FlexISP: A Flexible Camera Image Processing Framework

Felix Heide[1,2] Markus Steinberger[1,3] Yun-Ta Tsai[1] Mushfiqur Rouf[1,2] Dawid Pająk[1] Dikpal Reddy[1]
Orazio Gallo[1] Jing Liu[1,4] Wolfgang Heidrich[2,5] Karen Egiazarian[1,6] Jan Kautz[1] Kari Pulli[1]
[1]NVIDIA [2]UBC [3]TU Graz [4]UC Santa Cruz [5] KAUST [6] TUT

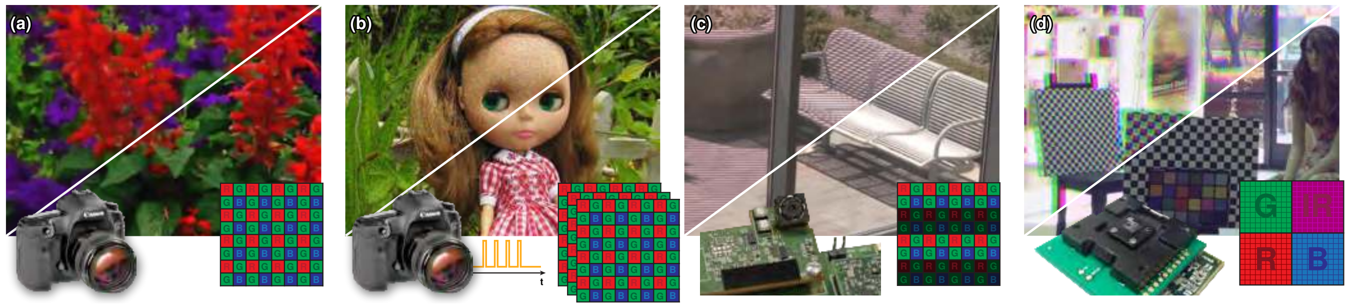


Figure 1: *Our end-to-end system reconstructs images and jointly accounts for demosaicking, denoising, deconvolution, and missing data reconstruction. Thanks to the separation of image model and formulation based on natural-image priors, we support both conventional and unconventional sensor designs. Examples (our results at lower right):* ***(a)*** *Demosaicking+denoising of a Bayer-sensor image (top: regular pipeline).* ***(b)*** *Demosaicking+denoising of a burst image stack (top: first frame shown).* ***(c)*** *Demosaicking+denoising+HDR from an interlaced exposure image (top: normalized exposure image).* ***(d)*** *Denoising+reconstruction of a color camera array image (top: naïve reconstruction).*

Abstract

Conventional pipelines for capturing, displaying, and storing images are usually defined as a series of cascaded modules, each responsible for addressing a particular problem. While this divide-and-conquer approach offers many benefits, it also introduces a cumulative error, as each step in the pipeline only considers the output of the previous step, not the original sensor data. We propose an end-to-end system that is aware of the camera and image model, enforces natural-image priors, while jointly accounting for common image processing steps like demosaicking, denoising, deconvolution, and so forth, all directly in a given output representation (e.g., YUV, DCT). Our system is flexible and we demonstrate it on regular Bayer images as well as images from custom sensors. In all cases, we achieve large improvements in image quality and signal reconstruction compared to state-of-the-art techniques. Finally, we show that our approach is capable of very efficiently handling high-resolution images, making even mobile implementations feasible.

CR Categories: I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Miscellaneous

Keywords: image processing, image reconstruction

Links: DL PDF WEB

1 Introduction

Modern camera systems rely heavily on computation to produce high-quality digital images. Even relatively simple camera designs reconstruct a photograph using a complicated process consisting of tasks such as dead-pixel elimination, noise removal, spatial upsampling of subsampled color information (e.g., demosaicking of Bayer color filter arrays), sharpening, and image compression. More specialized camera architectures may require additional processing, such as multi-exposure fusion for high-dynamic-range imaging or parallax compensation in camera arrays.

The complexity of this process is traditionally tackled by splitting the image processing into several independent pipeline stages [Ramanath et al. 2005]. Splitting image reconstruction into smaller, seemingly independent tasks has the potential benefit of making the whole process more manageable, but this approach also has severe shortcomings. First, most of the individual stages are mathematically ill-posed and rely heavily on heuristics and prior information to produce good results. The following stages then treat the results of these heuristics as ground truth input, aggregating the mistakes through the pipeline. Secondly, the individual stages of the pipeline are in fact not truly independent, and there often exists no natural order in which the stages should be processed. For example, if noise removal follows demosaicking in the pipeline, the demosaicking step must be able to deal with noisy input data when performing edge detection and other such tasks required for upsampling the color channels, and denoising is complicated as the noise statistics change due to the interpolation in demosaicking.

We present a framework that replaces the traditional pipeline with a single, integrated inverse problem (Fig. 2). We solve this inverse problem with modern optimization methods while preserving the modularity of the image formation process stages. Instead of applying different heuristics in each stage of the traditional pipeline, our system provides a single point to inject image priors and regularizers in a principled and theoretically well-founded fashion.

Despite the integration of the individual tasks into a single optimization problem, our system is flexible and we can easily extend it to include new image formation models and camera types, by simply

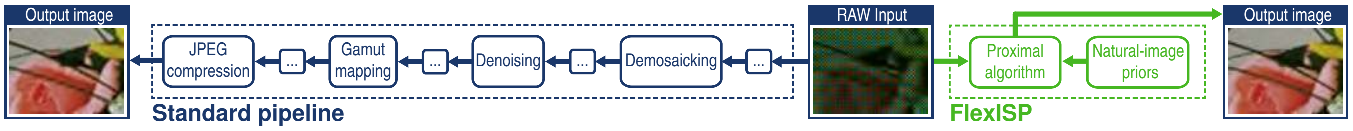


Figure 2: *The traditional camera processing pipeline consists of many cascaded modules, introducing cumulative errors. We propose to replace this pipeline with a unified and flexible camera processing system, leveraging natural-image priors and modern optimization techniques.*

providing a procedural implementation of the forward image formation model. This image formation model is typically composed of a sequence of independent linear transformations (e.g., lens blur followed by spatial sampling of the color information, followed by additive noise). To illustrate the flexibility of our system, we applied it to a number of image formation models (see Fig. 1), including joint Bayer demosaicking and denoising, deconvolution of camera shake and out-of-focus blur, interlaced HDR reconstruction and motion blur reduction, image fusion from color camera arrays, joint image stack denoising and demosaicking, and optimization all the way to the output representation.

All these image formation models are combined with a set of state-of-the-art image priors. The priors can be implemented independent of each other or the image formation system that they are applied to. This enables sharing and re-using high-performance implementations of both the image formation process stages and the image priors for different applications. We analyzed a variety of image priors and determined a single set that we use for all applications, and which consistently outperforms the best specialized algorithms for well-researched problems such as demosaicking, denoising, and deconvolution, sometimes by a significant margin. Since the optimization of the forward model and the image priors is highly parallelizable, we also provide a very efficient GPU implementation. This allows us to process high-resolution images even on mobile devices (such as modern tablets with integrated mobile GPUs). We call this flexible image signal processing system FlexISP.

Contributions Our main contributions include: a flexible end-to-end camera image processing system that can handle different applications, sensor types, and priors, with minimal code changes; an in-depth analysis of the design choices made to create our system—in particular of the priors used in our optimization; and an evaluation of our framework against many state-of-the-art methods, demonstrating the high image quality we achieve.

2 Related Work

Camera image processing pipelines Most modern digital cameras implement the early image processing as a pipeline of simple stages [Ramanath et al. 2005]. These image signal processors (ISPs) usually take in raw Bayer sensor measurements, interpolate over stuck pixels, demosaic the sparse color samples to a dense image with RGB in every pixel [Zhang et al. 2011], attempt to denoise the noisy signal [Shao et al. 2013], enhance edges, tonemap the image to 8 bits per channel, and optionally compress the image. The camera capture parameters are controlled by the auto exposure, focus, and white balancing algorithms [Adams et al. 2010].

Demosaicking Many methods exist for demosaicking Bayer images [Zhang et al. 2011], and some extend to videos (or stacks of images) [Wu and Zhang 2006; Bennett et al. 2006]. Unlike our proposed technique, these methods only focus on demosaicking and do not jointly handle denoising, for example.

Denoising Self-similarity and sparsity are two key concepts in modern image denoising. Non-local image modeling utilizes structural similarity between image patches. Non-local means (NLM) [Buades et al. 2005] filters a single image using a weighted average of similar patches for each pixel.

Many orthogonal transforms, such as DFT, DCT, and wavelets, have good decorrelation and energy compaction properties for natural images. This property has been utilized in local transform-based denoising schemes, such as wavelet shrinkage [Coifman and Donoho 1995] and sliding window DCT filtering [Egiazarian et al. 1999].

BM3D [Dabov et al. 2007a] was the first denoising algorithm to simultaneously exploit sparse transform-domain representations and non-local image modeling. We use BM3D as a self-similarity-inducing denoising prior. Combining internal (such as BM3D) and external (such as Total Variation – TV) denoising has been recently shown to be advantageous; Mosseri et al. [2013] run two methods separately and then merge the results. We combine internal and external information in a single optimization framework. We also modify standard BM3D to perform better as a natural image prior.

Using multiple images Capturing several images allows one to overcome some inherent camera limitations. High-dynamic-range (HDR) imaging [Reinhard et al. 2010] commonly captures an image stack with different exposure times, and reconstructs more detail in the very dark and bright areas than would otherwise be possible. Multiple frames can be combined to deblur images [Tico and Pulli 2009] or to superresolve more detail than is available from a single image [Irani and Peleg 1991]. We demonstrate an application where a low-light image burst is first registered and then jointly demosaicked and denoised to create a better output image.

Modified camera designs It is also possible to redesign the camera to extend its imaging capabilities. Camera arrays can capture a light field [Wilburn et al. 2005], which can be post-processed for changing the view point or focus, creating a large virtual aperture, extracting depth from the scene, etc. An extremely thin design without a main lens is possible by integrating several small lenses over a sensor, as in the PiCam [Venkataraman et al. 2013] design. Cameras have also been modified to extend the dynamic range that they capture. For example, the density of the color filter array can be varied per pixel [Nayar and Branzoi 2003], but this comes at the cost of reduced light efficiency. Alternatively, the sensor design itself can be modified to allow for per-row selection of the exposure time or sensor gain [Gu et al. 2010; Hajisharif et al. 2014]. Our system can be easily adapted to handle such new camera designs. For instance, in one of our applications, we use a camera sensor similar to the one by Gu et al., but instead reconstruct the HDR image with our joint optimization system, which significantly improves the results.

