Fast Burst Images Denoising

Ziwei Liu[1]* Lu Yuan† Xiaoou Tang* Matt Uyttendaele‡ Jian Sun†

*The Chinese University of Hong Kong †Microsoft Research ‡Microsoft Research Technologies

Figure 1: *Top-left: a burst of noisy images (10 frames with image size* 3072×1728*) by a smartphone. Bottom-left: the running time (sec.) of different denoising methods. Right: comparison of two close-up views. (a) input images (b) spatial-temporal filtering [Bennett and McMillan 2005] (c) BM4D [Maggioni et al. 2013] (d) optical flow [Liu 2009] + median (e) lucky imaging [Joshi and Cohen 2010] (f) our method. Our method produces a clean, ghost-free image with fine details. More importantly, our method is significantly faster than other methods.*

Abstract

This paper presents a fast denoising method that produces a clean image from a burst of noisy images. We accelerate alignment of the images by introducing a lightweight camera motion representation called *homography flow*. The aligned images are then fused to create a denoised output with rapid *per-pixel* operations in temporal and spatial domains. To handle scene motion during the capture, a mechanism of selecting *consistent pixels* for temporal fusion is proposed to "synthesize" a clean, ghost-free image, which can largely reduce the computation of tracking motion between frames. Combined with these efficient solutions, our method runs several orders of magnitude faster than previous work, while the denoising quality is comparable. A smartphone prototype demonstrates that our method is practical and works well on a large variety of real examples.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement—Smoothing

Keywords: denoising, burst images, homography flow, ghost-free

Links: DL PDF

[1]This work was done when Ziwei was an intern at MSR Asia.

1 Introduction

Burst, a shooting mode in most cameras, allows multiple shots to be captured in a quick succession by either pressing or holding the shutter button. It is designed to allow selection of the best shot or record the motion. Recently, burst capturing has become ubiquitous in many hand-held imaging devices (*e.g.*, smartphone, compact and DSLR cameras). For example, iPhone 5s supports a burst of up to 10 shots per second.

The burst mode has been successfully exploited in computation photography for reducing blur [Cai et al. 2009], or improving shadow/highlight details [Reinhard et al. 2010], or increasing resolution [Farsiu et al. 2004], or clarity [Joshi and Cohen 2010], or depth of the field [Jacobs et al. 2012].

In this paper, we present a practical solution for "burst images denoising" - turning a burst of noisy images (typically captured in a low-light condition) into a single clean image, as shown in Figure 1. This problem is not new. It has been studied in the context of multiple images/video denoising [Buades et al. 2010; Liu and Freeman 2010; Zhang et al. 2009]. But we focus on practicality - our goal is to design a highly efficient method while producing a high-quality result so that the algorithm can be run on a mobile device with limited computational resources.

A practical approach needs to tackle two challenges. First is efficiency. The state-of-the-art methods heavily rely on optical flow or patch matching to establish temporal and spatial correspondence, which is unacceptably slow. Second is quality. Fast methods like averaging or filtering [Tomasi and Manduchi 1998] are insufficient on both noise reduction and avoiding "ghost" artifacts caused by either camera motion (by hand shake) or scene motion (by dynamic objects). Moreover, even some complicated methods are also fragile in the presence of strong noise or complex dynamic motion.

We propose a fast noise reduction method that produces a clean image from a burst of images. The high speed of our method is enabled by introducing three accelerating steps. In the first step, we use a lightweight, parametric motion representation - *homog-*

raphy flow - to model the motion caused by camera movements. This representation is inspired by the recent multiple homographies model [Grundmann et al. 2012; Liu et al. 2013] for video stabilization. Since estimating homography flow only requires spare feature matching, this step is both efficient and robust to noise.

In the second step, we handle the scene motion by identifying *consistent pixels* (*i.e.*, pixels with similar colors) along the temporal axis from all aligned images (by the first step) per pixel location. These selected consistent pixels are used in our temporal pixel fusion (in the third step) by averaging. Thus, we can generate ghost-free results while avoiding complex motion tracking on dynamic objects, which is too slow or too difficult. The idea was successfully applied in recent HDR deghosting [Granados et al. 2013]. We extend this idea to find as many consistent pixels as possible at every pixel location for the purpose of better denoising.

In the third step, we apply temporal and multiscale pixel fusions in succession to obtain the denoised result. The temporal fusion is based on a simple, optimal linear estimator. The multiscale fusion is complementary to temporal fusion and further enables significant denoising. Meanwhile, the whole step is also very efficient by design because it only involves pixel-wise operations.

We have evaluated our algorithm on a variety of real data. In the presence of moderate or strong noise, our algorithm performs on par with state-of-the-art multi-image denoising methods (*e.g.*, VBM3D [Dabov et al. 2007a], BM4D [Maggioni et al. 2013]). Furthermore, our algorithm is two or three orders of magnitude faster. Figure 1 shows a comparison.

2 Related Work

Single image denoising has great progresses in recent decades. Representative methods include bilateral filtering [Tomasi and Manduchi 1998], wavelet (GSM) [Portilla et al. 2003], Field-Of-Expert [Roth and Black 2005], non-local means [Buades et al. 2005], BM3D [Dabov et al. 2007b] and so on. To improve the efficiency, a few fast variants have been proposed, such as fast bilateral filtering [Paris and Durand 2009], Gaussian kd-trees [Adams et al. 2009], and geodesic paths [Chen et al. 2013]. Most recently, Levin et al. [2011] pointed out that single image denoising may be approaching its performance limit. NoiseBrush [Chen et al. 2009] provided a interactive way for further quality improvement.

Multiple image denoising is superior to single image denoising because of its use of more information. Some denoising techniques have been successfully used on burst images [Tico 2008; Buades et al. 2009; Joshi and Cohen 2010], videos [Bennett and McMillan 2005; Liu and Freeman 2010; Dabov et al. 2007a; Chen and Tang 2007], multiple-view images [Zhang et al. 2009], and volumetric MRI data [Maggioni et al. 2013].

Estimating camera motion. Optical flow [Brox et al. 2004] is the most general representation for establishing correspondences between frames. Recent work [Liu and Freeman 2010] showed its importance in video denoising. But optical flow itself has difficulties with occlusion/large displacement, and is fragile to noise. Patch matching is more robust to noise and has been widely used in multiple image processing [Tico 2008; Buades et al. 2009; Zhang et al. 2009; Maggioni et al. 2013; Sen et al. 2012; Kalantari et al. 2013]. However, in the presence of strong noise, both nonparametric methods degrade rapidly. Camera motion in burst mode is similar to the motion studied in video stabilization. Recent work [Grundmann et al. 2012; Liu et al. 2013] demonstrated the success of using a spatially-variant homography for the camera motion. In this work, we use a similar but more lightweight parametric motion representation - homography flow.

