nearest neighbor of the corresponding pixel at frame i. The non-local means estimate for pixel $\mathbf{z}$ can be written as:

$$\hat{I}(\mathbf{z}) = \frac{1}{Z} \sum_{i=t-H}^{t+H} \gamma^{|i-t|} \sum_{j=1}^{K} I(\mathbf{z}_{ij}) \exp\left\{ -\frac{D_w\Big(P(\mathbf{z}), P(\mathbf{z}_{ij})\Big)}{2\sigma_t^2} \right\}, \tag{5}$$

where Z is the normalization factor:

$$Z = \sum_{i=t-H}^{t+H} \gamma^{|i-t|} \sum_{j=1}^{K} \exp\left\{ -\frac{D_w\Big(P(\mathbf{z}), P(\mathbf{z}_{ij})\Big)}{2\sigma_t^2} \right\}, \tag{6}$$

and $D_w(\cdot, \cdot)$ is a weighted SSD function, summed over spatial, but not temporal, offsets:

$$D_w\Big(P(\mathbf{z}_1), P(\mathbf{z}_2)\Big) = \frac{1}{Z'} \sum_{\mathbf{u} \in [-s,s] \times [-s,s] \times 0} \Big(P(\mathbf{z}_1 + \mathbf{u}) - P(\mathbf{z}_2 + \mathbf{u})\Big)^2 \exp\left\{ -\frac{\|\mathbf{u}\|^2}{2\sigma_p^2} \right\}, \tag{7}$$

where $\sigma_p = \frac{s}{2}$, and Z' is a normalization constant. We set $\gamma = 0.9$ to control temporal decay. σ_t is related to the noise level in the video sequence, which will be discussed in the next subsection.

For a fixed number of iterations, the complexity of our denoising algorithm for a frame is $O(NHK \log K)$, where N is the number of pixels per frame, H is the temporal window size, and K is the number of approximate K-nearest neighbors. This is a significant reduction compared to $O(N^2)$ of the original NLM algorithm, since $K << N$ (typically $K = 10$ and $N = 640 \times 480$). Even if the search space of the original NLM algorithm is reduced to a 3D volume $M \times M \times (2H + 1)$ [11] (typically $M = 40$), the complexity remains as $O(NHM^2)$, still greater than our algorithm, which considers patches over the entire image lattice and $2H+1$ frames.

3.3 Noise estimation for adaptive noise removal

It is important to set the parameter σ_t appropriately in Eqn (5). Intuitively, when the noise level is low in the original sequence, we should set σ_t small to avoid over-smoothing, and when the noise level is high, we should set σ_t large to smooth out

noise. Instead of using a single-frame noise estimator as in [24], we propose a simple noise model based on optical flow.

Theoretically, as we warp frame I_{t+1} back to t according to the forward flow field $w^f(\mathbf{z})$, the difference between the warped frame and I_t should be the difference of independent noise. However, motion estimation can be unreliable especially at textureless regions and the brightness constancy assumption fails for occluded regions. Therefore, we introduce an outlier in noise estimation:

$$I_t(\mathbf{z}) = I_{t+1}\big(\mathbf{z} + w^f(\mathbf{z})\big) + \alpha_{\mathbf{z}} n_{\mathbf{z}} + (1-\alpha_{\mathbf{z}}) u_{\mathbf{z}}. \tag{8}$$

In the above equation, $n_{\mathbf{z}}$ is a pixel-wise Gaussian random variable: $E(n_{\mathbf{z}})=0,\ E(n_{\mathbf{z}}^2)=\sigma_n$, and $u_{\mathbf{z}} \sim U[-1,1]$ is a pixel-wise uniform random variable. These two random variables are balanced by weight $\alpha_{\mathbf{z}}$. Let $J_t(\mathbf{z}) = I_t(\mathbf{z}) - I_{t+1}(\mathbf{z} + w^f(\mathbf{z}))$. We use an expectation-maximization (EM) algorithm to estimate parameters:

1. Initialize $\sigma_n = 20$. Loop between step 2 and 3 until convergence.

2. (E-step) Evaluate $\alpha_{\mathbf{z}} = \dfrac{\exp\left\{-\dfrac{J_t(\mathbf{z})}{2\sigma_n^2}\right\}}{\exp\left\{-\dfrac{J_t(\mathbf{z})}{2\sigma_n^2}\right\} + \dfrac{1}{2}\sqrt{2\pi}\sigma_n}$.