Joint optimization approaches Addressing subproblems separately does not yield the best-quality reconstructions, especially for complex image formation models. Recently, a number of researchers have identified this problem, and proposed joint solutions to several subproblems (Table 1). Examples include joint demosaicking and denoising [Chatterjee et al. 2011; Jeon and Dubois 2013], demosaicking and deblurring [Schuler et al. 2011], demosaicking and HDR reconstruction [Narasimhan and Nayar 2005; Ajdin et al. 2008], demosaicking and superresolving an image stack [Farsiu et al. 2006; Bennett et al. 2006], and image fusion with superresolution for color camera arrays [Venkataraman et al. 2013]. These

Paper	Method	Application	Priors
[Chatterjee et al. 2011]	Unkown	DM + DN	p-norm ($p = 0.65$)
[Schuler et al. 2011]	Alt. Min.	DM + DC	TV + color
[Schuler et al. 2013]	ML	DC (restoration)	(MLP)
[Narasimhan and Nayar 2005]	Lin. Regr.	DM + HDR	—
[Ajdin et al. 2008]	Stoch. Opt.	DM for HDR	Color
[Farsiu et al. 2006]	Grad. Desc.	DM + SR	Bilat. TV + (cross-)color
[Bennett et al. 2006]	Bayes	DM + SR	2-color
[Venkataraman et al. 2013]	Bayes	Fusion + SR	Bilat. TV
[Afonso et al. 2010]	ADMM	DC | inpainting	TV | wavelets
[Chambolle and Pock 2011]	PD	DN | DC | flow | ...	TV | Huber-ROF | curvelet
[Heide et al. 2013]	PD	DC	TV + cross-channel
[Venkatakrishnan et al. 2013]	ADMM	Tomographic DN	KSVD | BM3D | TV | ...
Ours	PD	Flexible	BM3D + TV + color

Table 1: *Table comparing previous work and ours.* ML = *machine learning,* PD = *primal dual,* DM = *demosaic,* DN = *denoise,* DC = *deconvolution,* SR = *superresolution.*

techniques all address a specific subset of the image processing pipeline for a single camera design. We extend these ideas into a single, flexible image optimization framework called FlexISP, that can be applied to many different applications and camera designs. Our approach uses proximal operators and the primal-dual method for optimization (see [Parikh and Boyd 2013] for an accessible introduction). Primal-dual methods, and the related Alternating Direction Method of Multipliers (ADMM), have been used in many imaging applications, such as deconvolution, inpainting [Afonso et al. 2010], superresolution, optical flow [Chambolle and Pock 2011], denoising [Venkatakrishnan et al. 2013], and deconvolution for simple lenses [Heide et al. 2013] using various priors (Table 1). In contrast, we formulate a framework that allows easy integration of various natural image priors, and demonstrate that a specific set (BM3D + TV + cross-channel) is sufficient for all our applications while outperforming all state-of-the-art methods. We discuss the application of the priors, their effects, and the way they should be combined; we also discuss some priors that we discarded and explain why they are not needed, given the priors we adopted. Our approach improves on the current state-of-the-art methods in demosaicking (Figs. 8 and 10), deconvolution (Table 4), interlaced HDR reconstruction (Fig. 11), burst denoising (Fig. 13), and JPEG deblocking (Fig. 14).

3 Optimization for Image Reconstruction

In this section we present our image reconstruction framework. We start by formulating the problem as a linear least-squares problem with non-linear and possibly non-convex regularizers. Once cast in this framework, we show how standard, non-linear optimization algorithms can be applied to the solution of this inverse problem using proximal operators for the data term and the individual regularizers.

3.1 Inverse Problem

We represent the unknown latent image $\mathbf{x} \in \mathbb{R}^N$ and the observed image $\mathbf{z} \in \mathbb{R}^n$ as vectors. Depending on the camera and its lens, sensor, and so forth, the latent image undergoes various transformations. In the example of the Bayer pattern sensor, the observed image $\mathbf{z}$ is an n-vector with only one color sample per pixel, while the latent image $\mathbf{x}$ is a $N = 3n$-vector, where the first n terms are the red channel values, the next n are the greens, and the rest the blues. This subsampling can be modeled as a projection operator, expressed as an $n \times N$ matrix $\mathbf{D}$. As another example, consider that the latent image usually undergoes blurring due to the camera's anti-aliasing filter and scattering in the optics. Due to the linear nature of these optical processes, the image transformation can be expressed as a matrix $\mathbf{B}$ operating on the image vector. We subsume these transformations in a matrix $\mathbf{A}$, and express our observation model as

$$\mathbf{z} = \mathbf{A}\mathbf{x} + \eta, \tag{1}$$

where η is additive Gaussian noise. The definition of matrix $\mathbf{A}$ changes based on the exact application (see Sec. 4), but does not vary based on the image content.

Our goal is to find the underlying latent image $\mathbf{x}$ from the (usually) sparse and noisy observations $\mathbf{z}$. Given the Gaussian noise model, this can be achieved with a standard ℓ_2 minimization. Unfortunately, the data term by itself is typically not sufficient due to information loss inherent in transformations such as blurring and subsampling. The inverse problem must therefore be regularized with a non-linear, and possibly non-convex term $\Gamma(\mathbf{x})$, which narrows down the solution space. $\Gamma(\cdot)$ is itself a weighted sum of individual natural image priors (Sec. 3.3.2). Solving for $\mathbf{x}$ in Eq. 1 then amounts to solving the following minimization problem:

$$\min_{\mathbf{x}} \underbrace{\frac{1}{2}\|\mathbf{z} - \mathbf{A}\mathbf{x}\|_2^2}_{\text{data fidelity}} + \underbrace{\Gamma(\mathbf{x})}_{\text{regularization}}. \tag{2}$$

For mapping this problem onto known non-linear solvers, it is useful to rewrite the data term and image formation model as $G(\mathbf{x}) = \frac{1}{2}\|\mathbf{z} - \mathbf{A}\mathbf{x}\|_2^2$ and the regularization term as $\Gamma(\mathbf{x}) = F(\mathbf{K}\mathbf{x})$, where F is a non-linear function, and $\mathbf{K}$ is a matrix. For example, for the well-known TV regularizer $\Gamma(\cdot) = \|\cdot\|_{TV}$, we define $F(\cdot) = \|\cdot\|_1$ as the ℓ_1 norm, and $\mathbf{K} = \nabla$ as a discrete gradient operator. Using this notation, we can rewrite Eq. 2 as a canonical constrained optimization problem using a new slack variable $\mathbf{y}$:

$$\min_{\mathbf{x},\mathbf{y}} G(\mathbf{x}) + F(\mathbf{y}), \quad \text{subject to } \mathbf{K}\mathbf{x} = \mathbf{y}. \tag{3}$$

We note that finding appropriate priors is crucial for yielding high-quality image reconstructions. We provide an analysis of different priors in Sec. 5.

3.2 Solvers

Equation 3 is a standard problem in numerical optimization, and a large array of existing and well-documented non-linear solvers provide convergence guarantees when F and G are both convex. Many such solvers can be expressed in terms of the *proximal operators* for G and F [Parikh and Boyd 2013]:

$$\operatorname{prox}_{\tau H}(\mathbf{v}) = \arg\min_{\mathbf{u}} \left(H(\mathbf{u}) + \frac{1}{2\tau}\|\mathbf{u} - \mathbf{v}\|_2^2 \right), \tag{4}$$

where H stands for either G or F. These proximal operators can be interpreted as minimizing G and F, while staying close to the starting point $\mathbf{v}$. After deriving the proximal operators for our data term (Sec. 3.3.1) and regularizers (Sec. 3.3.2), we can use a wide range of non-linear solvers. We have experimented both with ADMM [Parikh and Boyd 2013] and the primal-dual algorithm (Algorithm 1 of Chambolle and Pock [2011], reproduced below for reference). We found that both algorithms produce very similar results, with the latter being about 20% faster in our applications, which is why we settled on the primal-dual algorithm for all examples in this paper.

```
Algorithm 1 – First order primal-dual
Initialization: $\gamma\tau\|\mathbf{K}\|^2 < 1$, $\theta \in [0,1]$, $(\mathbf{x}^0, \mathbf{y}^0)$, $\bar{\mathbf{x}}^0 = \mathbf{x}^0$.
Repeat until convergence:
    penalty:          $\mathbf{y}^{k+1} = \operatorname{prox}_{\gamma F^*}\left(\mathbf{y}^k + \gamma\mathbf{K}\bar{\mathbf{x}}^k\right)$
    data fidelity:    $\mathbf{x}^{k+1} = \operatorname{prox}_{\tau G}\left(\mathbf{x}^k - \tau\mathbf{K}^T\mathbf{y}^{k+1}\right)$
    extrapolation:    $\bar{\mathbf{x}}^{k+1} = \mathbf{x}^{k+1} + \theta\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right)$
```

We initialize $\mathbf{x}^0$ with the observations—missing values are copied or averaged from observed nearby values; $\mathbf{y}^0$ can be set to zero.

The primal-dual algorithm actually requires the proximal operator for the *convex conjugate* F^* of F [Chambolle and Pock 2011]. This operator can be expressed in terms of the original operator $\text{prox}_{\gamma F}$ using the Moreau decomposition [Parikh and Boyd 2013]

$$\text{prox}_{\gamma F^*}(\mathbf{v}) = \mathbf{v} - \gamma \cdot \text{prox}_{\frac{1}{\gamma}F}(\frac{\mathbf{v}}{\gamma}). \tag{5}$$

We can therefore focus on deriving the primal proximal operators for all regularization terms.

3.3 Proximal Operators

Applying the solver to our problem requires deriving the proximal operators for a given image formation model and regularizers.

3.3.1 Data Fidelity Operator

The data fidelity (primal) step in Alg. 1 updates the current estimate $\mathbf{x}$ of the latent image. Loosely speaking, this operator takes a gradient descent step of the data term while remaining close to the argument of the operator. From Eqs. 2 and 4 we see that

$$\mathbf{x} = \text{prox}_{\tau G}(\mathbf{v}) = \arg\min_{\mathbf{u}} \left(\frac{1}{2}\|\mathbf{z} - \mathbf{A}\mathbf{u}\|_2^2 + \frac{1}{2\tau}\|\mathbf{u} - \mathbf{v}\|_2^2 \right). \tag{6}$$

This is a simple least-squares problem, whose solution $\mathbf{x}$ is obtained as the solution to the following linear system:

$$\left(\tau \mathbf{A}^T\mathbf{A} + \mathbb{I}\right)\mathbf{x} = \left(\tau \mathbf{A}^T\mathbf{z} + \mathbf{v}\right). \tag{7}$$

There are many ways to solve this linear system, but as the matrix is symmetric and positive semi-definite, we can use the Conjugate Gradient (CG) algorithm. The advantage of CG in our setting is that the image formation model $\mathbf{A}$ and its transpose can be represented algorithmically, without explicitly generating the system matrix.

3.3.2 Regularization Operators

The regularization (penalty) step in Alg. 1 updates our current estimate of the slack variable $\mathbf{y}$, making a gradient step on the regularization terms. We now describe the set of regularization terms that work well in our framework (cf. Sec. 5).

Image gradient sparsity prior The first regularization term is the standard (isotropic) Total Variation regularizer, which minimizes the ℓ_1 norm of the gradient magnitudes [Rudin et al. 1992]. We express the term in our framework by defining $F_0(\cdot) = \|\cdot\|_1$, for which the proximal operator is the component-wise soft shrinkage

$$\text{prox}_{\gamma_0 F_0}(\mathbf{v}_0) = \begin{cases} \mathbf{v}_0 - \gamma_0 & \text{if } \mathbf{v}_0 > \gamma_0 \\ \mathbf{v}_0 + \gamma_0 & \text{if } \mathbf{v}_0 < -\gamma_0 \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

$\mathbf{K}_0 = \nabla$, which means that $\mathbf{K}_0\mathbf{x}$ produces *x*- and *y*-gradient images.