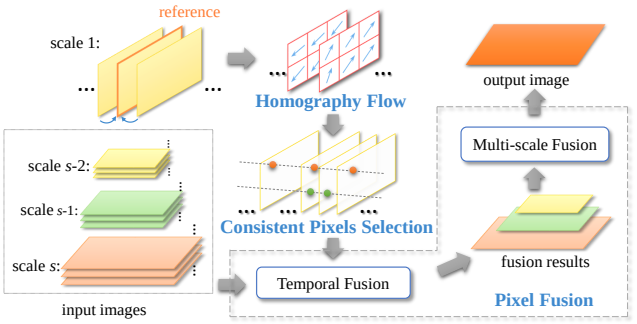

Figure 2: *Algorithm pipeline.*

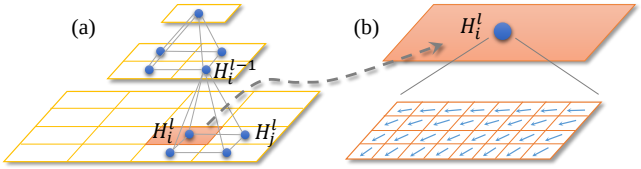

Figure 3: *(a) Pyramid homography graph. (b) homographies (at the finest level) are discretized to obtain a per-pixel homography flow field.*

Handling scene motion. Since optical flow or patch matching is an intrinsically hard problem, recent work [Gallo et al. 2009; Granados et al. 2013] in HDR reconstruction bypasses the motion estimation by finding a consistent subset of colors for every pixel to reconstruct a ghost-free image. Granados et al. [2013]'s consistency test relies on accurate estimation of the noise distribution, which may require complex calibration and super-pixels computation. We were inspired by this idea and extend it for image denoising.

Multiscale denoising is an effective way to exploit cross-scale similarity for noise reduction. Recently, Zontak *et al.* [2013] proposed a directional pyramid technique to find corresponding patches across scales, which produces state-of-the-art results. Zhang and Gunturk [2008] extended the bilateral filter in a multiscale framework. We use a pyramid-based pixel fusion method to improve the result quality.

3 Algorithm

Figure 2 is our algorithm pipeline. We first build Gaussian pyramids[1] of all noisy images and set the midmost frame as the reference frame by default. Then, we estimate *homography flow* (in Section 3.1) to represent the motion (by camera) between the reference frame and any of the other frames. Across the aligned images (by homography flow), at every pixel location, we select a set of *consistent pixels* to handle scene motions (in Section 3.2) or possible small misalignment caused by the homography flow. Finally, we apply a *pixel fusion* (in Section 3.3) to aggregate consistent pixels at all scales for producing the final result.

3.1 Homography Flow (for Camera Motion)

Pyramid homography graph. We represent the motion between two frames through a pyramid homography graph, as shown in Figure 3 (a). The coarse level node provides robustness while the fine level node helps produce details. Note that independently estimating the homography at each node is unreliable because some nodes may have insufficient matched features. Next, we introduce a coarse-to-fine optimization to robustly obtain accurate results.

[1]The long side of image at the coarsest scale is no more than 400 pixels.

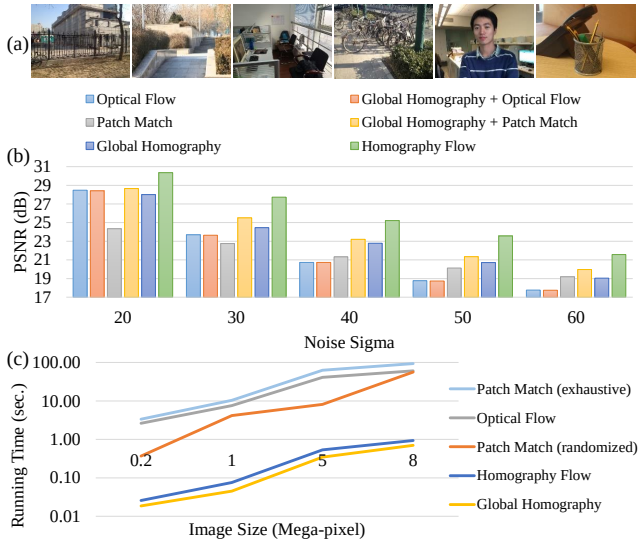

Figure 4: *Comparisons of registration errors and time cost for various motion models. (a) sample images. (b) registration errors measured in PSNR, which is computed on registered clean images, but the motion is estimated from the noisy images. (c) running time (sec.) of different methods. We use the randomized patch match [Barnes et al. 2009] source code for acceleration, but it is still* $1 \sim 2$ *orders slower than homography flow.*

Optimization. We perform a level-by-level optimization starting from a global homography at the coarsest level ($l = 0$). Let H_i^l be the homography of node i at level l , and $\{H_j^l\}$ be its 4 neighboring homographies at the same level. We estimate $\{H_i^l\}$ by minimizing the following energy function:

$$\{\hat{H}_i^l\} = \arg\min_{\{H_i^l\}} \sum_i \left(\left\| H_i^l - R_i^l \right\|^2 + \lambda \sum_j \left\| H_i^l - H_j^l \right\|^2 \right), \tag{1}$$

where λ (by default, $\lambda = 0.1$) controls the amount of spatial regularization enforced by the second smoothness term.

In the first data term, $R_i^l = best(\hat{H}_i^{l-1}, F_i^l)$ is selected from two candidates: one is its parent homography $\hat{H}_i^{l-1}$ at the level $l-1$, the other is its own estimated homography F_i^l (using all matched features within the grid of node i). The basic idea is to use $\hat{H}_i^{l-1}$ as backup when we cannot reliably compute F_i^l. In our implementation, we pick $\hat{H}_i^{l-1}$ if the feature number in the grid of i is insufficient (< 8) or the rigidness [Hartley and Zisserman 2003] of F_i^l is too weak[2]; otherwise, we choose the best model which has lower matching errors of all features within the grid.