3. (M-step) Estimate $\sigma_n = \sqrt{\dfrac{\sum_{\mathbf{z}} J_t(\mathbf{z})^2 \alpha_{\mathbf{z}}}{\sum_{\mathbf{z}} \alpha_{\mathbf{z}}}}$.

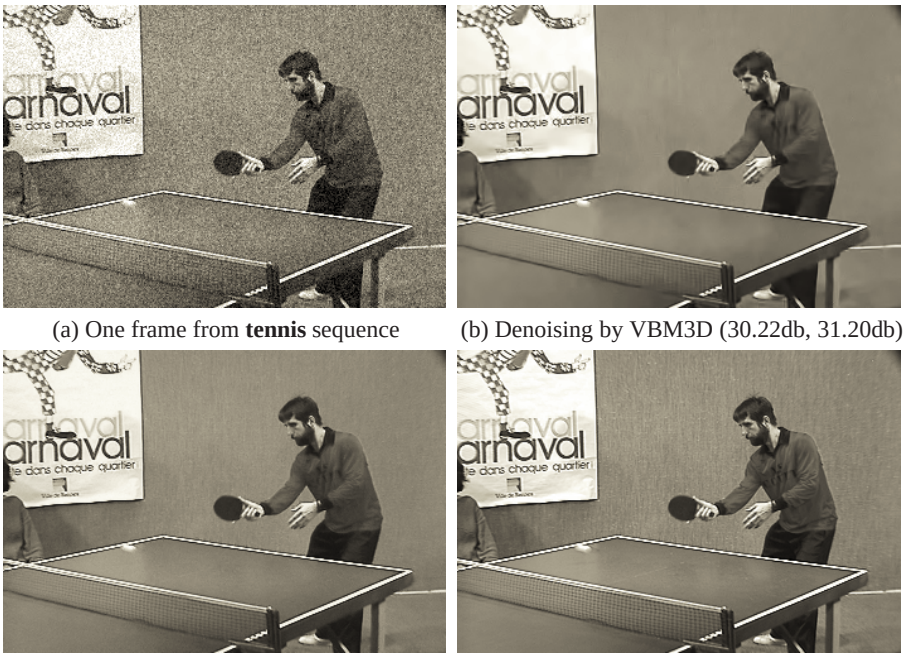
We perform this estimation for each of R, G and B channels independently.

The relationship between the noise level σ_n and scaling parameter σ_t in Eqn. (5) depends on K and H. Empirically, we have found that when $K=11$ and $H=5$ (which means that there are in total $K(2H+1)=121$ patches in total for NLM at one pixel), $\sigma_t=\sigma_n$ generates visually pleasing results.

4 Experimental Results

We conducted experiments to examine whether, in the framework we use, video denoising requires reliable motion estimation. In this section, we will first verify that our denoising algorithm is comparable with the state of the art [10] on synthetic sequences. Then, we will show that our algorithm outperforms the state of the art on real video sequences. **Please see denoised videos in the supplementary materials or the authors' websites. Please also use your monitor to view the results in the paper.**

Here are some implementation details of our algorithm. We use 7×7 patches, and $K=11$ nearest neighbors (including the patch itself), and 11 temporal frames ($H=5$) in our system. We allow 4 iterations of random K-nearest neighbor matching for each frame. The EM algorithm for noise estimation converges in about 10 iterations. For the optical flow algorithm, we used a coarse to fine scheme on image pyramid (with downsample rate 0.7) to avoid local minimum. The objective function is optimized through



