Supplemental Material: Efficient Image Optimization using Proximal Algorithms

1 Mathematical Framework

1.1 Generalized Objective

We model an image optimization problem in our framework as a sum of penalties f_i on linear transforms $\mathbf{K}_i\mathbf{x}$ with $\mathbf{x} \in \mathbb{R}^n$ being the unknown:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{I} f_i\left(\mathbf{K}_i\mathbf{x}\right) \quad \text{with} \quad \mathbf{K} = \begin{bmatrix}\mathbf{K}_1 \\ \vdots \\ \mathbf{K}_I\end{bmatrix}, \tag{1}$$

where here $\mathbf{K} \in \mathbb{R}^{m\times n}$ is one large matrix that is composed of stacked linear operators $\mathbf{K}_1, \ldots, \mathbf{K}_I$. The linear operator $\mathbf{K}_i \in \mathbb{R}^{m_i\times n}$ selects a subset of m_i rows of $\mathbf{K}\mathbf{x}$. This subset of rows is then the input for the penalty functions $f_i : \mathbb{R}^{m_i} \to \mathbb{R}$.

The only assumption we make about the penalty functions $f_1, \ldots, f_I$ is that they provide a black box for evaluating the function's proximal operator. The proximal operator of a function f is defined as

$$\mathbf{prox}_{\tau f}(\mathbf{v}) = \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) + \frac{1}{2\tau}\|\mathbf{x}-\mathbf{v}\|_2^2\right),$$

where $\tau > 0$ and $\mathbf{v} \in \mathbb{R}^{m_i}$ [Parikh and Boyd 2013]. The proximal operator optimizes over the function in isolation, but incorporates a quadratic term that can be used to link the optimization with a broader algorithm. Proximal operators are an alternative to the traditional gradient and Hessian oracles used by first and second order methods.

Similarly, the only assumption we make about each linear operator $\mathbf{K}_i$ is that it provides a black box for evaluating the forward operator $\mathbf{x} \to \mathbf{K}_i\mathbf{x}$ and the adjoint operator $\mathbf{z} \to \mathbf{K}_i^T\mathbf{z}$. This is a useful abstraction because many linear operators that arise in optimization problems from image processing are fast transforms, *i.e.*, they have methods for evaluating the forward and adjoint operator that are more efficient than standard multiplication by the operator represented as a dense or sparse matrix. Common fast transforms in image processing include the discrete Fourier transform (DFT), convolution, and wavelet transforms; see [Diamond and Boyd 2016] for many more examples.

1.1.1 Compiling Problems to Efficient Solvers

We can solve Problem (1) using a variety of different existing proximal algorithms. ProxImaL supports the Chambolle-Pock algorithm [Chambolle and Pock 2011; Pock et al. 2009] ADMM [Boyd et al. 2001], linearized ADMM [Parikh and Boyd 2013, §4.2.2], and half-quadratic splitting [Geman and Yang 1995]. Each of these algorithms splits the objective into a sum of two functions g and h.

As discussed in the paper, the ProxImaL compiler splits problem (1) into g and h via

$$g(\mathbf{x}) = \sum_{f_i\in\Omega} f_i(\mathbf{x}), \qquad h(\mathbf{z}) = \sum_{f_i\in\Psi} f_i(\mathbf{z}),$$

where Ω and Ψ are a partition of the set of functions $\{f_1, \ldots, f_I\}$.

In the paper we discussed our implementation of the Chambolle-Pock and ADMM algorithms in detail. In the supplement we explain our implementation of the linearized ADMM and half-quadratic splitting algorithms, and discuss the convergence properties of all four algorithms.

Linearized ADMM implementation. The pseudo-code for linearized ADMM is given in Algorithm 1. Our compiler uses the default hyper-parameters $\rho = 1/\|\mathbf{K}\|_2$, $\mu = \|\mathbf{K}\|_2$, $\mathbf{x}^0 = 0$, $\mathbf{z}^0 = 0$, and $\lambda^0 = 0$. With the default problem scaling, we have $\|\mathbf{K}\|_2 = 1$. In linearized ADMM the terms f_i of $g = \sum_{i\in\Omega} f_i$ can be any function with an efficient proximal operator. The default splitting into Ω and Ψ is to include all quadratic functions in Ω.