Because the objective function (1) is quadratic, we can obtain the global optimum by a Jacobi solver [Bronshtein and Semendyayev 1997]. The form of our motion model is similar to a mesh-based homography [Liu et al. 2013]. But our coarse-to-fine optimization is more efficient. For 500 features, our method takes 5 ms per frame while the mesh-based homography requires 50 ms per frame.

Homography flow. Since the pyramid homography graph is a parametric representation, we require an image warping or coordinate transformation in the later denoising step. But such operations for all pixels (in all frames, at all scales) are very expensive. To address this critical issue in our application, we discretized the homography graph (the finest level only) to obtain per-pixel translation vector - *homography flow*.

As shown in Figure 3 (b), we compute the translation vector by mapping each pixel from one frame to another frame according to the homography graph. Finally, the estimated homography flows are scaled accordingly and rounded off for use at other scales.

Algorithm validation. We evaluate global homography, optical flow, patch match, and our homography flow on a set of burst images. We asked four different subjects to capture 20 sets of clean burst images (with low ISO, under good lighting conditions). Then we added Gaussian noise with different standard deviations (from 20 to 60) to synthesize 100 sets of noisy burst images. To register these noisy images, we compare six algorithms: global homography, optical flow [Liu 2009], global homography + optical flow, patch match (exhaustive search), global homography + patch match, and our homography flow. We compute the PSNR to measure the difference between registered clean image pairs. Figure 4 (b) shows the results.

From the results, we can observe that two optical flow based methods perform well when noise level is small ($\sigma = 20$). But when the noise level increases, they degrade more quickly than others; global homography can improve patch match but not optical flow. We believe the reason is that the coarse-to-fine mechanism in the optical flow has already handled the global motion. Our homography flow is consistently the best at all noise levels.

Figure 4 (c) further shows the running time of these methods on various image sizes. Since global homography and our homography flow only rely on sparse feature detection and matching, they both outperform patch match (exhausive search or even randomized search [Barnes et al. 2009]) and optical flow [Liu 2009] in speed. There is only a small margin between the two running time curves (of global homography and our homography flow), which indicates that our pyramid optimization is very efficient.

Efficient implementation. The bottleneck in this step is sparse features extraction and matching. In our implementation, we work on the coarse scale ($s = 1$) in the pyramids of luminance channel for efficiency and robustness (to noise). Compared with original-scale implementation, the time cost is greatly reduced (by a factor of 6, on average) and the PSNR (for measuring registration error) is improved by 0.005dB to 0.045dB for various noise sigma (from 20 to 60). Specially, we use the Harris corner detection [Harris and Stephens 1988] and 128-bit BRIEF descriptor [Calonder et al. 2010], which can achieve real-time performance even on a mobile phone. We reject incorrectly matched features using local RANSAC [Grundmann et al. 2012].

In addition, we estimate homography flow in each non-overlapped block (8×8 pixels) instead of per pixel. All pixels in each block share the same translation vector, which is computed by mapping the block center between two frames. The approximation only slightly scarifies the quality (by nearly 0.01dB in PSNR), but accelerates pixels mapping by 2.5 times.

3.2 Consistent Pixels Selection (for Scene Motions)

Consistent pixels. To handle scene motion, we borrow a simple idea from HDR deghosting [Granados et al. 2013] to avoid complex motion tracking. At every pixel location (on a reference frame), we identify a set of *consistent pixels* on a 1D profile (traced by the estimated homography flow) across all images for temporal pixel fusion. Different from HDR deghosting, the purpose of selecting consistent pixels is not only avoiding ghost artifacts (caused by dynamic motion and small frame misalignment), but also finding as many consistent pixels as possible for denoising.

[2]Shear of homography > 1.25 or modulus of perspective > 0.1. These parameters are empirically set and fixed in all experiments.

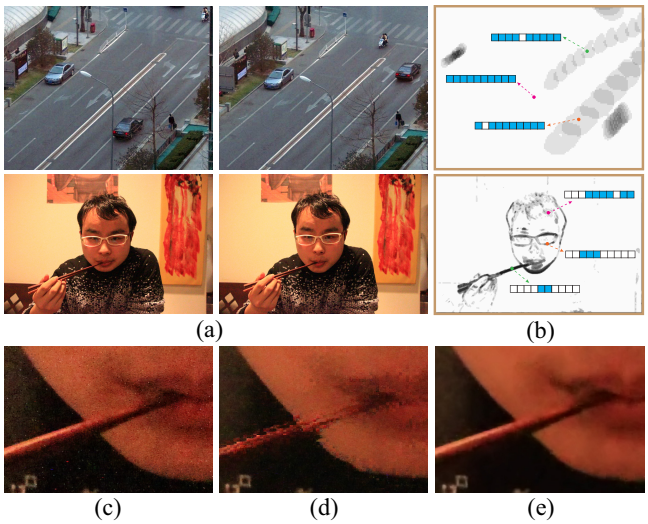

Figure 5: *Consistent pixels selection for rapid motions (*e.g.*, cars, on top) and small motions (*e.g.*, head and hand, on bottom). (a) two of the input frames. (b) map recording the selected frame indices (blue cells) for each pixel. (c) temporal fusion result by only reference-based pixels selection. (d) temporal fusion result by only median-based pixels selection. (e) temporal + multiscale fusion result from a combined strategy.*

There are two methods which separately satisfy one of two goals. One is reference-based: we bi-directionally trace the profile from the pixel at the reference frame and collect consistent pixels until the accumulated pixel difference exceeds a threshold τ. The other is median-based: we collect pixels consistent to the median (below the same threshold τ) of all pixels on the profile. The reference method can guarantee a ghost-free result. But for a pixel on a dynamic object, we may get insufficient samples for denoising (shown on Figure 5 (c)). The median method collects more consistent pixels but might lead to ghosting because the median may happen to be a color on the moving object (shown on Figure 5 (d)).

Combined strategy. We propose a simple combination strategy of both methods: for each pixel (on the reference frame), we compute two sets of consistent pixels $\{\mathcal{M}, \mathcal{R}\}$ separately by the median and the reference methods. $\mathcal{M}$ (or $\mathcal{R}$) records the frame indices of consistent pixels for every pixel. If the reference frame belongs to $\mathcal{M}$, we take the union of $\mathcal{M}$ and $\mathcal{R}$ as the final result because both methods agree. Otherwise, we choose the median result if it is reliable (i.e., the size of $\mathcal{M}$ is more than half of the frame number), and choose the reference result if it is unreliable. To further reduce the chance of ghosting (by enforcing spatial coherence), we do not measure reliability pixel by pixel. Instead, we find all connected-components of undecided pixels (where the reference frame does not belong to $\mathcal{M}$), and then determine the reliability of each connected component as a whole, by majority voting.