Denoising prior We show that *any* Gaussian denoiser can be expressed as a proximal operator and be used in our framework. While the derivation is general, we are particularly interested in using powerful self-similarity-imposing denoisers. For specific denoisers with simple structure (such as TV), the properties have been well explored [Oymak and Hassibi 2013]. However, we extend this to complex cases that do not have a closed-form solution. We use a prior probability distribution $g(\mathbf{x})$ to express that we expect the latent image $\mathbf{x} \in \mathbb{R}^N$ to be locally self-similar. Furthermore, we assume that our observation $\mathbf{z} \in \mathbb{R}^N$ may have been polluted by per-pixel Gaussian noise with standard deviation $\sqrt{\gamma_1}$, yielding the likelihood $f(\mathbf{z}|\mathbf{x})$ of noisy observations $\mathbf{z}$ given the latent image $\mathbf{x}$:

$$f(\mathbf{z}|\mathbf{x}) = \frac{1}{\sqrt{(2\pi\gamma_1)^N}} \exp\left(-\|\mathbf{z} - \mathbf{x}\|_2^2 \big/ 2\gamma_1\right). \tag{9}$$

This enables us to treat $\mathbf{x}$ as a random variable and, assuming the prior distribution $g(\cdot)$, characterize its uncertainty using Bayesian statistics. Applying Bayes' rule, the posterior distribution of the latent image $\mathbf{x}$ given the noisy measurement $\mathbf{z}$ is $f(\mathbf{x}|\mathbf{z}) \propto g(\mathbf{x})\, f(\mathbf{z}|\mathbf{x})$. The maximum-a-posteriori estimate $\mathbf{x}_{MAP}$ is the mode

$$\mathbf{x}_{MAP} = \arg\max_{\mathbf{x}} g(\mathbf{x})\, f(\mathbf{z}|\mathbf{x}). \tag{10}$$

Note that this is only the MAP estimate for the denoising prior, not for the inverse problem in Eq. 2.

By estimating the mode, we find a tradeoff between the likelihood (i.e., trust in our observations) and the prior knowledge. We maximize the negative log of the posterior to get

$$\begin{aligned} \mathbf{x}_{MAP} &= \arg\max_{\mathbf{x}} g(\mathbf{x})\, f(\mathbf{z}|\mathbf{x}) = \arg\max_{\mathbf{x}} \log\left(g(\mathbf{x})\, f(\mathbf{z}|\mathbf{x})\right) \\ &= \arg\max_{\mathbf{x}} \log\left(g(\mathbf{x})\right) + \log\left(f(\mathbf{z}|\mathbf{x})\right) \\ &= \arg\min_{\mathbf{x}} -\log\left(g(\mathbf{x})\right) + \tfrac{1}{2\gamma_1}\|\mathbf{z} - \mathbf{x}\|_2^2. \end{aligned} \tag{11}$$

Comparing Eq. 11 with the definition of the proximal operator in Eq. 4, we see that setting $F_1 \equiv -\log(g(\cdot))$ yields

$$\text{prox}_{\gamma_1 F_1}(\mathbf{v}_1) = \mathbf{x}_{MAP} \tag{12}$$

for $\mathbf{x} = \mathbf{u}$ and $\mathbf{z} = \mathbf{v}_1$. This means that any Gaussian denoising algorithm that can be formulated as a MAP estimation, can be interpreted as a proximal operator, is consistent with our framework, and can be integrated if desired.

To apply the proximal operator, we set $\mathbf{K}_1$ to identity $\mathbb{I}$ and then simply run the chosen denoiser on the image $\mathbf{v}_1 = \mathbf{y}_1^k + \gamma_1 \mathbf{K}_1 \bar{\mathbf{x}}^k$.

We have experimented with various denoisers: Sliding DCT which uses collaborative filtering, NLM which uses self-similarity, and patch-based NLM and BM3D which use both. The choice depends on the application and on the desired reconstruction speed and quality, which we will discuss in detail in Sec. 5. Out of those properties, self-similarity, which helps to inpaint in case of missing data, appears to be the most important.

Cross-channel gradient correlation To ensure edge consistency between color channels and avoid color fringing we include the cross-channel prior [Heide et al. 2013]. We enforce that chroma-gradients of channels l, k are similar and sparse by equating $\nabla\mathbf{z}_l/\mathbf{z}_l \approx \nabla\mathbf{z}_k/\mathbf{z}_k \Leftrightarrow \nabla\mathbf{z}_l \cdot \mathbf{z}_k \approx \nabla\mathbf{z}_k \cdot \mathbf{z}_l$ with an ℓ_1-penalty term. The corresponding $\mathbf{K}_2$ computes the difference between scaled gradients of the channels (see the details in the code in the supplemental material). The penalty F_2 is the ℓ_1 norm, and the proximal operator is again a component-wise soft shrinkage (Eq. 8).

Combining the operators We combine the individual regularizers and image priors as a weighted sum

$$F(\mathbf{K}\mathbf{x}) = \sum_{i=0}^{k} \phi_i F_i(\mathbf{K}_i\mathbf{x}). \tag{13}$$

The combined $\mathbf{K}$ is obtained by stacking the component matrices; with our three regularizers $\mathbf{K} = [\mathbf{K}_0; \mathbf{K}_1; \mathbf{K}_2] \equiv [\mathbf{K}_0^T\mathbf{K}_1^T\mathbf{K}_2^T]^T$, where the semicolon notation stacks the components vertically. Note that we actually never form and multiply the full $\mathbf{K}$ matrix, instead, for $\mathbf{K}_0$ we just compute *x*- and *y*-gradient images, for $\mathbf{K}_1$ we simply use the image, and for the components of $\mathbf{K}_2$ we multiply an image with a gradient image and take the difference of two such images.

The proximal operator of F becomes the vector

$$\text{prox}_{\gamma F}(\mathbf{v}) = \left[\text{prox}_{\gamma_0 F_0}(\mathbf{v}_0)\,;\,\ldots\,;\,\text{prox}_{\gamma_k F_k}(\mathbf{v}_k)\right], \tag{14}$$

where $\gamma_i = \gamma\phi_i$, and the component $\mathbf{v}_i$ is obtained as shown in Alg. 1 (penalty step), which together with Eq. 5 produces the next iteration $\mathbf{y}^{k+1}$ of the slack variable.

4 Applications

Next we describe our applications and their objective functions.

Matrix	Function
$\mathbf{D}$	Image down-sampling or sub-sampling
$\mathbf{M}$	Masking of pixels (e.g., saturated or broken pixels)
$\mathbf{B}$	Image blur
$\mathbf{S}$	Resampling matrix implementing an image warp
$\mathbf{C}$	Color matrix
$\mathcal{F}$	Fourier transform or DCT
$\mathbf{Q}$	JPEG quantization matrix

Table 2: *Matrix symbols used in the image formation models.*

4.1 Demosaicking

Our first application performs demosaicking of a Bayer raw image and simultaneous denoising, if required. The data model is quite simple: we set $\mathbf{A} = \mathbf{D}$, the Bayer array decimation matrix, and get

$$G(\mathbf{x}) = \|\mathbf{z} - \mathbf{D}\mathbf{x}\|_2^2 . \tag{15}$$

4.2 Deblurring

We next demonstrate non-blind deconvolution, i.e., assuming a known (pre-calibrated) PSF. Like Xu and Jia [2010] and Heide et al. [2013], we encode the blur in the matrix $\mathbf{A} = \mathbf{B}$ and get

$$G(\mathbf{x}) = \|\mathbf{z} - \mathbf{B}\mathbf{x}\|_2^2 . \tag{16}$$

As demonstrated by Schuler et al. [2011], one can also incorporate Bayer array sensing into deconvolution by setting $\mathbf{A} = \mathbf{DB}$.

4.3 Interlaced HDR

Our third application focuses on HDR imaging. Some recent sensors (e.g., Aptina AR1331CP and Sony IMX135) can record two exposures in a single frame by allowing the exposure time to be set independently for even and odd macro-rows (a pair of consecutive rows that cover all the color samples) of an otherwise normal Bayer sensor. Figure 1(c) depicts the exposure pattern; odd macro-rows integrate light for a longer time than the even ones. The advantage is that two exposures, together covering a wider dynamic range than a single shot, are captured essentially at the same time, while traditional stack-based HDR allows time to pass and objects to move between the shots. However, samples are more sparse: odd macro-rows may saturate in bright regions and even macro-rows may be too noisy to be useful in dark regions. We easily adapt our framework to reconstruct high-quality HDR images from such sensors.

We introduce a binary diagonal indicator matrix $\mathbf{M}$ for masking bad pixels, where 0 indicates a useless pixel (either saturated or too noisy), and 1 means that the pixel has useful data. The blur matrix $\mathbf{B}$ models the optics and sensor anti-aliasing filter by measuring the camera's PSF; all applications model blur in this fashion, unless otherwise noted. We measure the blur like Xu and Jia [2010]. Other calibration-target-based methods, such as Brauers et al. [2010], could be used as well. Finally, we set $\mathbf{A} = \mathbf{MDB}$ for the blurred, decimated, and saturated observation matrix, i.e.,

$$G(\mathbf{x}) = \|\mathbf{Mz} - \mathbf{MDBx}\|_2^2 . \tag{17}$$

Note that, when solving Eq. 7, we also need to replace $\mathbf{z}$ with $\mathbf{Mz}$.

4.4 Color Array Camera

Single-chip array cameras provide thin camera designs and a potential for depth-map-based applications [Venkataraman et al. 2013]. The color filters are per lens and not per pixel, and each lens can be optimized for the wavelength passed by the respective filter. Since each sub-image has only one color, no demosaicking is required. Instead, the images have to be registered, but this is a challenging task since each sub-image has a different color and a slightly different viewpoint, leading to parallax and occlusions.

We use our framework to fuse images from a 2×2 single-chip color array camera, shown in Fig. 1(d). This prototype uses R, G, and B sensors, with a 1.4mm baseline. Additionally, we use a second 2×2 camera array with a larger baseline (2.55cm) for larger scenes.

We first estimate the registration between the sub-images, which we encode in the matrix $\mathbf{A}$, and use the matrix $\mathbf{M}$ as a confidence mask modeling the uncertainty in this registration. This uncertainty occurs both in flow-based or depth-based registration. We have opted for a novel cross-channel optical flow for registration across different color channels, as it avoids any geometric calibration. Each channel is replaced with a normalized image $\tilde{\mathbf{z}} = \frac{\|\nabla \mathbf{z}\|}{\mathbf{z}+\varepsilon}$ (Fig. 3), and we solve for the cross-channel optical flow Δv between the color channels $\mathbf{z}_C$ (for $C = \{R, B\}$) and the fixed green channel $\mathbf{z}_G$ with

$$\min_{\Delta v} \left\| \tilde{\mathbf{z}}_C + (\nabla \tilde{\mathbf{z}}_C)^T \Delta v - \tilde{\mathbf{z}}_G \right\|_2^2 + \|\Delta v\|_1 \tag{18}$$

using a standard optical flow algorithm [Liu 2009].

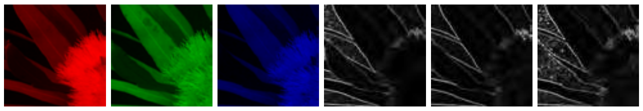


Figure 3: *R, G, and B color planes, and corresponding normalized images. The normalized gradient magnitude images look similar, apart from noise. Prominent edges show up on all channels, helping the optical flow algorithm to converge on a good motion estimate.*

We detect the reliable flow estimates through forward-backward consistency check and mark them in the indicator matrix $\mathbf{M}$.