Algorithm 1 Linearized ADMM to solve Problem (1)

1: Initialization: $\mu > \rho\|\mathbf{K}\|_2^2$, $(\mathbf{x}^0, \mathbf{z}^0, \lambda^0)$.
2: **for** $k = 1$ to V **do**
3: $\quad \mathbf{x}^{k+1} = \mathbf{prox}_{\frac{g}{\mu}}(\mathbf{x}^k - (\rho/\mu)K^T(K\mathbf{x}^k - \mathbf{z}^k + \lambda^k))$
4:
5: $\quad \mathbf{z}_j^{k+1} = \mathbf{prox}_{\frac{f_j}{\rho}}(\mathbf{K}_j\mathbf{x}_j^{k+1} + \lambda_j^k) \quad \forall j \in \Psi$
6: $\quad \lambda_j^{k+1} = \lambda_j^k + (\mathbf{K}_j\mathbf{x}^{k+1} - \mathbf{z}_j^{k+1}) \quad \forall j \in \Psi$
7: **end for**

Linearized ADMM is a natural variant of standard ADMM. Recall that ADMM applied to Problem (1) solves the optimization problem

$$\text{minimize} \quad g(\mathbf{x}) + (\rho/2)\|\mathbf{K}\mathbf{x} - \mathbf{z}^k + \lambda^k\|_2^2$$

as a subroutine. In linearized ADMM we replace the term

$$(\rho/2)\|\mathbf{K}\mathbf{x} - \mathbf{z}^k + \lambda^k\|_2^2$$

with its linearization plus quadratic regularization:

$$\rho\mathbf{K}^T(\mathbf{K}\mathbf{x} - \mathbf{z}^k + \lambda^k) + (\mu/2)\|v - v^k\|_2^2.$$

The Pock-Chambolle algorithm is in fact linearized ADMM applied to the dual of Problem (1) [Chambolle and Pock 2011]. Pock-Chambolle could therefore be implemented in ProxImaL as a symbolic transformation that converts the problem representation into its dual, combined with the linearized ADMM implementation. The dual problem involves the conjugates of $f_1, \ldots, f_I$ from Problem (1) (see Problem (3)). The proximal operator for the conjugate f_i^* can be evaluated using the proximal operator for f_i and Moreau's Identity [Moreau 1965].

Half-quadratic splitting implementation. The pseudo-code for half-quadratic splitting is given in Algorithm 2. Our compiler uses the default hyper-parameters $\rho^0 = 1$, $\alpha = 2$, $\rho_{\max} = 2^8$, $\mathbf{x}^0 = 0$, and $\mathbf{z}^0 = 0$. Our compiler only allows quadratic functions to be included in Ω because then step 3 reduces to solving a least squares problem. The default splitting into Ω and Ψ is the same as for ADMM.

On the first glance, the half-quadratic splitting (HQS) implementation looks similar to our ADMM implementation. However, the key difference is the Lagrange multipliers in ADMM have been

Algorithm 2 Half-Quadratic Splitting to solve Problem (1)

1: Initialization: $\rho^0 > 0$, $\rho_{\max} > 0$, $\alpha > 1$, $(\mathbf{x}^0, \mathbf{z}^0)$.
2: **for** $k = 1$ to V **do**
3: $\quad \mathbf{x}^{k+1} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i\in\Omega} f_i(\mathbf{K}_i\mathbf{x}) + \rho^k \sum_{j\in\Psi} \|\mathbf{K}_j\mathbf{x} - \mathbf{z}_j\|_2^2$
4: $\quad \mathbf{z}_j^{k+1} = \mathbf{prox}_{\frac{f_j}{\rho^k}}(\mathbf{K}_j\mathbf{x}_j^{k+1}) \quad \forall j \in \Psi$
5: $\quad \rho^{k+1} = \max\{\rho^k * \alpha, \rho_{\max}\}$
6: **end for**

eliminated in HQS. The use of the Lagrange multipliers in ADMM allows to have a fixed ρ. In HQS, by contrast, we need to scale $\rho \to \infty$. This scaling can cause both the quadratic step in line 3 and the proximal operator step in line 4 to be unstable. Thus, for the HQS method to be stable, it is crucial to minimize the number of splitting variables and find numerically accurate solutions to both steps (which is not necessary for ADMM).