After the combination, we obtain the sets of consistent pixels for all pixels. To further make the combination seamless, we run a 3×3 morphological (majority) filter [Gonzalez and Woods 2007] on the stack of the indices of consistent pixels, frame-by-frame. Figure 5 (a-b) shows two real examples and the maps which record the number of consistent pixels at every pixel location.

Efficient implementation. The consistent pixels are also selected at the coarse scale (*i.e.*, $s = 1$) in the pyramids for the purpose of enabling fast computation and detecting motion at a relatively clean scale. The approximation could achieve 98.7% outlier detection rate as operating at the original scale at the expense of half the time. Other scales just reuse the indices of computed consistent pixels by upsampling or downsampling. In both median and reference based methods, we use patch (5×5) difference instead of single pixel difference and use the threshold [3] $\tau = 10$. We use integral image [Viola and Jones 2001] to compute patch differences more efficiently.

3.3 Pixels Fusion

Given consistent pixels for each pixel at all scales, we fuse all of them in a temporal and multi-scale fusion. We keep our fusion as simple as possible, while being able to significantly denoise.

Temporal fusion. Suppose $\{x_t\}$ are consistent pixels at a pixel location (at a certain scale), where x_t is pixel color from the t-th frame. We compute the fusion result $\hat{x}$ by a linear minimum mean squared error (LMMSE) estimator, which is widely used in previous work (*e.g.*, [Zhang and Wu 2005]) for optimal unbiased denoising:

$$\hat{x} = u + \frac{\sigma_c^2}{\sigma_c^2 + \sigma^2}(x_t - u), \tag{2}$$

where u is the mean of all consistent pixels $\{x_t\}$. The variance σ_c^2 of true pixels is approximated by $\max(0, \sigma_t^2 - \sigma^2)$. σ_t and σ are the standard deviation of $\{x_t\}$ and noise.

The LMMSE estimator can help us adaptively handle outliers. Some severely misaligned pixels occurring at discontinuities of depths (our homography flow is more suitable for spatially smooth depth variations) or subtle residual moving pixels at a fine scale (our scene motion detection is performed at a coarse scale) wold make the variance of $\{x_t\}$ much larger than the noise variance. Thus, our fusion result would be $\hat{x} \to x_t$ (no denoising). Otherwise, the result $\hat{x}$ has a value close to the mean u of $\{x_t\}$.

The temporal fusion runs independently at every scale. Next, we describe how to aggregate results in all scales.

Multi-scale fusion. We aggregate the results from top to bottom, in a point-wise manner. Let x^s and x^{s-1} be temporal fusion results at two adjacent scales s and $s-1$. We update the result x^s by:

$$\hat{x}^s = \omega \times x^s + (1 - \omega) \times (x^{s-1}) \uparrow, \tag{3}$$

where $\uparrow$ is a bilinear upscale operator. $\omega = \sqrt{m/N}$ is an adaptive fusion weight.

N is the total frame number and m is the number of inliers (x_t is an inlier when $|x_t - \hat{x}| < 3\sigma_t$, where σ_t is the standard deviation of $\{x_t\}$) identified by the temporal fusion. A larger ω means we have more consistent pixels at the current scale and we should trust current estimation more; otherwise, we should borrow more from the parent scale. Furthermore, the multi-scale processing is effective for handling non-gaussian types of noise (*e.g.*, splotches [Chatterjee et al. 2011]) in the real imaging pipeline.

The above fusion does not exploit spatial information which plays a central role in single image denoising algorithms [Zontak et al. 2013]. Here, we replace the temporal fusion result x^s in Equation (3) with a very fast filtering in the spatial domain:

$$x^s = p_{tex} \times f(x^s) + (1 - p_{tex}) \times (x^{s-1}) \uparrow, \tag{4}$$

where the filtering operator $f(x^s)$ is a directional spatial pixel fusion. We find all spatially consistent pixels along the most probable

[3] Since we detect motion at a coarser and relatively clean scale, we can use a constant threshold instead of an adaptive threshold, which may require complex noise modeling. We find that it works well in our experiments.

edge direction within a 5×5 window. Then a LMMSE estimator described in Equation (2) is used on these pixels. For efficiency, we only apply the spatial fusion on texture pixels ($p_{tex} > 0.01$).

The textureness probability p_{tex} is computed by a sigmoid function $1/(1 + \exp(-5 \times (g/\sigma - 3)))$, in which g is the maximum absolute difference between the pixel and its 4 neighbors, and σ is the estimated standard deviation of noise. For efficiency, we estimate σ by computing the standard deviation of pixels differences between the median image and the reference image within the flat (non-textured) areas. On these areas, the median image (generated in the coarse scale) is a good approximation to a clean version of the reference image. For real noise, we use an approach similar to [Liu et al. 2008] to divide the illuminance into 10 discrete bins and estimate the corresponding σ for each bin.

Extension to patch. The idea of combining temporal and multi-scale fusion can also be extended to the patch level for better denoising quality. Different from point-wise fusion, the LMMSE estimator for patches is applied in the frequency domain, which is similar to the Wiener filter used in the transform domain [Dabov et al. 2007b]. In addition, the patch-based LMMSE estimator provides overlapping estimates for every pixel, which need patch aggregation (along the temporal axis or spatial edge direction) to get the final fusion result.

Algorithm validation. We perform a quantitative evaluation on a synthetic data set. The ground-truth clean images come from 68 images of BSD300 [Martin et al. 2001]. To simulate the camera motion, we reuse the estimated global homographies from the real data (Figure 4) and randomly apply them on one of the clean images to generate a burst of images (10 frames). Then, we add Gaussian noise with various noise levels ($\sigma = 20 \sim 60$). Note that our synthetic data set ignores many key factors in real data: parallax, non-Gaussian noise, and non-rigid object motion. Ignoring these key factors may lead to incomplete conclusions. However, we still provide a preliminary evaluation here for reference and to help us gain a better understanding.

Figure 6(a) shows the average PSNRs of: average (baseline), optical flow + median filtering, VBM3D [Dabov et al. 2007a], BM4D [Maggioni et al. 2013], our method (with pixel fusion), and our method (with patch fusion). We applied the same global homography to all methods for better results.