Having computed the flows Δv_R and Δv_B, we form flow matrices $\mathbf{S}_i$ for each capture i. We also account for the optical blur in the 2×2 camera and recover some of the sharpness in the reconstructed image. The blur is encoded in the matrix $\mathbf{B}$ as before. The mosaicking matrix is not needed for this camera. We set $\mathbf{A}_i = \mathbf{M}_i\mathbf{B}\mathbf{S}_i$ for each capture i (out of k), and get

$$G(\mathbf{x}) = \|[\mathbf{M}_1\mathbf{z}_1; \ldots; \mathbf{M}_k\mathbf{z}_k] - [\mathbf{M}_1\mathbf{B}\mathbf{S}_1; \ldots; \mathbf{M}_k\mathbf{B}\mathbf{S}_k]\mathbf{x}\|_2^2 . \tag{19}$$

As before, $F(\cdot)$ contains the BM3D denoiser and the cross-channel image prior, which now helps to reconstruct missing colors in occluded regions by using information from the other channels.

4.5 Burst Denoising and Demosaicking

Many compact cameras produce very noisy results in low-light situations due to their small sensor size. Instead of a single photograph, one can take a rapid burst of multiple exposures and then combine them to reduce noise. This is not a new idea in itself [Tico 2008; Buades et al. 2009], and some cameras have a multi-frame denoising mode. However, previous techniques demosaic each image first and then denoise the stack of images. This is not ideal since demosaicking will modify and possibly even amplify noise. Furthermore, additional frames can aid the demosaicking process [Wu and Zhang 2006; Bennett et al. 2006]. We demonstrate that our framework can jointly demosaic and denoise a stack of burst images, yielding results that are superior to methods that pipeline these two processes.

The image stack is likely to be slightly misaligned due to small camera and object motion. We handle this by aligning all k observations $\mathbf{z}_i$ to the reference frame (e.g., the first one) by computing a brute-force $1/8^{\text{th}}$ sub-pixel-accurate nearest-neighbor search (ℓ_1-norm on 15×15 patches). We represent it as a warp matrix $\mathbf{S}_i$. Note that we also bake the resampling filter into this matrix. While our captures did not contain any motion blur, one could easily incorporate and calibrate the blur as described for interlaced HDR in Section 4.3. Finally, we set $\mathbf{A}_i = \mathbf{M}_i\mathbf{D}\mathbf{S}_i$ to get

$$\begin{aligned} G(\mathbf{x}) &= \sum_{i=1}^{k} \|\mathbf{M}_i\mathbf{z}_i - \mathbf{M}_i\mathbf{D}\mathbf{S}_i\mathbf{x}\|_2^2 \\ &= \|[\mathbf{M}_1\mathbf{z}_1;\ldots;\mathbf{M}_k\mathbf{z}_k] - [\mathbf{M}_1\mathbf{D}\mathbf{S}_1;\ldots;\mathbf{M}_k\mathbf{D}\mathbf{S}_k]\mathbf{x}\|_2^2 . \end{aligned} \tag{20}$$

The masking matrix $\mathbf{M}_i$ can be used to mask out pixels for which no good alignment is found; however, we did not find this necessary.

4.6 Beyond Linear RGB

We next demonstrate that we do not need to stop at linear RGB, or even at non-linear YUV with subsampled chroma, we can extend our processing pipeline all the way to the JPEG-compressed image. JPEG compression works by taking an RGB image $\mathbf{x}$, converting it to YUV space with a linear color transform $\mathbf{C}$, downsampling the chroma components ($\mathbf{D}_J$), performing a block-based DCT on the individual channels ($\mathcal{F}$), and reweighting the individual frequencies according to a quantization matrix $\mathbf{Q}$ [Wallace 1991]. Finally, the integer JPEG coefficients are obtained through rounding:

$$[\mathbf{c}] = \text{round}(\underbrace{\mathbf{Q}\mathcal{F}\mathbf{D}_J\mathbf{C}}_{\mathbf{J}}\ \mathbf{x}). \tag{21}$$

We can see that all steps except for the final quantization can be expressed as a linear operator. Likewise, JPEG decompression can be expressed as a linear operator $\mathbf{J}^{-1} = \mathbf{C}^{-1}\mathbf{U}_J\mathcal{F}^{-1}\mathbf{Q}^{-1}$, where the upsampling operator $\mathbf{U}_J = \mathbf{D}_J^{\dagger}$ is a pseudo-inverse of the chroma subsampling.

With this observation it becomes possible to directly perform image reconstruction in the JPEG transform space, by setting $\mathbf{A} = \mathbf{M}\mathbf{D}\mathbf{B}\mathbf{J}^{-1}$, where $\mathbf{D}$ denotes the Bayer decimation, as below:

$$\min_{\mathbf{c}} \left\|\mathbf{M}\mathbf{z} - \mathbf{M}\mathbf{D}\mathbf{B}\mathbf{J}^{-1}\mathbf{c}\right\|_2^2 + \Gamma(\mathbf{J}^{-1}\mathbf{c}), \tag{22}$$

where $\Gamma(\mathbf{J}^{-1}\mathbf{c})$ is simply $F(\mathbf{K}\mathbf{J}^{-1}\mathbf{c})$. Hence, we can add this compression-space optimization to any image-formation model and priors, directly optimizing for the best JPEG DCT coefficients $\mathbf{c}$. If we do not want to optimize all the way to the JPEG coefficients, but only to a YUV or other color space representation, we can set $\mathbf{A} = \mathbf{M}\mathbf{D}\mathbf{B}\mathbf{C}^{-1}\mathbf{U}_J$.

Note that we can use a similar approach for JPEG image decompression in an image viewer. Depending on the quantization matrix $\mathbf{Q}$, JPEG-compressed images may exhibit severe artifacts, including blocking, ringing, and loss of texture. We can use our usual image priors to alleviate these artifacts by using $\mathbf{A} = \mathbf{J}$ to get

$$G(\mathbf{x}) = \|\mathbf{c} - \mathbf{J}\mathbf{x}\|_2^2, \tag{23}$$

where $\mathbf{c}$ are the JPEG coefficients, and $\mathbf{x}$ the reconstructed image. Intuitively, using Eq. 23 as the data term in our minimization scheme determines the image that best matches the image priors while still compressing to the given JPEG coefficients $\mathbf{c}$. In effect, we are deblocking the displayed JPEG image [Foi et al. 2007].

4.7 Applying Our Framework to New Applications

We have described a large number of different applications. This is possible due to the flexibility of our system; adapting it to a new image formation model is very easy and only requires changing the matrices $\mathbf{A}$ and $\mathbf{M}$ with a few lines of code (see supplemental material).

5 Design Choices

We now discuss our decisions that led to the proposed framework.

5.1 Optimization Method

While many optimization methods can be used to solve Eq. 2, we decided to pose our framework as a non-linear optimization problem (Eq. 3). In order to facilitate code reuse, we opted to express our priors as proximal operators, which are used by a number of optimization methods, such as ADMM [Parikh and Boyd 2013] and the primal-dual method [Chambolle and Pock 2011]. This enabled us to quickly test both methods with much of the same code. We decided to use the primal-dual method as it turned out to converge more quickly than ADMM. However, ADMM is easier to program and requires less code.

Convergence Both ADMM and the primal-dual algorithm are guaranteed to converge to the global optimum for convex F and G, with the primal-dual algorithm having provably optimal convergence. However, we make use of non-convex regularization terms in F, so theoretical convergence guarantees are not available. That said, we always combine both convex and non-convex terms in F (a deliberate choice, see the following subsection), and therefore can control the degree of non-convexity simply by adjusting the weights for the individual terms. This weight selection depends on the ill-posedness of the data term G for a given camera design and image formation model $\mathbf{A}$, but in our experiments we have not found a need to vary it based on the image content. In practice, we get convergence for all initial iterates we tried (even for $\mathbf{x}^0 = \mathbf{0}$, see Fig. 4).

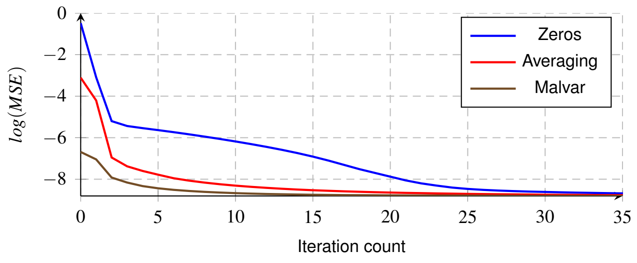


Figure 4: *Effects of different initial iterate $\mathbf{x}^0$ on the convergence for a simple demosaicking example. We compare starting with $\mathbf{x}^0 = \mathbf{0}$, averaging of neighbors, and the output of a classical demosaicking method. All cases converge to the same solution; with a good starting point, 5 iterations suffice. More details in the supplemental.*

5.2 Choice and Importance of Priors

The choice and combination of priors is crucial for high-quality results [Mitra et al. 2014]. We now detail our reasoning for our choice of priors. The priors can be divided into two main categories: *internal* priors use intra-image information, while *external* priors use external knowledge about natural images. While most priors fall in one or the other category, a few have mixed characteristics, so the boundary is somewhat vague. We have experimented with many internal and external priors to yield a combination that achieves the highest quality but remains efficient to compute. We next analyze the internal and external priors separately.

5.2.1 Choice of External Priors

We considered three different external priors (in addition to the complementary cross-channel prior; for its analysis we refer the reader to Heide et al. [2013]): the simple TV prior [Rudin et al. 1992], a curvelet prior [Candès and Donoho 1999], and the EPLL prior [Zoran and Weiss 2011]. We explore their use for reconstructing interlaced HDR (Sec. 4.3) images with a simulated lens blur (Gaussian, varying σ) using a synthetic dataset consisting of 12 different images. This dataset provides ground truth and helps to determine which priors are adequate. We set the internal prior to BM3D and reconstruct the 12 images by varying the weights between the internal (BM3D) and the three investigated external priors. The reconstruction quality is listed in Table 3.

The complex EPLL prior can give a slight image quality increase (f) in the case of a large blur ($\sigma = 1.83$), but its computational cost is prohibitive (60 times higher than TV). Just using TV gives a similar boost. For a smaller lens blur ($\sigma = 0.14$), EPLL also only yields a minor increase in image quality, but larger gains are achieved by the much cheaper TV prior. The curvelet prior does not provide any benefit for $\sigma = 1.83$; on the contrary, the quality decreases slightly with its use. For $\sigma = 0.14$, the curvelet prior can achieve gains, but is quite sensitive to the chosen weighting. Furthermore, using the curvelet prior is about 15% slower than using TV.

We note that the benefit of using TV in addition to BM3D lies in making the problem more convex. To further illustrate this, we show results for five different Gaussian blurs (bottom of Table 3). Smaller blurs make the data term less convex, thus making the problem successively more difficult to solve by only using BM3D. Adding TV convexifies the problem, and image quality increases. Convergence plots (see supplemental) demonstrate that adding TV makes the solver more stable and makes it converge faster than just using the non-linear, non-convex BM3D prior.