1.2 Convergence properties

In this section we discuss the conditions under which the solver algorithms in ProxImaL converge. We assume that problem (1) has a global minimizer and observes certain regularity conditions. Formally, we assume that the problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^I f_i(\mathbf{z}_i) \\ \text{subject to} & \mathbf{K}_i\mathbf{x} = \mathbf{z}_i, \quad i = 1, \ldots, I, \end{array} \tag{2}$$

is feasible and bounded below, with optimal value equal to the optimal value of its Lagrange dual

$$\begin{array}{ll} \text{maximize} & -\sum_{i=1}^I f_i^*(\lambda_i) \\ \text{subject to} & \mathbf{K}_i^T\lambda_i = 0, \quad i = 1, \ldots, I. \end{array} \tag{3}$$

Under our assumption on problem (1), our implementation of Chambolle-Pock is guaranteed to converge to a global minimizer of problem (1) if the functions $f_1, \ldots, f_I$ are proper, convex, and lower semicontinuous. The hyper-parameters τ and σ must be chosen such that $\sigma\tau\|\mathbf{K}\|_2^2 < 1$ [Chambolle and Pock 2011].

Our implementation of ADMM is guaranteed to converge to a global minimizer of problem (1) if the functions $f_1, \ldots, f_I$ are closed, proper, and convex. Any choice of $\rho > 0$ is valid [Boyd et al. 2001]. Variations of ADMM where we use a relaxation parameter $\alpha \in (0, 2)$ or compute the variable updates inexactly still guarantee convergence to a global minimizer under reasonable conditions [Eckstein and Bertsekas 1992].

Our implementation of linearized ADMM is guaranteed to converge to a global minimizer of problem (1) if the functions $f_1, \ldots, f_I$ are closed, proper, and convex. The hyper-parameters ρ and μ must be chosen such that $\rho > 0$ and $\mu > \rho\|\mathbf{K}\|_2^2$ [Esser et al. 2010].

The convergence analysis for our implementation of half-quadratic splitting is more complex than for the other algorithms in ProxImaL . When $\Omega = \{f_1, \ldots, f_I\}$, and the $\mathbf{x}$-update is computed exactly, convergence to a global minimizer is guaranteed for $\rho_{\max} = \infty$ [Nocedal and Wright 2006, Chap. 17]. If the functions $f_1, \ldots, f_I$ are closed, proper, and convex, and $\mathbf{K}_1, \ldots, \mathbf{K}_I$ are full rank, then for constant ρ^k (*i.e.*, $\alpha = 1$), our implementation converges to a global minimizer of the problem

$$\text{minimize} \quad \sum_{i=1}^I f_i(\mathbf{z}_i) + \rho^k\|\mathbf{K}_i\mathbf{x} - \mathbf{z}_i\|_2^2,$$

where the equality constraints have been relaxed [Beck 2015]. In the fully general scenario where $f_1, \ldots, f_I$ are convex but $\Psi \neq \emptyset$ and ρ^k increases, our implementation is a heuristic, but one used successfully in applications, and strongly motivated by theoretical analysis [Wang et al. 2008].

We have only discussed convergence when problem (1) is convex. Recent work has shown, however, that for nonconvex problems under certain conditions Chambolle-Pock, ADMM, linearized ADMM, and variants of half-quadratic splitting are guaranteed to converge to a stationary point [Attouch et al. 2011; Möllenhoff et al. 2015; Li and Pong 2015; Robini and Zhu 2015].

ProxImaL does not force the user to obey the restrictions on the functions $f_1, \ldots, f_I$ and the algorithm hyper-parameters needed to guarantee convergence. Many state-of-the-art results in image optimization involve applying proximal algorithms to nonconvex problems for which it is difficult to provide guarantees about convergence (*e.g.*, [Krishnan and Fergus 2009; Heide et al. 2014]).