Overall, our method (with pixel fusion) performs comparably to VBM3D and BM4D (two state-of-the-art denoising methods) and better than averaging and optical flow at all noise levels. When the noise level increases, our method performs slightly worse than BM4D, but VBM3D drops more quickly than ours. This evaluation partially demonstrates the true power of our method in the presence of real camera motion and registration error. Keep in mind that our method is 2-3 orders of magnitude faster than VBM3D, optical flow, and BM4D. Figure 6(b) shows the running time of these methods on different image sizes.

In addition, we can achieve the best results by extending our fusion to the patch level. Compared with two patch-based methods (VBM3D and BM4D), our patch-based fusion is still efficient (1-2 orders of magnitude faster). Figure 7 also demonstrates that patches perform better in both temporal and multi-scale fusion than pixels. The conclusion is consistent to previous work since patches can usually use more spatially correlated information than pixels. More interestingly, multiscale fusion as complementary to temporal pixel fusion plays an important role in our pixel-based method. It helps greatly reduce the gap between our pixel-based method and our patch-based method.

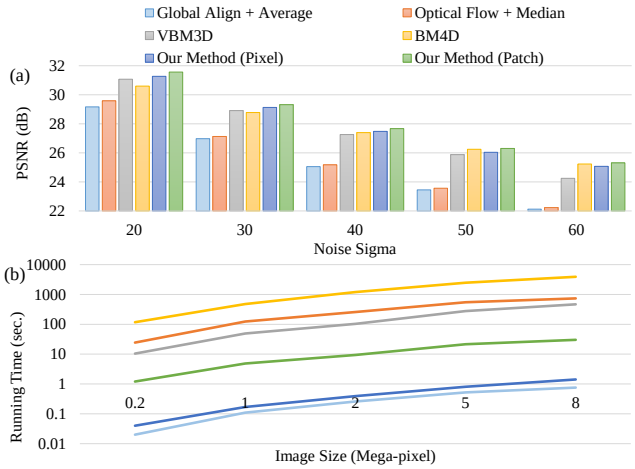

Figure 6: *PSNR of different methods on the synthetic data. For VBM3D [Dabov et al. 2007a] and BM4D [Maggioni et al. 2013], we apply a pre-warp using the same estimated global homography for better results.*

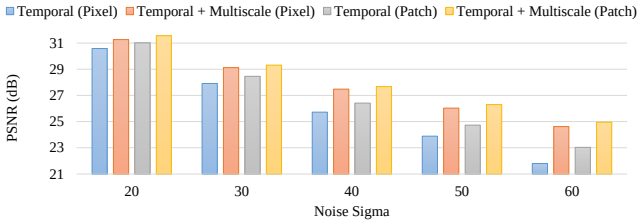

Figure 7: *PSNR comparison of our different fusions.*

4 Experiments

We acquired 20 sets of burst images on various content with 5 cameras, including 3 mobile phones, 1 DSLR camera, and 1 compact camera. Each burst contains 10 shots. All our results are generated with a set of fixed parameters and by pixel-based fusion. All original sequences and more results are provided on our webpage[4].

4.1 Comparisons

We compare our method with three point-wise methods (spatial-temporal filtering [Bennett and McMillan 2005], lucky imaging [Joshi and Cohen 2010], and optical flow [Liu 2009] + temporally median filtering), and two state-of-the-art patch-based methods (VBM3D [Dabov et al. 2007a] and BM4D [Maggioni et al. 2013]). The former two are based our own implementations, and optical flow and the latter two are from the authors. For all methods, we apply the same global homography estimated by our method to help them to obtain more reliable correspondences. Since some algorithms require a known noise variance, we try all possible noise levels and choose the result with the best visual quality through a balanced tradeoff between detail recovery and noise reduction.

Static scene. The example in Figure 8 was captured by a HTC 802d Android phone. The motion is mainly caused by camera movement. The challenges in this case are on how to remove strong noise in the sky and recover building structures. As we can see, spatial-temporal filtering, VBM3D, and BM4D still leave certain noises in flat regions (*e.g.*, sky area). The building structures (in Figure 8(b)(f)) are not well restored by VBM3D and BM4D. This is because, in the presence of structure noise, either the spatial-temporal bilateral