Figure 5 demonstrates the complementary benefits of using BM3D and TV. The top row illustrates that by exploiting structural self-similarity (BM3D) we can reconstruct significantly more detail than just by using a TV prior. The bottom row illustrates that the BM3D prior may fail for correlated noise or reconstruction artifacts, but those problems can be fixed by the external TV prior.

We conclude that TV is the most cost-effective external prior giving similar performance as much more computationally complex priors and can help to increase convexity. In all our applications, a small amount of TV has improved the resulting reconstruction quality. The cross-gradient prior further complements the TV and BM3D priors.

5.2.2 Choice of Internal Priors

We showed in Sec. 3.3.2 that we can incorporate *any* Gaussian denoiser into our framework. We are particularly interested in *internal* denoising priors that exploit self-similarity within the image [Buades et al. 2005; Dabov et al. 2007a]. We experimented with the most-commonly-used non-local operators: BM3D [Dabov et al. 2007a], NLM [Buades et al. 2005], patchwise NLM [Buades et al. 2011], Sliding DCT [Egiazarian et al. 1999], and simple averaging of similar patches, which we compared for demosaicking, interlaced HDR, and burst image stack processing in Fig. 6.

All denoisers except Sliding DCT operate on a stack of similar patches. For this experiment, we configured all of them to search for the 16 most-similar 8×8 patches in a 15×15 pixel vicinity. BM3D performs a 3D transform (DCT on *xy* and Haar on *z*) on this stack

	(a)	(b)	(c)	(d)	(e)	(f)	(g)
$\phi_0^{\text{ext.}}$	0	1/32	1/16	1/4	1/2	1/2	1
ϕ_1^{BM3D}	1	1	1	1	1	1/2	0
EPLL (σ = 1.83)	0.00	+0.01	0.00	0.00	-0.03	+0.07	-0.34
Curvelets (σ = 1.83)	0.00	-0.02	-0.01	-0.41	-1.03	-0.90	-1.78
TV (σ = 1.83)	0.00	+0.03	+0.04	+0.05	-0.02	-0.19	-0.61
EPLL (σ = 0.14)	0.00	0.00	+0.03	+0.04	+0.07	+0.05	-0.82
Curvelets (σ = 0.14)	0.00	-0.02	-0.02	-0.25	+0.12	-0.05	-0.40
TV (σ = 0.14)	0.00	+0.04	+0.13	+0.16	+0.02	-0.39	-0.95
TV (σ = 1.83)	0.00	+0.03	+0.04	+0.05	-0.02	-0.19	-0.61
TV (σ = 1.38)	0.00	+0.05	+0.10	+0.12	+0.03	-0.18	-0.79
TV (σ = 1.00)	0.00	+0.05	+0.12	+0.14	+0.05	-0.21	-1.12
TV (σ = 0.55)	0.00	+0.02	+0.05	+0.10	0.00	-0.40	-1.23
TV (σ = 0.14)	0.00	+0.04	+0.13	+0.16	+0.02	-0.39	-0.95

Table 3: *Different weighting between internal and external prior and the resulting change in SNR.*

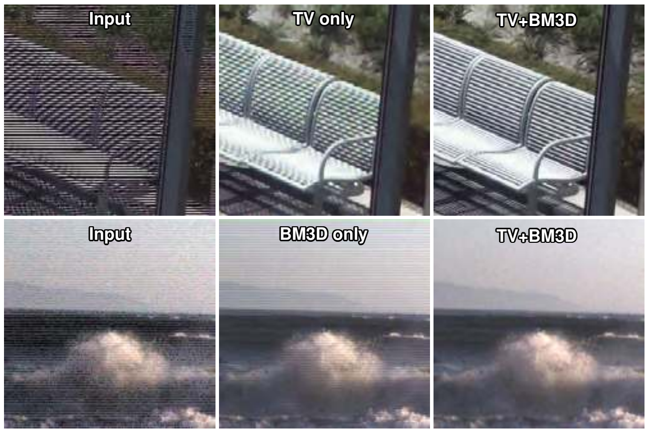


Figure 5: *Illustration of using both the TV and BM3D priors for reconstructing images from the interlaced HDR sensor (Sec. 4.3).*

and thresholds all coefficients followed by Wiener filtering; NLM computes a weighted average of the patch centers based on patch similarity; patchwise NLM computes the weighted average for all 8×8 pixels; simple averaging assigns a uniform weight to all patches. Sliding DCT simply thresholds DCT-transformed patches for each pixel and aggregates the results. We only use the denoising prior for this comparison.

Figure 6 shows the resulting PSNR values for each denoiser. Overall, BM3D and patchwise NLM achieve very similar results, with BM3D being slightly better in all applications. Hence, all results in this paper were generated using BM3D. Simple averaging works well for reconstructing interlaced HDR images and burst denoising, but not for Bayer demosaicking. Pure NLM achieves reasonable PSNR values in all applications, but is always worse than BM3D and patchwise NLM. Sliding DCT achieves fairly good PSNR for demosaicking and burst denoising, but not when applied to interlaced HDR with a significant number of pixels being either too dark or bright. Since significant information is missing in interlaced HDR, a self-similarity prior is needed to recover it.

BM3D modifications One can expect improved results by modifying the BM3D parameters for a given application, and we experimented with this for demosaicking. Note that we only varied the prior parameters (just like the data-term weights for different applications), but not the prior itself. We found that two changes to the standard BM3D parameters both improve the running time and increase the result image quality. The first parameter is the size of the patch to be matched. Instead of using the standard 8×8 patch, we use much smaller 4×4 patches. Obviously, smaller patches are

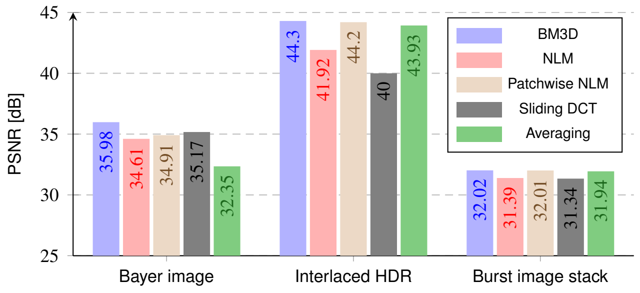


Figure 6: *The impact of selected natural-image priors on the reconstruction quality of various image models.*

faster to compare, but more importantly, it is likely that we can find more patches that are similar. This is particularly important in areas with irregular textures, such as foliage or grass.

The second parameter relates to the color spaces that we operate in, and the 2D transformations we use. In the first stage, when patches are matched, the modified version uses the YUV color space, and matches patches based on the luminance component only; we use DCT as the 2D spatial transform to sparsify the signal, and 1D Haar for thresholding. For the second stage, Wiener filtering, we use the 3-point DCT to decorrelate colors and then use DST (Discrete Sine Transform) as the 2D spatial transformation, and again Haar for thresholding. Use of different color spaces and transformations decorrelates the two processing steps, yielding improved results. The numbers reported in Sec. 6.1 use these improved settings. The average improvement is +1.73 dB in pure demosaicking and +0.6 dB in joint demosaicking and denoising over the default BM3D. Please note that, *even with the default settings, our technique still beats the competing methods*.

5.3 Choice of Weights

We found that a single setting for the prior weights is sufficient for each application—no image-dependent modifications are necessary. Furthermore, we need to find the right overall weighting between the data term and the priors (Eq. 14), as illustrated in Fig. 7. We experimented with different weights, and again found that a single setting per application works well for any given image. User preferences such as sharp but grainy vs. smooth with less noise, do influence the settings. All settings are detailed in the supplemental material.

6 Results

Now we present our results for various applications.

6.1 Demosaicking of Conventional Bayer Images

First, we demonstrate that our system outperforms state-of-the-art demosaicking methods. Zhang et al. [2011] compared several algorithms on the McMaster color image dataset. We follow their procedure and additionally run our method, DCRaw, and Adobe Photoshop on the same dataset. Figure 8 lists the results of the comparison. The full table and set of images are provided in the supplemental material; representative examples are shown in Fig. 10. Our framework shows a gain of 2.23 dB over the best existing demosaicking methods. Note that this is a very significant improvement, given that demosaicking is a mature problem with hundreds of publications over the past decade.

In Fig. 9, we compare joint demosaicking and denoising against other state-of-the-art methods, following Jeon and Dubois [2013].

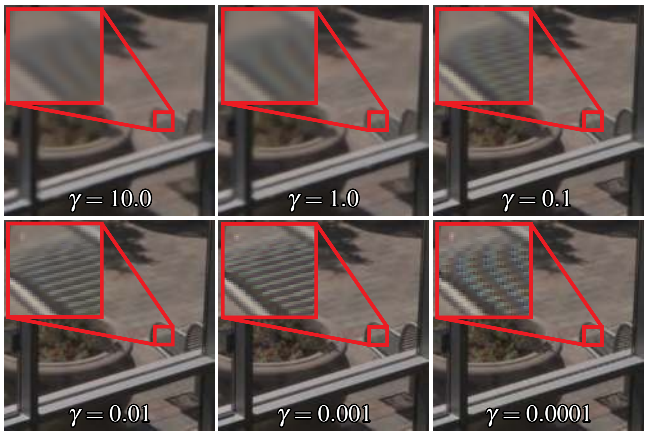


Figure 7: *Impact of the regularizer weights (relative to the data term) on the solution. The weights are $\gamma_i = \gamma\phi_i$: γ is shown in the images, while ϕ_i are $\phi_0 = 0.1$ (TV), $\phi_1 = 1.0$ (BM3D), and $\phi_2 = 0.1$ (cross-channel). High γ-values impair the reconstruction process by enforcing too much sparsity in the output. On the other hand, a too small γ ignores the priors, leading to sharp but aliased results.*

Our method outperforms all competing methods by at least 0.68 dB. A detailed list of PSNR values can be found in the supplemental.

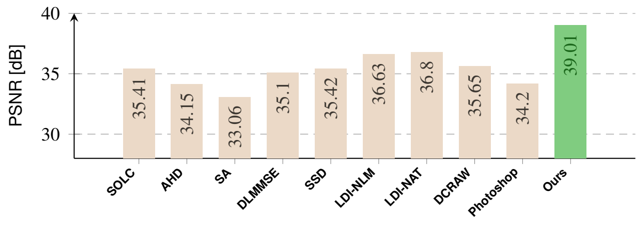


Figure 8: *Demosaicking results for the McMaster color image dataset. We show average PSNR over all images and color channels.*

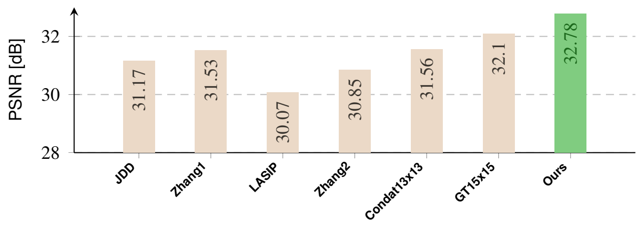


Figure 9: *Joint demosaicking and denoising results for the Kodak color image dataset. We show average PSNR over all images, color channels, and sigmas.*

6.2 Deblurring of Out-of-focus and Camera-shake Blur

Deconvolution methods differ mostly by the image priors used. We adopt the comparison of Schuler et al. [2013] with five state-of-the-art priors: EPLL (Gaussian mixture models) [Zoran and Weiss 2011], heavy-tailed gradient distributions [Levin et al. 2007; Krishnan and Fergus 2009], self-similarity [Danielyan et al. 2012; Dabov et al. 2008], Field-of-Experts [Roth and Black 2009], and neural networks [Schuler et al. 2013].