(a) One frame from **tennis** sequence (b) Denoising by VBM3D (30.22db, 31.20db)

(c) Denoising by our system (30.21db) (d) Ground truth

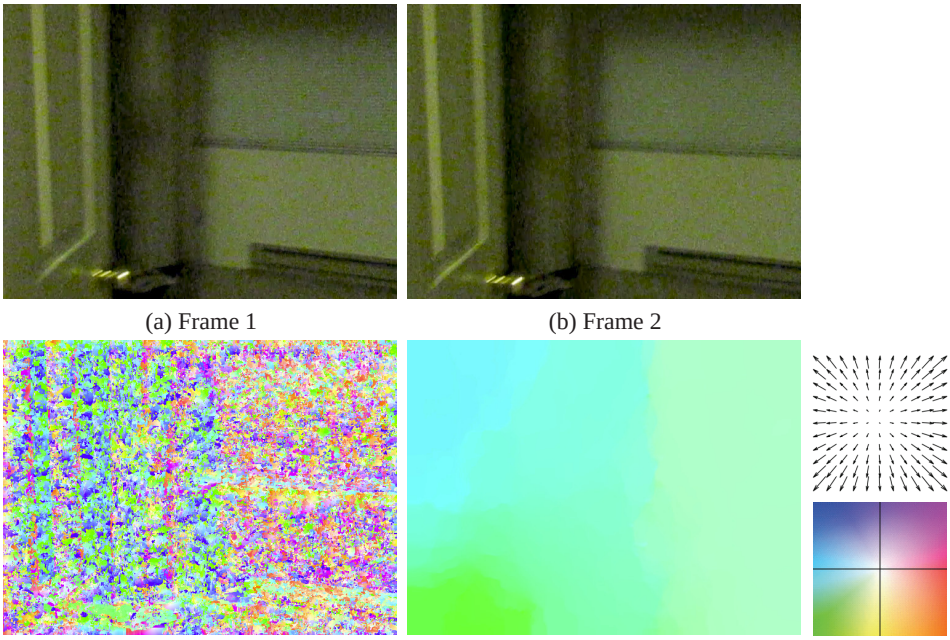
Fig. 4. For the **tennis** sequence, although the PSNR of our denoising system is slightly lower, the visual difference is subtle. VBM3D tends to generate smoother regions (but the background is over-smoothed), whereas our system preserves texture better (but the table is under-smoothed).

iterative reweighted least square (IRLS). More details of the flow estimation algorithm can be found in [25], and the source code is available online [1].

We first run our system on the **tennis** sequence with synthetically generated AWGN ($\sigma = 20$) to compare with existing methods. The average peak signal to noise ratio (PSNR) is 30.21db. We also downloaded the code from the BM3D webpage [2] for evaluation. In their MATLAB package, function `VBM3D.m` is used for gray-scale frames and `CVBM3D.m` is for color frames. We used `VBM3D` for the **tennis** sequence and `CVBM3D` for other sequences, but we will call it VBM3D in general. The first denoising step of VBM3D produces PSNR 30.22db, and the second step boosts it to 31.20db. The gain comes from re-matching patches from the denoising results at the first step and joint Wiener filtering at the second step, which are missing in our model. The backbone of our system is non-local means and therefore performs slightly worse in terms of PSNR. But the visual difference between ours and VBM3D is subtle, as shown in Figure 4.