[4] http://personal.ie.cuhk.edu.hk/∼lz013/projects/BurstDenoising.html

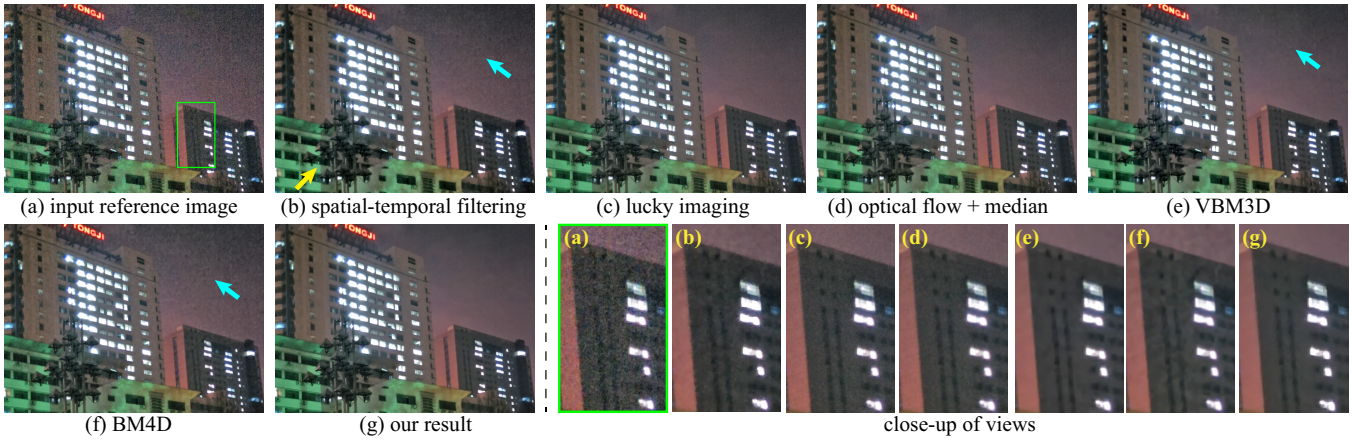

Figure 8: *Static scene.*

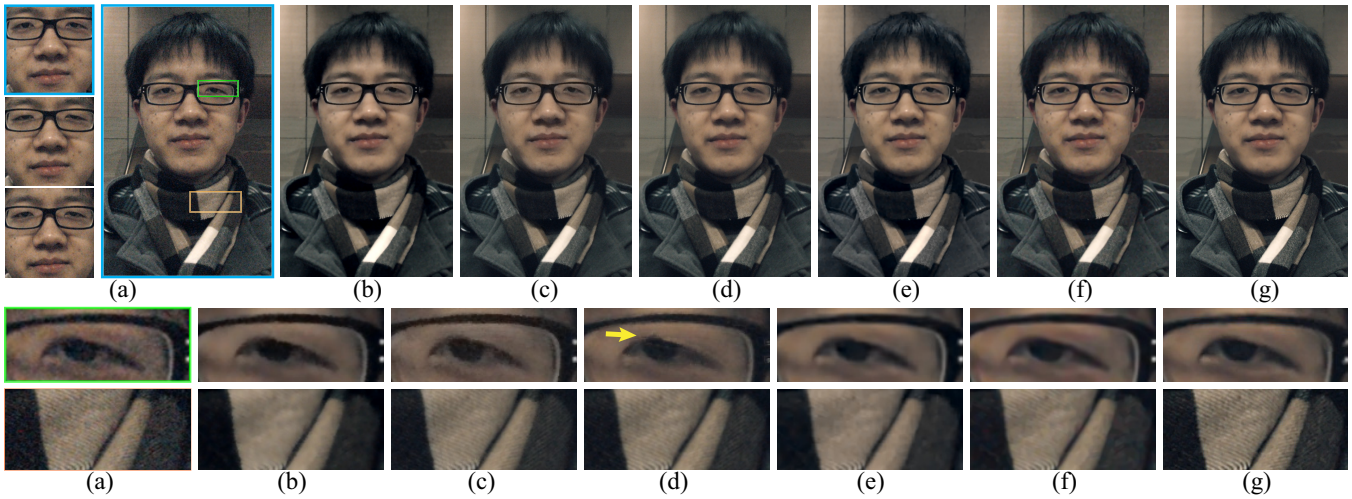

Figure 9: *Portrait with small motion. (a) input images. (b) spatial-temporal filtering [Bennett and McMillan 2005] (c) lucky imaging [Joshi and Cohen 2010] (d) optical flow [Liu 2009] + median, (e) VBM3D [Dabov et al. 2007a], (f) BM4D [Maggioni et al. 2013], (g) our result.*

filter or patch matching has the risk to find mismatched correspondences which will eventually lead to undesired results. Overall, the results by optical flow, lucky imaging, and ours are comparable, with our result being slightly cleaner.

Portrait with small motion. This is a typical scenario for taking a portrait photo in dark lighting. The example in Figure 9 was recorded by a JVC GC-PX10 camera. Spatial-temporal filtering, lucky imaging, and the optical flow method produced "staircase" artifacts around the eyes (Figure 9 (b)(c)(d)). This is because of small non-rigid motion of the subject. VBM3D and BM4D do not have this issue but blurred fine details on scarves (Figure 9 (e)(f)). Our result achieves the best on both kinds of regions.

Complex scene motion. Figure 10 and Figure 11 show two cases of complex dynamic scenes. The first example was captured by a Cannon EOS 500D with ISO 6400. Its noise level is relatively low. The later example was obtained by a Nokia Lumia920. As we can see from both examples, lucky imaging and optical flow (+ median filtering) based results contain noticeable ghosting while the VBM3D results are over-smoothed. BM4D is better than the former two methods but still leaves a certain amount of chrominance noise patterns on the background. In our solution, we can automatically choose pixel colors consistent to the reference frame on the dynamic regions and collect more consistent pixels on the static regions (*e.g.*, cloth and door). As a result, our result strikes the best balance among removing noise, reconstructing fine details, and avoiding ghosting.

4.2 More Results

Handling motion blur. During the capture, some individual frames (*e.g.*, $10\% \sim 30\%$ of the total frames) may be blurry due to sudden camera shake or object motion. Figure 12 shows such an example captured with an iPhone 5S. We examined what would happen if a blurry frame were selected as the reference frame. To know this, we separately select frame 4 (sharp) and frame 5 (blurry) as the reference frame and generate two results. Figure 12 (b) and (d) show that our method is insensitive to the selection. This is because our method is capable of finding consistent pixels from the the majority of frames. A similar method was proposed in video deblurring [Cho et al. 2012], which found similar patches from sharp frames for deblurring.

Handling extreme low light. Figure 13 is a sequence captured by an iPhone 4S under an extreme low-light condition. Since the inputs are extremely dark, we preprocess the inputs by boosting the brightness (applying a shadow/highlight adjustment). While the noise after the boosting is very strong, our method still managed to produce a clean image with fine details (thin wires in the air and steel tower on the left).

Handling large occlusions. Figure 14 shows a sequence with a large, fast moving foreground (person). The consistent pixel map in the figure reveals how the mechanisms in our algorithm can reliably deal with (fast) large occlusion.

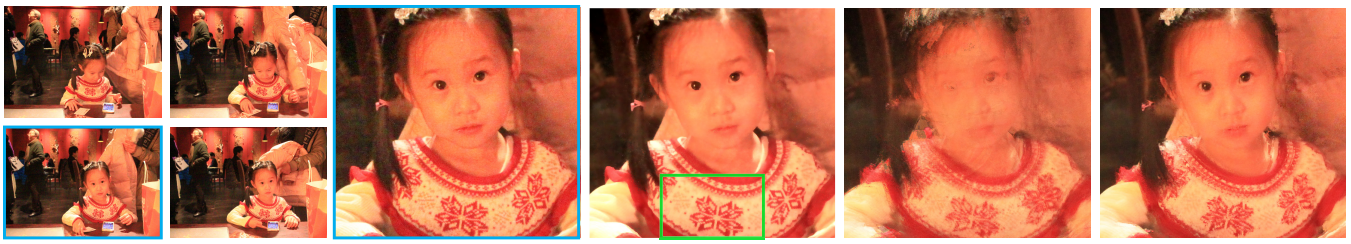

Figure 10: *Complex scene motion I.*

Figure 11: *Complex scene motion II.*

4.3 Time complexity.

We run our method on an Intel i5 3.2GHZ machine with 16G RAM. Our unoptimized C++ implementation (single core, no SSE SIMD acceleration) takes 920 ms on average to process 10 frames of 5M-pixel image. Specifically, our method takes 82ms, 177ms, 51ms, 30ms, 253ms, 328ms to build pyramid, extract and match sparse features, estimate homography flow, select consistent pixels, run temporal fusion, and execute multi-scale fusion. Our prototype on a smartphone (Nokia Lumia 920) costs about 4.7 seconds on average, without using multi-core or NEON instructions or GPU acceleration. As our solution is mainly based on point-wise operations, we expect it can be significantly accelerated. Table 1 further demonstrates the processing time of different methods on the same machine for Figure 8, 9, 10, and 11.