1.3 Stopping criteria

There are many possible ways to determine when the solver algorithms have converged, and different choices may work better for different problems. A reasonable, general purpose stopping criteria is that proposed in [Boyd et al. 2011], in which the algorithm terminates when the norms of the primal residual $r^{k+1} = \mathbf{K}\mathbf{x}^{k+1} - z^{k+1}$ and dual residual $s^{k+1} = \rho\mathbf{K}^T(\mathbf{z}^{k+1} - \mathbf{z}^k)$ fall below certain threshholds. The threshhold for the norm of the primal residual is given by

$$\epsilon^{\text{pri}} = \sqrt{m}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\{\|\mathbf{K}\mathbf{x}^k\|_2, \|\mathbf{z}^k\|_2\},$$

where ϵ^{abs} and ϵ^{rel} are chosen by the user. By default ProxImaL uses $\epsilon^{\text{abs}} = \epsilon^{\text{rel}} = 10^{-3}$. Similarly, the threshhold for the norm of the dual residual is given by

$$\epsilon^{\text{pri}} = \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}}\|\mathbf{K}^T\lambda^k\|_2.$$

This criterion is used in our framework for ADMM, LADMM and PC. Note that all of these methods are variants of the ADMM method [Chambolle and Pock 2011]. HQS suffers from instabilities for $\rho \to \infty$. Hence, in practice often a stopping criterion is used that limits ρ, see [Krishnan and Fergus 2009]. In our implementation we use default value of $\rho_{\max} = 2^8$. To add robustness for small values of α we add a progress-based stopping criterion

$$\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2 + \|\mathbf{z}_j^{k+1} - \mathbf{z}_j^k\|_2 < n\epsilon^{\text{hq}},$$

where our default value for ϵ^{hq} is $1e-6$. Finally, we allow the user to change all parameters of the stopping criteria for each of the implemented methods. In practice often an estimate of modest accuracy, but low computational cost, might be desired.

1.4 Example problem

As an example illustrating our theoretical framework, additional to the one in the main draft, we consider deconvolution with cross-channel prior [Heide et al. 2013]. The corresponding objective can be formulated as:

$$\begin{aligned} \mathbf{v}_{\text{opt}} = \underset{\mathbf{x}}{\operatorname{argmin}} \; & \|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2 + \mu \sum_a \|\mathbf{H}_a\mathbf{x}\|_1 + \\ & \gamma \sum_a \|\mathbf{H}_a\mathbf{x} - \mathbf{H}_a\mathbf{b}_r\|_1, \end{aligned} \tag{4}$$

We can now frame our problem from Eq. (4) in the more general form from Eq. (1):

$$\begin{aligned}\mathbf{x}_{\text{opt}} &= \underset{\mathbf{x}}{\text{argmin}}\ \|\mathbf{D}\mathbf{x}-\mathbf{b}\|_2^2 + \mu\sum_a \|\mathbf{H}_a\mathbf{x}\|_1 + \\ &\qquad \gamma\sum_a \|\mathbf{H}_a\mathbf{x}-\mathbf{H}_a\mathbf{b}_r\|_1 \\ &= \underset{\mathbf{x}}{\text{argmin}}\ f_1(\mathbf{D}\mathbf{x}) + \sum_{a=1}^{2} f_{(i+1)}(\mathbf{H}_a\mathbf{x}) + f_{(i+3)}(\mathbf{H}_a\mathbf{x}) \\ &= \underset{\mathbf{x}}{\text{argmin}}\ \sum_{i=1}^{5} f_i\left(\mathbf{K}(i)\mathbf{x}\right), \quad \text{with} \quad \mathbf{K}=\begin{bmatrix}\mathbf{D}\\ \mathbf{H}_1\\ \mathbf{H}_2\end{bmatrix}, \quad \text{and} \\ f_1(\mathbf{v}) &= \|\mathbf{v}-\mathbf{b}\|_2^2, \quad f_{2,3}(\mathbf{v}) = \mu\|\mathbf{v}\|_1, \quad f_{4,5}(\mathbf{v}) = \|\mathbf{v}-\alpha\|_1\end{aligned} \tag{5}$$