We move on to a real video sequence named **room** captured by a Canon S90. This is a challenging sequence as the camera moves in between bright and dark rooms. We first examine the importance of regularization in motion estimation by comparing block

[1] `http://people.csail.mit.edu/celiu/OpticalFlow/`
[2] `http://www.cs.tut.fi/~foi/GCF-BM3D/index.html`



(a) Frame 1 (b) Frame 2

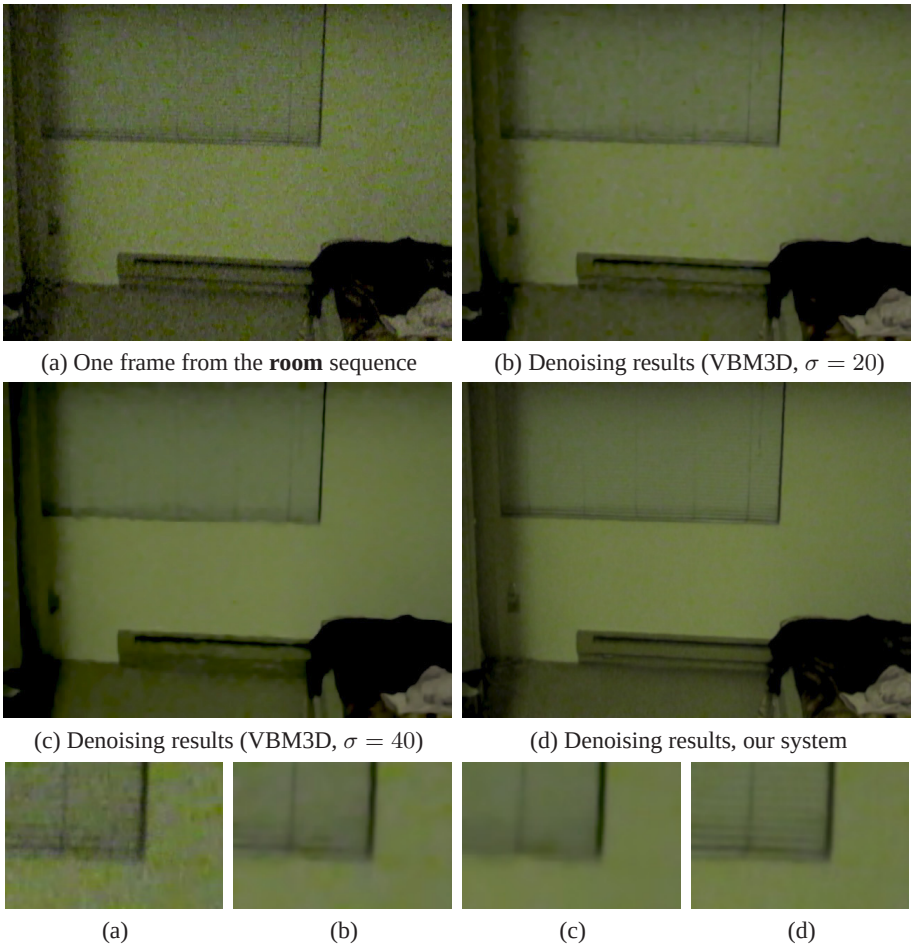
(c) Motion obtained by block matching (d) Motion obtained by optical flow (e) Flow code

Fig. 5. The right motion estimation algorithm should be chosen for denoising. For two consecutive frames in the **room** sequence, we apply both block matching [10] and optical flow [21] to estimate motion, shown in (c) and (d), respectively. We used the color scheme in [26] to visualize flow fields (e).

matching to the optical flow algorithm with spatial regularization. The motion estimation of one frame is shown in Figure 5, where motion vectors are visualized by the color scheme proposed in [26]. Clearly, spatially independent block matching in (c) is highly affected by the presence of structured noise. On the contrary, the optical flow with spatial regularization in (d) produces a smooth, discontinuity preserving temporal motion field that corresponds to the human perception of motion, and to the known smooth character of the optical flow induced by a camera moving through this piecewise smooth planar, static scene.

The quality of our motion estimation determines the quality of our video denoising. Because the code we downloaded from VBM3D does not allow input of frame-based noise intensities, we try two parameters $\sigma = 20$ and $\sigma = 40$ to denoise the **room** sequence, with results shown in Figure 6 (b) and (c), respectively. The result of our adaptive denoising system is shown in Figure 6 (d). Although there is no ground truth of this video, it is clear that our system outperforms VBM3D in both smoothing regions and preserving boundaries. The visual difference is more obvious when watching the videos in the supplementary materials.