	Figure 8	Figure 9	Figure 10	Figure 11
Size	1520 × 2688	1600 × 1200	2352 × 1568	1280 × 720
(c)	337.85	139.88	304.75	81.54
(d)	74.06	40.27	64.94	28.46
(e)	577.39	298.73	538.02	126.63
(f)	214.83	123.42	198.36	53.92
(g)	1867.27	1019.31	1576.49	517.83
(h)	0.80	0.48	0.77	0.23

Table 1: *Processing times (sec.) of different denoising methods. (c) spatial-temporal filtering, (d) lucky imaging, (e) optical flow + median filtering, (f) VBM3D, (g) BM4D and (h) our method.*

5 Concluding Remarks

In an image burst, we expect camera motion from hand shake and small/moderate motion of the main subject(s). Our method is not designed for handling dramatic motion (e.g., in sports), or denoising a general video. When the motion between two frames cannot be well represented by our homography flow, such as scene transition or fast camera panning, or even non-rigid deformation (*e.g.*, water wave motion, flag waving), our approach will break.

Besides, there are two cases in which our consistent pixels selection may fail. The first case is when motion blur caused by dynamic objects appears on a majority of frames (more than half of all frames). On the dynamic regions, our pixel selection would automatically

(a) frame 4 (reference) (b) our result (use frame 4)

(c) frame 5 (reference) (d) our result (use frame 5) close-up views

Figure 12: *A sequence with motion blurs in individual frames. We respectively use the 4th frame (sharp) or the 5th frame (blurred) as the reference frame. Our solution produces similar results for both.*

(a) (b) (c)

Figure 13: *A sequence captured under an extreme low light condition. (a) input image sequence. (b) reference frame after brightness amplification. (c) our result.*

choose the reference-based strategy. If the reference frame contains blur, our result would retain the blur effect. But if a sharp frame is chosen as the reference, the issue can be avoided. Figure 15 (a) shows such an example. Therefore, we need a better strategy for selecting the reference frame. Besides, fast moving objects would be automatically removed by the median-based strategy for aggressive denoising. To avoid this issue, we may provide another option that allows users to choose the reference frame and constrain the reference region for pixels selection.

The second case is when different moving objects and the background may have similar colors in the same pixel locations. Our pixels selection algorithm relies on per-pixel color difference, which is too weak to distinguish such objects. Figure 15 (b) shows an example. Different moving persons have very similar color regions and such ambiguous regions (indicated by highlight box) occur in a majority of frames (*i.e.*, more than half of all frames). Finally, it will lead to ghosting because our selection wrongly regard it as the background. This issue also appears in ghost-free HDR reconstruction [Granados et al. 2013], which requires interactions to exclude these ambiguous regions.

Despite the above issues, we believe that our highly efficient solution is practical enough to be deployed for improving the photo experience of users in a broad range of lighting conditions.

Acknowledgements

We thank all the reviewers for their helpful discussions, Steve Lin and Jiangyu Liu for their help in proofreading.

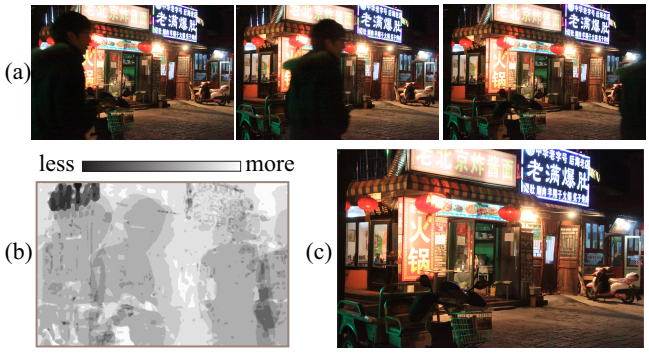

Figure 14: *A sequence with large occlusions (moving person). Top: inputs. Bottom: the left map shows the number of selected consistent pixels, and the right image is our result.*

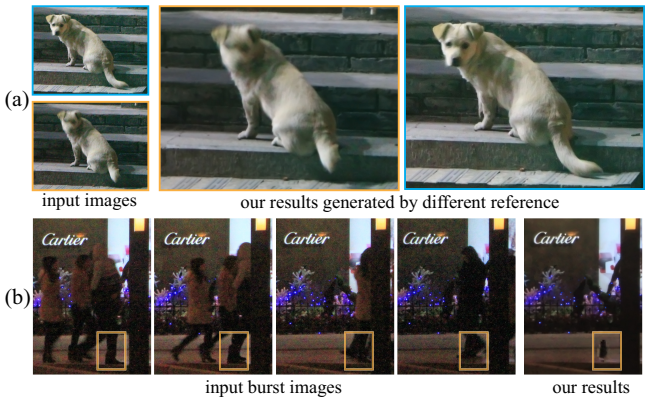

Figure 15: *Two failure cases. (a) motion blurs on dynamic objects (on a majority of frames). (b) ambiguous regions (with similar colors) from different moving objects.*

References

ADAMS, A., GELFAND, N., DOLSON, J., AND LEVOY, M. 2009. Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. Graph. (Proc. of SIGGRAPH) 28*, 3.

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. Patchmatch: A randomized correspondence algorithm for structural image editing. *SIGGRAPH 28*, 3.

BENNETT, E. P., AND MCMILLAN, L. 2005. Video enhancement using per-pixel virtual exposures. *ACM Trans. Graph. (Proc. of SIGGRAPH) 24*, 3, 845–852.

BRONSHTEIN, I. N., AND SEMENDYAYEV, K. A. 1997. *Handbook of Mathematics*. Springer-Verlag, New York, NY, USA.

BROX, T., BRUHN, A., PAPENBERG, N., AND WEICKERT, J. 2004. High accuracy optical flow estimation based on a theory for warping. In *Proc. ECCV*.

BUADES, A., COLL, B., AND MOREL, J.-M. 2005. A non-local algorithm for image denoising. In *Proc. CVPR*.

BUADES, A., LOU, Y., MOREL, J.-M., AND TANG, Z. 2009. A note on multi-image denoising. In *In Proceedings of the International Workshop on Local and Non-Local Approximation (LNLA) in Image Processing*.