Having formulated our problem as a sum of functions operating on the stacked matrix $\mathbf{K}$, we can formulate the proximal operators for the functions $f_{2\ldots5}$:

$$\begin{aligned}&\mathbf{prox}_{\theta\|\cdot\|_1}(\mathbf{v}) = \max\left(1-\frac{\theta\beta}{|\mathbf{v}|},0\right)\odot\mathbf{v} \qquad \textbf{Shrinkage} \\ &\mathbf{prox}_{\theta\|\ \cdot\ -\alpha\|_1}(\mathbf{v}) = \underset{\mathbf{x}}{\text{argmin}}\ \theta\|\mathbf{x}-\alpha\|_1 + \frac{1}{2}\|\mathbf{x}-\mathbf{v}\|_2^2 \\ &= \alpha + \underset{\mathbf{z}}{\text{argmin}}\ \theta\|\mathbf{z}\|_1 + \frac{1}{2}\|\mathbf{z}+\alpha-\mathbf{v}\|_2^2 \quad \text{with} \quad \mathbf{z}:=\mathbf{x}-\alpha \\ &= \alpha + \underset{\mathbf{z}}{\text{argmin}}\ \theta\|\mathbf{z}\|_1 + \frac{1}{2}\|\mathbf{z}-(\mathbf{v}-\alpha)\|_2^2 \\ &= \mathbf{prox}_{\theta\|\cdot\|_1}(\mathbf{v}-\alpha)+\alpha \\ &= \max\left(1-\frac{\theta\beta}{|\mathbf{v}-\alpha|},0\right)\odot\mathbf{v}+\alpha \qquad \textbf{Cross-shrinkage}\end{aligned} \tag{6}$$

where here $\alpha = \mathbf{H}_a\mathbf{b}_r$. To derive the second proximal operator, we have used a simple substitution trick via substitution of $\mathbf{z} := \mathbf{x}-\alpha$. The proximal operator then reduces to the shrinkage operator.

Note that we give here the proximal operators for the primal functions. The proximal operators for their convex conjugates can be easily computed by using Moreau's identity [Moreau 1965].

1.4.1 Solving the example problem using ADMM

ADMM solves the joint minimization of the sum of all f_i terms into a sequence of separable minimizations w.r.t. f_i. The quadratic subproblem in line 3 of Alg. 2 from the main draft can be efficiently solved in the Fourier domain (due to our specific choice of Ω) as shown below in Eq. (7). To simplify notation we use the substitute variable $\omega_j := \mathbf{z}_j - \lambda_j^k$ here.

$$\begin{aligned}\mathbf{x}_{\text{opt}} &= \underset{\mathbf{x}}{\text{argmin}}\ \frac{1}{2}\sum_{i\in\{1\}} f_i(\mathbf{x}) + \frac{\rho}{2}\sum_{j\in\{2,3\}}\|\mathbf{K}_j\mathbf{x}-\omega_j\|_2^2 \\ &= \underset{\mathbf{x}}{\text{argmin}}\ \underbrace{\frac{1}{2}\|\mathbf{D}\mathbf{x}-\mathbf{b}\|_2^2 + \frac{\rho}{2}\|\mathbf{H}_1\mathbf{x}-\omega_2\|_2^2 + \frac{\rho}{2}\|\mathbf{H}_2\mathbf{x}-\omega_3\|_2^2}_{\Omega(\mathbf{x})} \\ \Leftrightarrow \frac{\partial\Phi(\Omega\mathbf{x})}{\partial\mathbf{x}} &= (\mathbf{D}^T\mathbf{D}+\rho\mathbf{H}_1^T\mathbf{H}_1+\rho\mathbf{H}_2^T\mathbf{H}_2)\mathbf{x}_{\text{opt}} - \\ &\qquad \mathbf{D}^T\mathbf{b}+\rho\mathbf{H}_1^T\omega_2+\rho\mathbf{H}_1^T\omega_3 \stackrel{!}{=} 0 \\ \Leftrightarrow (\mathbf{D}^T\mathbf{D}+\rho\sum_{a=1}^{2}\mathbf{H}_a^T\mathbf{H}_a)\mathbf{x}_{\text{opt}} &= \mathbf{D}^T\mathbf{b}+\rho\sum_{a=1}^{2}\mathbf{H}_a^T\omega_{a+1} \\ \Leftrightarrow \mathbf{x}_{\text{opt}} &= \boldsymbol{\mathcal{F}}^{-1}\left(\frac{\boldsymbol{\mathcal{F}}(\mathbf{D})^*\boldsymbol{\mathcal{F}}(\omega_1)+\rho\sum_{a=1}^{2}\boldsymbol{\mathcal{F}}(\mathbf{H}_a)^*\boldsymbol{\mathcal{F}}(\omega_{a+1})}{\boldsymbol{\mathcal{F}}(\mathbf{D})^*\boldsymbol{\mathcal{F}}(\mathbf{D})+\rho\sum_{a=1}^{2}\boldsymbol{\mathcal{F}}(\mathbf{H}_a)^*\boldsymbol{\mathcal{F}}(\mathbf{H}_a)}\right)\end{aligned} \tag{7}$$