Table 4 shows a quantitative comparison for five different blur and noise scenarios (a)–(e) averaged over a test-set of 11 images (see Schuler et al. [2013], Table 1). We outperform MLP (and others

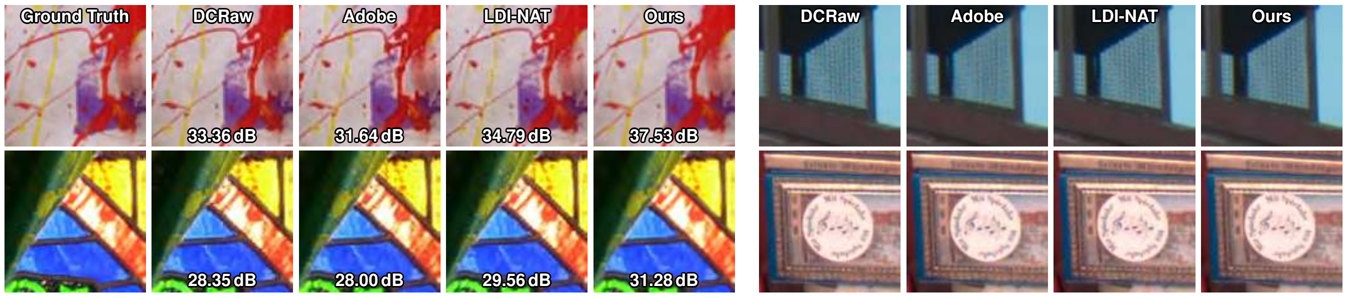


Figure 10: *We compare our demosaicking method with commonly used demosaicking tools (DCRaw and Adobe Camera Raw) and the best competing published method (LDI-NAT) on two patches of the McMaster dataset (left) and on two real-world examples (right). Note how our method produces results virtually indistinguishable from ground truth, whereas other methods contain visible artifacts (please zoom in to the pdf). More examples can be found in the supplemental.*

Deconvolution method	Reconstruction PSNR [dB]				
	(a)	(b)	(c)	(d)	(e)
EPLL [2011]	24.04	26.64	21.36	21.04	29.25
[Levin et al. 2007]	24.09	26.51	21.72	21.91	28.33
[Krishnan and Fergus 2009]	24.17	26.60	21.73	22.07	28.17
DEB-BM3D [2008]	24.19	26.30	21.48	22.20	28.26
IDD-BM3D [2012]	24.68	27.13	21.99	22.69	29.41
FoE [2009]	24.07	26.56	21.61	22.04	28.83
MLP [Schuler et al. 2013]	24.76	27.23	22.20	22.75	29.42
Ours	**24.83**	**27.31**	**22.24**	**22.89**	**29.78**

Table 4: *Reconstruction quality for five non-blind deconvolution scenarios as in Schuler et al. [2013].*

more clearly), with the highest margin in the motion blur case (e), which probably is the most common application of deconvolution. Our mixed BM3D + TV (+ cross-channel) prior outperforms both BM3D-based and gradient-based methods. Both internal and external priors are needed for peak performance in image reconstruction, only one is usually not enough [Mosseri et al. 2013]. The images in this dataset were grayscale, so the cross-channel prior could not be used. For visual comparisons, including color images using the cross-channel prior, see the supplemental material.

6.3 HDR Image Reconstruction

The interlaced HDR sensor poses a challenging reconstruction problem. For scenes with a wide dynamic range, we use an exposure time ratio as high as 16:1, so that odd macro-rows are exposed for 16 times longer than the even macro-rows. Now the short exposure has a better chance to catch highlights and the long exposure can capture shadows; unfortunately, short exposure may be too noisy in shadows and long exposure may saturate in bright areas. While the reduced number of valid input observations increases the difficulty of the reconstruction task, our approach addresses this case efficiently. The self-similarity prior of BM3D or NLM is now crucial to meaningfully filling in the missing data. We demonstrate this in Fig. 11 (a), where we compare our method to the standard image processing pipeline of demosaicking (DCRaw) plus HDR de-interlacing of Gu et al. [2010], which is based on upsampling. We also compare against the *Magic Lantern* firmware, which provides an efficient method by Hajisharif et al. [2014] for fusing interlaced HDR images (some Canon EOS sensors can capture interlaced dual-ISO images)—followed by demosaicking with DCRaw, since Magic Lantern only outputs mosaicked data. Note how Magic Lantern produces some artifacts, whereas our reconstruction is virtually artifact-free.

In Fig. 11 (b, c top) we show the sparse input observations, with pixels marked black when indicated by the matrix $\mathbf{M}$ as too noisy or saturated (simple thresholding on intensities). Even though a significant portion of the input data is missing, our unified optimization successfully reconstructs geometric structures, including detailed textures (grass) and thin edges. Also, in Fig. 11 (d) we demonstrate its performance on scenes with local motion (a runner), where different amounts of motion blur in short- and long-exposure rows create a challenging interpolation problem. By default, our optimizer prefers the longer blur of the longer exposure, avoiding noise from the short exposure, and producing a visually consistent result.

6.4 Fusion of Color Array Camera Images

We next apply our unified optimization framework to color-channel fusion from a 2×2 array. We perform the experiment on both a large and a small baseline design. In Fig. 12 we show the large-baseline (top) and the small-baseline results (bottom). The first column shows an overlay of the different channels to illustrate the extent of chromatic aberration due to the parallax. The second column shows the images warped by the computed cross-channel flow. Note that there are still color artifacts in the occluded and poorly-estimated flow regions. In the third column we show the result by a method adapted from Joulin and Kang [2013], which was originally proposed to reconstruct full color stereo images from anaglyph (red/cyan) images. This method works by iteratively computing SIFTFlow [Liu et al. 2008] between the channels, detecting poorly matching regions through flow consistency check, and then colorizing these regions. But the method does not enforce the cross-channel edge consistency illustrated in Fig. 12, and produces visible artifacts. In the fourth column we show that our combination of priors manages to maintain the consistency of R and B channels with the G channel in just a single pass and avoids color fringing. While some small artifacts remain, this problem is hard and we believe we have made a large step towards a robust solution. *Note that we do not require a perfect registration*, but rely on our image priors to refine the effective registration. This is an important advantage, because registration errors are unavoidable—whether images are registered via optical flow or via depth estimates [Venkataraman et al. 2013]—and may lead to very visible color bleeding at occlusion edges (Fig. 12). Our conclusion is that the proposed framework is suitable for processing color camera images, even without first extracting per-pixel depths, but only operating in the 2D image plane.

6.5 Burst Denoising and Demosaicking

We demonstrate joint denoising and demosaicking on a real-world and a simulated dataset in Fig. 13. To account for multiplicative and Poisson noise in this application, we perform the generalized Anscombe transform to stabilize variance [Starck et al. 1998]. It is applied to the observation $\mathbf{z}$, transforming the observations to fulfill

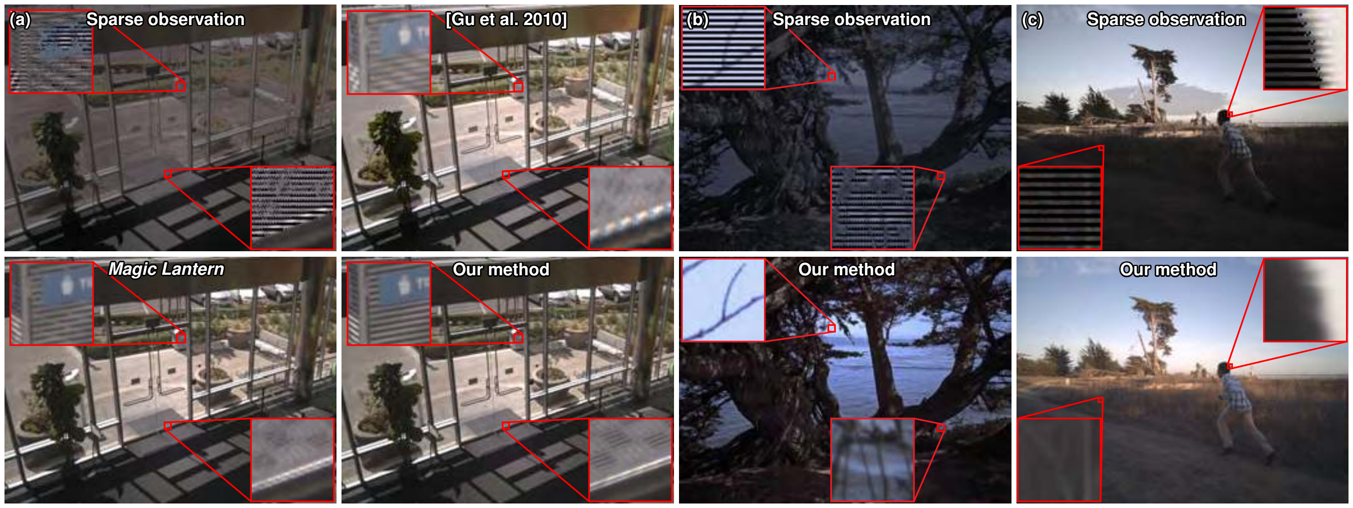


Figure 11: *Image reconstruction for interlaced HDR sensor data.* ***(a)*** *Upsampling-based reconstruction [Gu et al. 2010] improves the dynamic range, but fails to recover highly structured details. The* Magic Lantern *method provides higher quality, but still loses resolution and produces artifacts. Our method generates an artifact-free result and provides the best overall reconstruction quality.* ***(b, c)*** *In addition to extending the dynamic range (bottom row), we account for local scene motion and simultaneously denoise and reduce interlacing artifacts (see insets).*

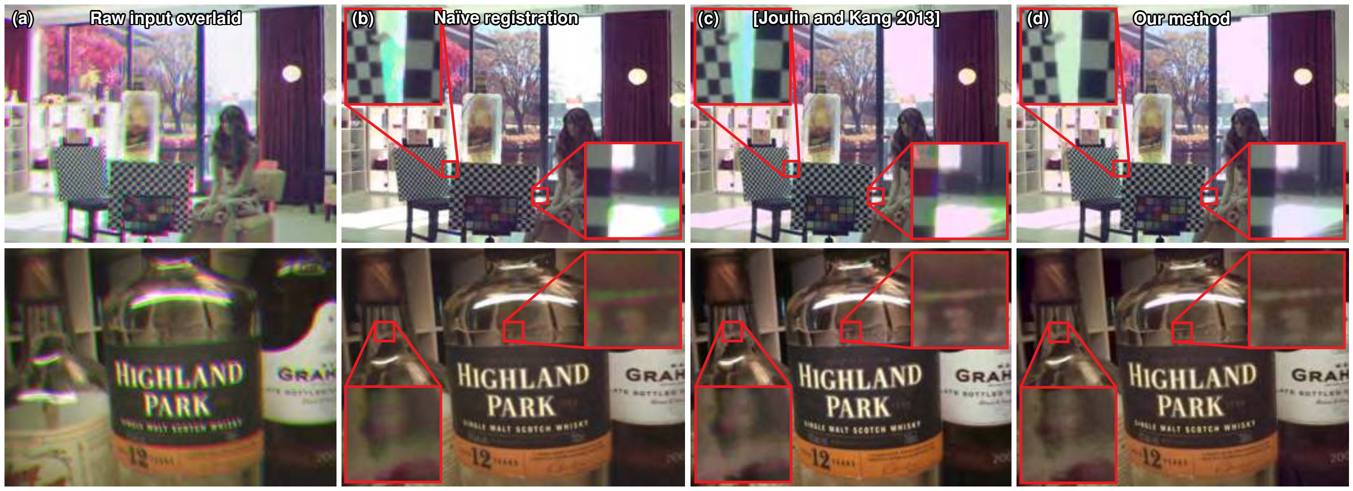


Figure 12: *Image reconstruction for color-array camera.* ***(a)*** *Raw input red, green, and blue images overlaid.* ***(b)*** *Naïve registration of color channels via optical-flow produces color leaks.* ***(c)*** *Anaglyph-based reconstruction [Joulin and Kang 2013] partially reduces the color leakage.* ***(d)*** *Our method reconstructs the image without such artifacts.*