Average PSNR over the video sequence has been used to measure video denoising qualities, but temporal coherence was not included in the quality assessment. We feel



(a) One frame from the **room** sequence (b) Denoising results (VBM3D, $\sigma = 20$)

(c) Denoising results (VBM3D, $\sigma = 40$) (d) Denoising results, our system

(a) (b) (c) (d)

Fig. 6. We run our system on the **room** sequence and compare the results with VBM3D [10]. Top: whole frames; bottom: the blowup view of a region. Our system outperforms VBM3D in both smoothing regions and preserving boundaries. **Please view this figure on your monitor.**

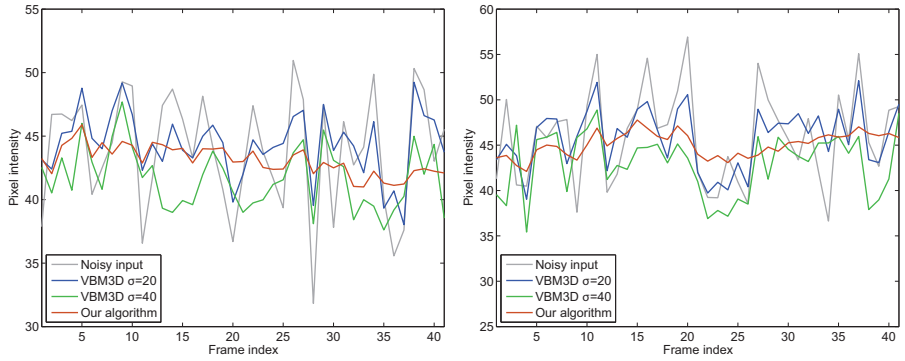


Fig. 7. Temporal smoothness of different denosing algorithms. We measure pixel intensities along motion paths over frames. Two motion paths are shown here. Our system (red curve) has the least amount of temporal fluctuation.

that temporal coherence is indeed vital to evaluate video denoising algorithms. For this purpose, we downloaded the human-assisted motion annotation tool [27] to annotate the ground-truth motion of the **room** sequence. Using the annotated motion we can analyze how pixel intensities change over time for different denoising algorithms. Two exemplar motion paths are plotted in Figure 7, and the average standard deviation for each of the RGB channels is listed in Table 1. Clearly, our system has overall the least temporal fluctuation, which we feel is crucial for visual quality.

Std. Dev.	Input data	VBM3D $\sigma = 20$	VBM3D $\sigma = 40$	Our algorithm
Red	4.20	2.22	1.52	1.23
Green	3.81	1.78	1.45	1.13
Blue	9.55	5.77	2.81	2.91

Table 1. The average standard deviation along motion paths is measured for different algorithms at different RGB channels. Our system has overall the least temporal fluctuation.

Lastly, we run our system on another video sequence with real noise, **baby**, a 720P HD video clip captured by SONY HDR-XR150. One input frame and denoised frame are shown in Figure 8. Our video denoising system is able to remove the structured noise and preserve image details without introducing artifacts. This example shows the broad applications for reliable video denoising algorithms.

5 Conclusion

We argue that robust motion estimation is essential for high-quality video denoising, especially in the presence of real, structured noise. Based on the non-local means framework, we introduce an efficient, approximate K-nearest neighbor patch matching algorithm that can search for similar patches in a neighborhood as large as the entire image. This random matching algorithm significantly reduces the complexity of classical NLM methods. A robust optical flow algorithm with spatial regularity was used to estimate temporal correspondence between adjacent frames. The spatial regularity is the key to robust motion estimation in the presence of structured noise. We use the temporal correspondence to enlarge the set of supporting patches over time and to ensure temporal coherence. Experimental results show that our system is comparable with the state of the art in removing AWGN, and significantly outperforms the state of the art in removing real, structured noise. Our system is easy to implement, with broad applications in digital video enhancement.