The remaining parts Alg. 2 in the main draft are the proximal operators in line 4 and the Lagrange multiplier update in line 5. Previously, in Eq. (6), we have defined very efficient point-wise updates for the proximal operators of $\mathbf{prox}_{1/\rho f_\Psi}$. The lagrange multiplier update from line 5 is a point-wise operation as well, and therefore very efficiently solvable.

For this specific ADMM implementation, we note that the iterates are empirically converged in about 50 iterations. The primal-dual method from [Chambolle and Pock 2011] performs similar in this case. Note that the specific partition of the penalties and all substitutions apply directly for the derivation of our method using the method from [Chambolle and Pock 2011].

1.4.2 Solving the example problem using Half-Quadratic Splitting

The subproblem in line 3 of Alg. 2 can be solved in the Frequency domain analogously to the first step in the ADMM method from Eq. (7). The proximal operators are also same as in the ADMM case and given in Eq. (6).

Using Half-Quadratic Splitting, we observe that we can achieve almost converged results in 5 iterations, which means a *speedup of* $\times 60$ *compared to [Heide et al. 2013] and* $\times 10$ *compared to ADMM*. Note that accurately solving the proximal operators from Eq. (6) is key here (as approximations will lead to instabilities in enforcing the consensus constraint).

1.5 Example using Poisson Noise Fitting

A further example is extending the problem from Eq. (4) by a Poisson noise model. In this case, we model the observed image $\mathbf{b}$ as a sample of a random variable $\tilde{\mathbf{b}}$:

$$p(\tilde{\mathbf{b}}=\mathbf{b}\mid\lambda) = \prod_{i=1}^{n}\frac{\lambda_i^{\mathbf{b}_i}e^{-\lambda_i}}{\mathbf{b}_i!}, \tag{8}$$

where here the notation $(\cdot)_i$ denotes the selection of the i-th component of the image vector given as argument. Following the Bayesian maximum a posteriori criterion, which is also proposed in [Figueiredo and Bioucas-Dias 2009], the quadratic dataterm

(a) Sharp ground truth

(b) Blurred observation corrupted by Poisson noise (PSF in the top left)

(c) Deconvolved result using ℓ_2-optimization assuming Gaussian noise

(d) Reconstruction using our optimization assuming Poisson noise

Figure 1: *Effect of Poisson noise on deconvolution with large kernels: This example demonstrates that proper noise modeling leads to significant gains in reconstruction quality for large blur kernels. To only demonstrate the effect of the noise model, we did* not use cross-channel information *for this example. Since the Poisson distribution can be approximated well by a normal distribution for large values of the mean, the most significant improvements of using the proper noise model can be found in the low-intensity regions.*

$\|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2$ from Eq. (4) then becomes the negative log-likelihood of $p(\tilde{\mathbf{b}} = \mathbf{b} \mid \mathbf{x})$, that is

$$\begin{aligned} -\log\left(p(\tilde{\mathbf{b}} = \mathbf{b} \mid \mathbf{x})\right) &= \sum_{i=1}^{n} \Gamma\left((\mathbf{D}\mathbf{x})_i, \mathbf{b}_i\right) \quad \text{with} \\ \Gamma(a,b) &= a - b\log(a) + \mathrm{ind}_{\mathbb{R}^+}(a), \end{aligned} \tag{9}$$

where $\mathrm{ind}_{\mathbb{R}^+}(a)$ is the indicator function for the positive orthant. We can now easily bring this into our generalized objective form from 1 by setting

$$f_1(\mathbf{x}) = \mathbf{x} - \mathbf{b}\log(\mathbf{x}) + \mathrm{ind}_{\mathbb{R}^+}(\mathbf{x}) \tag{10}$$