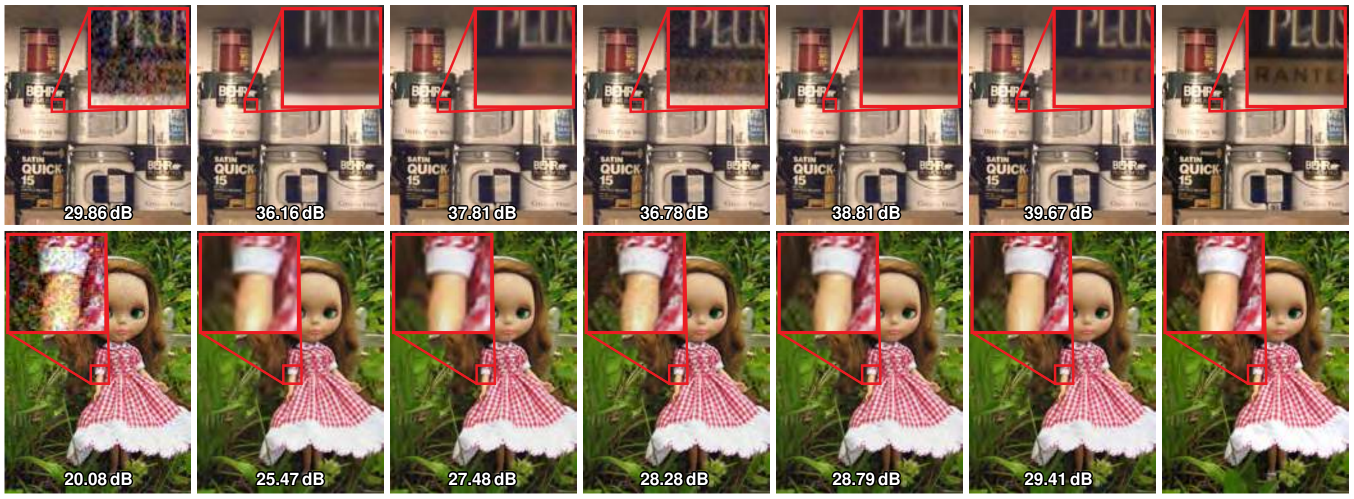
our additive Gaussian model assumption in the data term.

We compare our technique (column (f)) to a number of standard processing pipelines for burst denoising. Demosaicking the first frame (a) [Zhang et al. 2011] is the simplest approach, but also produces the worst results. Demosaicking followed by the state-of-the-art BM3D denoising method (b) significantly improves the image quality but is still inferior to our approach, which is not surprising, as the full stack contains a lot more information than a single image. We then compare against VBM3D [Dabov et al. 2007b] on the (aligned) stack of 8/16 demosaicked images, which yields good but still slightly inferior results (c). Just blending the aligned, demosaicked images together using exponential weights (as used in NLM [Buades et al. 2005]) computed from a 15×15 patch around each pixel produces surprisingly good, albeit still slightly noisy images (d). Applying BM3D to the NLM-weighted, demosaicked stack achieves good PSNR numbers (e), but is still inferior to our proposed framework (column (f)) which jointly optimizes for the best latent image given the noisy stack of images.

6.6 JPEG and Beyond RGB

To evaluate our joint optimization with JPEG's DCT encoding using Eq. 22, we have reconstructed 12 images from the interlaced HDR camera application. We first compare our joint optimization by directly reconstructing these images in YUV420 vs. first reconstructing them as full RGB and then subsampling and converting to YUV420. On these 12 images, our method achieves an average PSNR of 28.83 dB, in contrast to the pipeline approach, which only achieves 28.45 dB. Some examples are shown in the supplemental.

We also compare JPEG images reconstructed directly from the interlaced HDR inputs by our framework (quality 50) to the standard pipeline approach of first reconstructing in full RGB, followed by regular JPEG encoding (using the quality factor that best matches the file size of the first JPEG). PSNR values of the decoded JPEG images are then computed w.r.t. the ground-truth image. On average, we achieve an improvement of 0.1 dB over the sequential approach.



(a) First Frame of Stack **(b)** BM3D (First Frame) **(c)** VBM3D (Stack) **(d)** NLM-weighted Stack **(e)** NLM Stack + BM3D **(f)** Ours **(g)** Ground Truth

Figure 13: *Burst Image Comparison. The first row shows a real-world example taken in low light (8 images at ISO 12800 on a Canon EOS 650D; GT was taken at ISO 100). Each image in the simulated doll dataset (16 images, courtesy Flickr user susan402) has multiplicative white Gaussian noise with $\sigma = 0.1$ and additive white Gaussian noise with $\sigma = 0.1$. Parameters for each method were optimized for highest PSNR.*

The improvement is quite modest, as the JPEG coefficients do not couple much with the rest of the pipeline. However, strong JPEG compression inevitably creates blocking and quantization artifacts, which this approach cannot address. To overcome such issues, we can extend the image pipeline all the way to decompression and solve Eq. 23 on the receiver side, in essence deblocking the JPEG image [Foi et al. 2007].

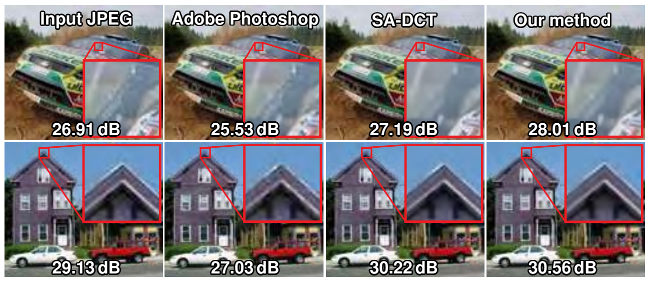


Figure 14: *Extending the reconstruction pipeline to the receiver. We compare the JPEG (quality = 30), Adobe Photoshop deblocking, SA-DCT, and our reconstruction on the display side (PSNR in images). (Images courtesy of Wikipedia user kallerna, and A. Torralba & B. Russell / LabelMe dataset.)*

We compare our method to Adobe Photoshop CC's deblocking algorithm and SA-DCT [Foi et al. 2007] in Fig. 14. Our method removes most of the blocking artifacts and ringing, and yields a significant PSNR improvement ($\sim$1.0 dB). Photoshop removes some artifacts, but does not reconstruct the original image as faithfully as our method does. SA-DCT achieves a good visual quality, but our reconstruction is about 0.3–0.6 dB better.

6.7 Performance

We have implemented FlexISP system in both Matlab (CPU) and CUDA (on desktop and tablet GPUs). We give the run times for several different applications and denoising priors in Table 5. Unsurprisingly, the CPU-implementation is slow, with a single iteration of BM3D (using the implementation from authors' web-page) taking 149.39 seconds for a 13 MPix image. Apart from a BM3D step, each iteration of our solver runs a conjugate gradient step and enforces the other priors. Full numerical convergence is usually reached in about 30 iterations as shown in Fig. 4 (note that the vertical scale is logarithmic) and in the supplemental. The overall computation cost seems high, however, in practice running 4–5 iterations from a good starting point gets close enough to the converged result and produces images with sufficient quality.

Our GPU-version is optimized for speed and memory usage. The self-similarity based denoising priors (BM3D, NLM, etc.) are accelerated with an approximate-nearest-neighbor (ANN) method [Tsai et al. 2014]. Furthermore, we split large images into tiles and process these tiles separately to save memory. We found that a small overlap of only four pixels is usually sufficient to prevent visible boundaries at tile borders. Now all intermediate buffers, even combined, consume less memory than the output image, enabling a memory-friendly image reconstruction. As a result, we can process high-resolution images in a matter of seconds on a recent desktop GPU (NVIDIA GTX TITAN) and can even enable interactive previews (1.5 sec) on a recent NVIDIA Tegra K1 tablet for 1 MPix images. As expected, the use of ANN leads to a slightly reduced quality compared to the Matlab implementation (of about -0.6 dB to -1.0 dB depending on the application). We have also experimented with a GPU-based accurate nearest neighbor search. While this increases the GPU run-time, e.g., when using BM3D as a prior it increases by a factor of 4, it achieves full accuracy and is still magnitudes faster than the CPU implementation. All the PSNR values provided in this paper were computed using the Matlab implementation.

7 Discussion and Limitations

Priors Good priors help to solve inverse problems that are ill-posed due to missing, noisy, or blurry data. Instead of choosing just one, we have concentrated on three: TV, the denoising / self-similarity prior, and the cross-channel prior. They are very useful, but they may also introduce artifacts: *TV* can create unnatural "watercolor-like" edges; *self-similarity* helps to reconstruct missing data, but it only works well if the image actually exhibits self-similarity—in regions with unique patterns, such as vegetation, it may over-smooth; and while the *cross-channel gradient prior* helps to avoid color fringes, it sometimes mistakes colored pixels for ar-

	Demosaic	HDR	Color Array	Burst
Resolution (MPix)	5	13	0.6×3	0.4×16
Iterations	4	5	5	5
CPU BM3D	312.36 s	1471.55 s	74.21 s	303.21 s
CPU NLM	196.45 s	1094.84 s	52.41 s	291.35 s
CPU Patchwise NLM	198.72 s	1102.23 s	52.83 s	291.80 s
CPU Sliding DCT	89.63 s	747.67 s	32.31 s	280.89 s
CPU Averaging	194.17 s	1087.45 s	51.98 s	291.35 s
GTX Titan BM3D	2.13 s	7.45 s	0.94 s	0.82 s
GTX Titan NLM	0.73 s	2.78 s	0.19 s	0.30 s
GTX Titan Patchwise NLM	0.77 s	2.94 s	0.20 s	0.32 s
GTX Titan Sliding DCT	0.29 s	1.20 s	0.13 s	0.27 s
GTX Titan Averaging	0.45 s	1.89 s	0.14 s	0.30 s
Tegra K1 BM3D	40.5 s	147.4 s	18.1 s	16.7 s
Tegra K1 NLM	26.5 s	56.1 s	3.8 s	7.3 s
Tegra K1 Patchwise NLM	28.0 s	59.2 s	4.0 s	7.7 s
Tegra K1 Sliding DCT	7.0 s	30.9 s	2.8 s	7.0 s
Tegra K1 Averaging	6.7 s	33.2 s	2.2 s	6.6 s

Table 5: *Running times for: desktop CPU (Core i7 2.4 Ghz), desktop GPU (NVIDIA GTX TITAN), and mobile GPU (NVIDIA Tegra K1).*

tifacts and removes the color altogether. However, combining the priors appears to overcome their individual disadvantages.