References

1. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381** (1996) 607–609
2. Mairal, J., Elad, M., Sapiro, G.: Multiscale sparse image representation with learned dictionaries. In: IEEE International Conference on Image Processing (ICIP). (2007)

(a) (b)

Fig. 8. Removing realistic video noise has broad applications. For example, we can turn a noisy HD home video (a) to a high-quality, noise-free video (b), which can be pleasantly played on an HDTV. Please see the **baby** video in the supplementary materials. **Please view this figure on your monitor.**

3. Yang, J., Wright, J., Huang, T., Ma, Y.: Image super-resolution as sparse representation of raw image patches. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2008)
4. Simoncelli, E.P., Adelson, E.H.: Noise removal via Bayesian wavelet coring. In: IEEE International Conference on Image Processing (ICIP). Volume I. (1996) 379–382
5. Portilla, J., Strela, V., Wainwright, M.J., Simoncelli, E.P.: Image denoising using scale mixtures of gaussians in the wavelet domain. IEEE Transactions on Image Processing (TIP) **12** (2003) 1338–1351
6. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2005)

7. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3d transform-domain collaborative filtering. IEEE Transactions on Image Processing (TIP) **16** (2007)
8. MPEG: Mpeg-2 video encoding (h.262) (2006) http://www.digitalpreservation.gov/formats/fdd/fdd000028.shtml.
9. Lee, C.U., Pian, D.: Interframe video encoding and decoding system (1996) US Patent 5,576,767.
10. Dabov, K., Foi, A., Egiazarian, K.: Video denoising by sparse 3d transform-domain collaborative filtering. In: European Signal Processing Conference EUSIPCO. (2007)
11. Buades, A., Coll, B., Morel, J.M.: Nonlocal image and movie denoising. International Journal of Computer Vision (IJCV) **76** (2008) 123–139
12. Roth, S., Black, M.J.: Fields of experts: A framework for learning image priors. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2005)
13. Weiss, Y., Freeman, W.: What makes a good model of natural images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2007)
14. Elad, M., Aharon, M.: Image denoising via learned dictionaries and sparse representation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2006)
15. Tschumperlé, D.: Fast anisotropic smoothing of multi-valued images using curvature-preserving PDE's. International Journal of Computer Vision (IJCV) **68** (2006) 65–82
16. Liu, C., Szeliski, R., Kang, S.B., Zitnick, C.L., Freeman, W.T.: Automatic estimation and removal of noise from a single image. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **30** (2008) 299–314
17. Buades, A., Coll, B., Morel, J.M.: Denoising image sequences does not require motion estimation. In: IEEE International Conference on Advanced Video and Signal Based Surveillance. (2005)
18. Cho, T., Butman, M., Avidan, S., Freeman, W.: The patch transform and its applications to image editing. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2008)
19. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. In: Proceedings of ACM SIGGRAPH. (2009)
20. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **30** (2008) 1068–1080
21. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/Kanade meets Horn/Schunk: combining local and global optical flow methods. International Journal of Computer Vision (IJCV) **61** (2005) 211–231
22. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence. (1981) 674–679
23. Horn, B.K.P., Schunck, B.G.: Determing optical flow. Artificial Intelligence **17** (1981) 185–203
24. Liu, C., Freeman, W.T., Szeliski, R., Kang, S.B.: Noise estimation from a single image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2006) 901–908
25. Liu, C.: Beyond pixels: exploring new representations and applications for motion analysis. PhD thesis, Massachusetts Insitute of Technology (2009)
26. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. In: Proc. ICCV. (2007)
27. Liu, C., Freeman, W.T., Adelson, E.H., Weiss, Y.: Human-assisted motion annotation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2008)