The corresponding proximal operator for the changed f_1 is

$$\mathbf{prox}_{\theta f_1(\cdot)}(\mathbf{v}) = \frac{\mathbf{v} - \theta}{2} + \sqrt{\theta\mathbf{b} + \frac{(\theta - \mathbf{v})^2}{4}} \quad \begin{matrix}\textbf{Poisson}\\ \textbf{Penalty}\end{matrix} \tag{11}$$

This analytic solution for the proximal operator of f_1 results in a root-finding problem of a second-order polynomial as shown in Eq. (12). Due to the positivity constraint in f_1 the minimum is uniquely defined by the positive root:

$$\begin{aligned} \mathbf{prox}_{\theta f_1(\cdot)}(\mathbf{v}) &= \operatorname*{argmin}_{\mathbf{x}} \; f_1(\mathbf{x}) + \frac{1}{2\theta}\|\mathbf{x} - \mathbf{v}\|_2^2 \\ &= \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^+} \; \underbrace{\mathbf{x} - \mathbf{b}\log(\mathbf{x}) + \frac{1}{2\theta}\|\mathbf{x} - \mathbf{v}\|_2^2}_{\Upsilon(\mathbf{x})} \end{aligned}$$

$$\begin{aligned} &\Leftrightarrow \frac{\partial\Upsilon(\mathbf{x}_{\text{opt}})}{\partial\mathbf{x}} = \mathbf{1} - \frac{\mathbf{b}}{\mathbf{x}_{\text{opt}}} + \frac{1}{\theta}\mathbf{x}_{\text{opt}} - \frac{1}{\theta}\mathbf{v} \stackrel{!}{=} 0 \quad \text{s.t.} \quad \mathbf{x}_{\text{opt}} \in \mathbb{R}^+ \\ &\Leftrightarrow \mathbf{x}_{\text{opt}}^2 + (\theta - \mathbf{v}) \cdot \mathbf{x}_{\text{opt}} - \theta\mathbf{b} = 0 \quad \text{s.t.} \quad \mathbf{x}_{\text{opt}} \in \mathbb{R}^+ \\ &\Leftrightarrow \mathbf{x}_{\text{opt}} = \frac{\mathbf{v} - \theta}{2} + \sqrt{\theta\mathbf{b} + \frac{(\theta - \mathbf{v})^2}{4}} \end{aligned}$$

Having defined the changed f_1 and the corresponding proximal operator, we can directly apply our framework derived in Sec 1.1, which maps directly to the ADMM algorithm or Half-Quadratic Splitting method as described above. The only thing that changes is that the set Ψ is now empty (since f_1 is now no longer a simple quadratic as defined previously) and consequently $\Psi := \{1, 2, 3\}$. This means new auxiliary variables for $\mathbf{K}_i\mathbf{x}$ are introduced. The price, that we pay for being able to solve for Poisson degraded observations are now again more iterations (necessary to enforce the consensus constraints). However, the quality is quite significantly affected by the more proper noise model as demonstrated in Fig. 1. Note also that we cannot simply use a variance stabilization transform here (such as the Anscombe transform) since the observations are mixed together in the convolution with the kernel.

2 Additional Details for Implementation

In Table 3, we compare runtimes for various options of the ProxImaL compiler and compare them for the specific example of TV-regularized deconvolution. We show runtimes for an implementation with NumPy and for the corresponding Halide implementation as well as for the gradient operator implemented in the primal domain and via the convolution theorem in the Fourier domain. Further, Table 4 lists runtimes for a range of different linear operators and some of the simple proximal operators for implementations in NumPy and Halide.

2.1 Proximal operators

In this section we discuss the full set of proxable functions provided by ProxImaL . The inputs to proximal operators are real vectors, obtained by flattening the input into a vector. We provide closed form expressions for the proximal operators when possible and otherwise give references with the precise definition of the proximal operator.

Sum-squares. The `sum_squares(x)` function represents the squared ℓ_2-norm $f(x) = \|x\|_2^2$, where $x \in \mathbb{R}^n$. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v}) = \frac{1}{2\tau + 1}\mathbf{v}.$$

The `weighted_sum_squares(D,x)` function is a variant defined as $f(x) = \|Dx\|_2^2$, where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v}) = (2\tau D^T D + I)^{-1}\mathbf{v}.$$

The weighted variant is used to absorb diagonal linear operators into the `sum_squares` proxable function.