While our priors are expressive, there are situations where they do not provide enough information. For instance, in the interlaced HDR application, if two consecutive rows are missing a nearly horizontal thin line, it may be difficult for the priors to connect the segments. Future priors that take content-aware information into account could help in such situation.

In our approach we do not exploit the bound information given by the data, such as the observation that a saturated (input) pixel should have a very high value after reconstruction. Using these bounds could improve the results. However, since we observe that our optimizer tends to retain these areas bright, and explicitly accounting for this would require constrained optimization, we did not choose to include it in our method.

Failure cases If an image violates our priors, e.g., the input offers no self-similarity (random noise for example), the result can appear over-smoothed. Misconfiguring the solver parameters can lead to overly sharp (the last row in the supplemental, Fig. 4) or overly smooth results (Fig. 7). Large errors in the warp estimation in burst imaging and color-array camera lead to inaccurate data-term operators, and artifacts may still appear (e.g., around boundaries). This can be seen, for instance, in Fig. 12 (d) in the supplemental. However, we note that many of the small errors in warp estimation are easily corrected by our choice of the cross-channel and self-similarity priors.

Processing times While our system is very fast considering what it can do, on a mobile device it cannot match the processing times of an embedded ISP with a specialized ASIC. In a practical camera, a traditional ISP could preprocess the image for immediate viewing, and a higher-quality image could be processed in background, or by a cloud service. Using the ISP-processed image as initialization would also give a better starting point, requiring fewer iterations for the optimizer to converge.

8 Conclusion

We have presented FlexISP, a framework and a system that replaces the traditional image processing pipeline for reconstructing photographs from raw sensor data by a single, integrated, and flexible system that is based on global optimization. The image formation model and any image priors and regularization terms are expressed as a single objective function, which is solved using a proximal operator framework. Proximal operators decouple the individual terms in the objective function in a principled way, making it possible to separately implement the operators for the data term and each regularizer. This approach enables mixing and matching different, highly optimized implementations of data terms and regularizers. Our framework therefore achieves both the improved image quality of a fully integrated optimization and the separation of concerns that gave rise to the traditional pipeline approach.

We detail and analyze our design choices, and conclude that a single specific set of priors can be used for a variety of applications, and still outperforms other state-of-the-art methods. While we demonstrate significant improvements in quality and simplicity for traditional camera designs, we believe that our approach will achieve its full potential with future computational cameras that have significantly more complex image formation models.

References

ADAMS, A., TALVALA, E.-V., PARK, S. H., JACOBS, D., AJDIN, B., GELFAND, N., DOLSON, J., VAQUERO, D., BAEK, J., TICO, M., LENSCH, H. P. A., MATUSIK, W., PULLI, K., HOROWITZ, M., AND LEVOY, M. 2010. The Frankencamera: an experimental platform for computational photography. *ACM TOG 29*, 4.

AFONSO, M. V., BIOUCAS-DIAS, J. M., AND FIGUEIREDO, M. A. 2010. Fast image recovery using variable splitting and constrained optimization. *IEEE TIP 19*, 9.

AJDIN, B., HULLIN, M. B., FUCHS, C., SEIDEL, H.-P., AND LENSCH, H. 2008. Demosaicing by smoothing along 1d features. In *CVPR*.

BENNETT, E. P., UYTTENDAELE, M., ZITNICK, C., SZELISKI, R., AND KANG, S. 2006. Video and image bayesian demosaicing with a two color image prior. In *ECCV*.

BRAUERS, J., SEILER, C., AND AACH, T. 2010. Direct psf estimation using a random noise target. In *Electronic Imaging*.

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

BUADES, T., LOU, Y., MOREL, J. M., AND TANG, Z. 2009. A note on multi-image denoising. In *Int. Workshop on Local and Non-Local Approximation in Image Processing*.

BUADES, A., COLL, B., AND MOREL, J.-M. 2011. Non-Local Means Denoising. *Image Processing On Line*.

CANDÈS, E., AND DONOHO, D. 1999. Curvelets: A surprisingly effective nonadaptive representation of objects with edges. In *Curves and Surfaces*. Vanderbilt University Press.

CHAMBOLLE, A., AND POCK, T. 2011. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision 40*, 1.

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

COIFMAN, R. R., AND DONOHO, D. L. 1995. Translation-invariant denoising. Tech. Rep. 475, Stanford University.

DABOV, K., FOI, A., KATKOVNIK, V., AND EGIAZARIAN, K. 2007. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE TIP 16*, 8.

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

DABOV, K., FOI, A., KATKOVNIK, V., AND EGIAZARIAN, K. 2008. Image restoration by sparse 3d transform-domain collaborative filtering. In *Electronic Imaging*.

DANIELYAN, A., KATKOVNIK, V., AND EGIAZARIAN, K. 2012. BM3D frames and variational image deblurring. *IEEE TIP 21*, 4.

EGIAZARIAN, K. O., ASTOLA, J., HELSINGIUS, M., AND KUOSMANEN, P. 1999. Adaptive denoising and lossy compression of images in transform domain. *J. Electronic Imaging 8*, 3.

FARSIU, S., ELAD, M., AND MILANFAR, P. 2006. Multiframe demosaicing and superresolution of color image. *IEEE TIP 15*, 1.

FOI, A., KATKOVNIK, V., AND EGIAZARIAN, K. 2007. Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. *IEEE TIP 16*, 5.

GU, J., HITOMI, Y., MITSUNAGA, T., AND NAYAR, S. 2010. Coded rolling shutter photography: Flexible space-time sampling. In *ICCP*.

HAJISHARIF, S., KRONANDER, J., AND UNGER, J. 2014. HDR reconstruction for alternating gain (ISO) sensor readout. In *Proc. of Eurographics (short paper)*.

HEIDE, F., ROUF, M., HULLIN, M. B., LABITZKE, B., HEIDRICH, W., AND KOLB, A. 2013. High-quality computational imaging through simple lenses. *ACM TOG 32*, 5.

IRANI, M., AND PELEG, S. 1991. Improving resolution by image registration. *Graphical Models and Image Processing 53*, 3.

JEON, G., AND DUBOIS, E. 2013. Demosaicking of noisy Bayer-sampled color images with least-squares luma-chroma demultiplexing and noise level estimation. *IEEE TIP 22*, 1.

JOULIN, A., AND KANG, S. B. 2013. Recovering stereo pairs from anaglyphs. In *CVPR*.

KRISHNAN, D., AND FERGUS, R. 2009. Fast image deconvolution using hyper-laplacian priors. In *NIPS*.

LEVIN, A., FERGUS, R., DURAND, F., AND FREEMAN, W. T. 2007. Deconvolution using natural image priors. *ACM Transactions on Graphics (TOG) 26*, 3.

LIU, C., YUEN, J., TORRALBA, A., SIVIC, J., AND FREEMAN, W. T. 2008. Sift flow: Dense correspondence across different scenes. In *ECCV*.

LIU, C. 2009. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, MIT.

MITRA, K., COSSAIRT, O., AND VEERARAGHAVAN, A. 2014. A framework for analysis of computational imaging systems: Role of signal prior, sensor noise and multiplexing. *IEEE PAMI 36*, 10.

MOSSERI, I., ZONTAK, M., AND IRANI, M. 2013. Combining the power of internal and external denoising. In *ICCP*.

NARASIMHAN, S. G., AND NAYAR, S. K. 2005. Enhancing resolution along multiple imaging dimensions using assorted pixels. *IEEE PAMI 27*, 4.

NAYAR, S., AND BRANZOI, V. 2003. Adaptive dynamic range imaging: Optical control of pixel exposures over space and time. In *ICCV*.

OYMAK, S., AND HASSIBI, B. 2013. Sharp MSE bounds for proximal denoising. *arXiv preprint arXiv:1305.2714*.

PARIKH, N., AND BOYD, S. 2013. Proximal algorithms. *Foundations and Trends in Optimization 1*, 3.

RAMANATH, R., SNYDER, W. E., YOO, Y., AND DREW, M. S. 2005. Color image processing pipeline in digital still cameras. *IEEE Signal Processing Magazine 22*, 1.

REINHARD, E., WARD, G., PATTANAIK, S., DEBEVEC, P., HEIDRICH, W., AND MYSZKOWSKI, K. 2010. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann.

ROTH, S., AND BLACK, M. J. 2009. Fields of experts. *International Journal of Computer Vision 82*, 2.

RUDIN, L. I., OSHER, S., AND FATEMI, E. 1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena 60*, 1.

SCHULER, C. J., HIRSCH, M., HARMELING, S., AND SCHÖLKOPF, B. 2011. Non-stationary correction of optical aberrations. In *ICCV*.

SCHULER, C. J., BURGER, H. C., HARMELING, S., AND SCHÖLKOPF, B. 2013. A machine learning approach for non-blind image deconvolution. In *CVPR*.

SHAO, L., YAN, R., LI, X., AND LIU, Y. 2013. From heuristic optimization to dictionary learning: A review and comprehensive comparison of image denoising algorithms. *IEEE Transactions on Cybernetics*, 99.

STARCK, J.-L., MURTAGH, F., AND BIJAOUI, A. 1998. *Image Processing and Data Analysis: The Multiscale Approach*. Cambridge University Press, Cambridge.

TICO, M., AND PULLI, K. 2009. Image enhancement method via blur and noisy image fusion. In *ICIP*.

TICO, M. 2008. Multiframe image denoising and stabilization. In *European Signal Processing Conference*.

TSAI, Y.-T., STEINBERGER, M., PAJĄK, D., AND PULLI, K. 2014. Fast ANN for high-quality collaborative filtering. In *High-Performance Graphics*.

VENKATAKRISHNAN, S. V., BOUMAN, C. A., AND WOHLBERG, B. 2013. Plug-and-play priors for model based reconstruction. In *IEEE GlobalSIP*.

VENKATARAMAN, K., LELESCU, D., DUPARRÉ, J., MCMAHON, A., MOLINA, G., CHATTERJEE, P., MULLIS, R., AND NAYAR, S. 2013. Picam: an ultra-thin high performance monolithic camera array. *ACM TOG 32*, 6.

WALLACE, G. K. 1991. The JPEG still picture compression standard. *Communications of the ACM 34*, 4.

WILBURN, B., JOSHI, N., VAISH, V., TALVALA, E.-V., ANTUNEZ, E., BARTH, A., ADAMS, A., HOROWITZ, M., AND LEVOY, M. 2005. High performance imaging using large camera arrays. *ACM TOG 24*, 3.

WU, X., AND ZHANG, D. 2006. Improvement of color video demosaicking in temporal domain. *IEEE TIP 15*, 10.

XU, L., AND JIA, J. 2010. Two-phase kernel estimation for robust motion deblurring. In *ECCV*.

ZHANG, L., WU, X., BUADES, A., AND LI, X. 2011. Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *Journal of Electronic Imaging 20*, 2.

ZORAN, D., AND WEISS, Y. 2011. From learning models of natural image patches to whole image restoration. In *ICCV*.