BUADES, A., LOU, Y., MOREL, J.-M., AND TANG, Z. 2010. Multi image noise estimation and denoising. In *HAL*.

CAI, J. F., JI, H., LIU, C., AND SHEN, Z. 2009. Blind motion deblurring using multiple images. *J. Comput. Physics 228*, 14, 5057–5071.

CALONDER, M., LEPETIT, V., STRECHA, C., AND FUA, P. 2010. Brief: binary robust independent elementary features. In *Proc. ECCV*.

CHATTERJEE, P., JOSHI, N., KANG, S. B., AND MATSUSHITA, Y. 2011. Noise suppression in low-light images through joint denoising and demosaicing. In *Proc. CVPR*.

CHEN, J., AND TANG, C.-K. 2007. Spatio-temporal markov random field for video denoising. In *Proc. CVPR*.

CHEN, J., TANG, C.-K., AND WANG, J. 2009. Noise brush: Interactive high quality image-noise separation. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA) 28*, 5.

CHEN, X., KANG, S. B., YANG, J., AND YU, J. 2013. Fast patch-based denoising using approximated patch geodesic paths. In *Proc. CVPR*.

CHO, S., WANG, J., AND LEE, S. 2012. Vdeo deblurring for hand-held cameras using patch-based synthesis. *Proc. ACM SIGGRAPH 31*, 4, 64:1–64:9.

DABOV, K., FOI, A., AND EGIAZARIAN, K. 2007. Video denoising by sparse 3d transform-domain collaborative filtering. In *Proc. European Signal Process. Conf., EUSIPCO*.

DABOV, K., FOI, A., EGIAZARIAN, K., AND EGIAZARIAN, K. 2007. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans. on Image Processing 16*, 8, 2080–2095.

FARSIU, S., ROBINSON, M. D., ELAD, M., AND MILANFAR, P. 2004. Fast and robust multiframe super resolution. *IEEE Trans. on Image Processing 13*, 10, 1327–1344.

GALLO, O., GELFAND, N., CHEN, W., TICO, M., AND PULLI, K. 2009. Artifact-free high dynamic range imaging.

GONZALEZ, R. C., AND WOODS, R. E. 2007. *Digital Image Processing*. Prentice Hall, 3rd edition.

GRANADOS, M., KIM, K. I., TOMPKIN, J., AND THEOBALT, C. 2013. Automatic noise modeling for ghost-free hdr reconstruction. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA) 32*, 6, 1–10.

GRUNDMANN, M., KWATRA, V., CASTRO, D., AND ESSA, I. 2012. Calibration-free rolling shutter removal. In *Proc. ICCP*.

HARRIS, C., AND STEPHENS, M. 1988. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*.

HARTLEY, R., AND ZISSERMAN, A. 2003. *Multiple View Geometry in Computer Vision*, 2 ed. Cambridge University Press, New York, NY, USA.

JACOBS, D. E., BAEK, J., AND LEVOY, M. 2012. Focal stack compositing for depth of field control. In *Stanford Computer Graphics Laboratory Technical Report*.

JOSHI, N., AND COHEN, M. F. 2010. Seeing mt. rainier: lucky imaging for multi-image denoising, sharpening, and haze removal. In *Proc. ICCP*.

KALANTARI, N. K., SHECHTMAN, E., BARNES, C., DARABI, S., GOLDMAN, D. B., AND SEN, P. 2013. Patch-based high dynamic range video. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA) 32*, 6, 202:1–202:8.

LEVIN, A., AND NADLER, B. 2011. Natural image denoising: Optimality and inherent bounds. In *Proc. CVPR*, 2833–2840.

LIU, C., AND FREEMAN, W. T. 2010. A high-quality video denoising algorithm based on reliable motion estimation. Proc. ECCV, 706–719.

LIU, C., SZELISKI, R., KANG, S. B., ZITNICK, C. L., AND FREEMAN, W. T. 2008. Automatic estimation and removal of noise from a single image.

LIU, S., YUAN, L., TAN, P., AND SUN, J. 2013. Bundled camera paths for video stabilization. *ACM Trans. Graph. (Proc. of SIGGRAPH) 32*, 4, 78:1–78:10.

LIU, C. 2009. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology.

MAGGIONI, M., KATKOVNIK, V., EGIAZARIAN, K., AND FOI, A. 2013. A nonlocal transform-domain filter for volumetric data denoising and reconstruction. *IEEE Trans. on Image Processing*, 1, 119–133.

MARTIN, D., FOWLKES, C., TAL, D., AND MALIK, J. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*.

PARIS, S., AND DURAND, F. 2009. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision 81*, 24–52.

PORTILLA, J., STRELA, V., WAINWRIGHT, M. J., AND SIMONCELLI, E. P. 2003. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans. on Image Processing 12*, 11, 1338–1351.

REINHARD, E., WARD, G., PATTANAIK, S. N., DEBEVEC, P. E., AND HEIDRICH, W. 2010. *High Dynamic Range Imaging - Acquisition, Display, and Image-Based Lighting (2. ed.)*. Academic Press.

ROTH, S., AND BLACK, M. J. 2005. Fields of experts: a framework for learning image priors. In *Proc. CVPR*.

SEN, P., KALANTARI, N. K., YAESOUBI, M., DARABI, S., GOLDMAN, D. B., AND SHECHTMAN, E. 2012. Robust patch-based hdr reconstruction of dynamic scenes. *ACM Trans. Graph. (Proc. of SIGGRAPH) 31*, 6, 203:1–203:11.

TICO, M. 2008. Multiframe image denoising and stabilization. In *EUSIPCO*.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proc. ICCV*, 839–846.

VIOLA, P., AND JONES, M. 2001. Robust real-time object detection. In *International Journal of Computer Vision*.

ZHANG, M., AND GUNTURK, B. K. 2008. Multiresolution bilateral filtering for image denoising. *IEEE Trans. on Image Processing 17*, 12, 2324–2333.

ZHANG, L., AND WU, X. 2005. Color demosaicking via directional linear minimum mean square-error estimation. *TIP 14*, 12, 2167–2178.

ZHANG, L., VADDADI, S., JIN, H., AND NAYAR, S. K. 2009. Multiple view image denoising. In *Proc. CVPR*, 1542–1549.

ZONTAK, M., MOSSERI, I., AND IRANI, M. 2013. Separating signal from noise using patch recurrence across scales. In *Proc. CVPR*.