ℓ_1-norm. The `norm1(x)` function represents the ℓ_1-norm $f(x) = \|x\|_1$, where $x \in \mathbb{R}^n$. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v})_i = \mathrm{sign}(\mathbf{v}_i)\max\{|v_i| - \tau, 0\}, \quad i = 1, \ldots, n,$$

Spatial Grad	direct_off + split_off	direct_off + split_on	direct_on + split_off	direct_on + split_on
CP	30.2	40.8	32.5	13.9
ADMM	313.7	70.2	323.2	67.6
LADMM	53.6	32.2	53.8	11.6
HQS	2.6	**2.4**	2.6	**2.4**
Conv. Grad	direct_off + split_off	direct_off + split_on	direct_on + split_off	direct_on + split_on
CP	51.8	59.1	51.8	26.8
ADMM	899.3	215.8	103.1	25.9
LADMM	116.3	71.7	116.7	43.6
HQS	5.3	5.4	1.4	**1.2**

Figure 2: *Runtime of a TV-regularized deconvolution problem for NumPy implementation.*

Spatial Grad	direct_off + split_off	direct_off + split_on	direct_on + split_off	direct_on + split_on
CP	16.3	14.0	15.9	7.2
ADMM	126.8	29.8	125.9	29.6
LADMM	27.9	12.8	28.4	6.8
HQS	1.1	**1.0**	1.1	**1.0**
Conv. Grad	direct_off + split_off	direct_off + split_on	direct_on + split_off	direct_on + split_on
CP	15.6	17.0	15.4	8.8
ADMM	188.0	48.1	28.0	6.9
LADMM	34.1	19.8	31.9	11.7
HQS	1.1	1.1	0.5	**0.5**

Figure 3: *Runtime (in seconds) of a TV-regularized deconvolution problem for Halide implementation. We evaluate four different algorithms that are implemented in ProxImaL. When the* direct *parameter is on, the ProxImaL compiler automatically replaces CG with a fast direct method. The* split *option further indicates whether the compiler's intelligent rewriting and splitting are used. We evaluate two implementations of the TV prior: a finite differences implementation in the spatial domain (top table) and a convolutional implementation via Fourier multiplication (bottom table).*

	ℓ_2-norm	dot product	subsample	subsample*	grad	grad*	convolution
Halide	41.6	15.6	72.6	72.6	94.8	237.4	121.4
NumPy	245.8	96.6	356.0	356.0	1188.0	713.1	7790.9

	convolution*	warp	warp*	norm1	group norm1	poisson prox	FFT inversion
Halide	121.4	457.6	367.8	27.1	67.8	44.7	9.4
NumPy	7790.9	153.1	474.4	201.8	1036.6	265.2	23.4

Figure 4: *Runtimes (in ms) for linear operators and some of the proximal functions implemented with NumPy and Halide.*

also known as soft-thresholding. The `weighted_norm1(D,x)` function is a variant defined as $f(x) = \|Dx\|_1$, where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v})_i = \text{sign}(\mathbf{v}_i)\max\{|v_i| - \tau|D_{ii}|, 0\}, \quad i = 1, \ldots, n.$$

The weighted variant is used to absorb diagonal linear operators into the `norm1` proxable function.

Poisson norm. The `poisson_norm(x)` function represents the negative log-likelihood under a Poisson noise model, given by

$$f(x) = \sum_{i=1}^{n} x_i - b_i \log(x_i) + \mathcal{I}_{(0,+\infty)}(x_i),$$

where $b \in \mathbb{R}^n_+$, $x \in \mathbb{R}^n$, and $\mathcal{I}_{(0,+\infty)}$ is the indicator function on the interval $(0, +\infty)$. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v})_i = \frac{\mathbf{v}_i - \tau}{2} + \sqrt{\tau b_i + (\tau - \mathbf{v}_i)^2/4}, \quad i = 1, \ldots, n.$$

The `weighted_poisson_norm(D,x)` function is a variant defined as

$$f(x) = \sum_{i=1}^{n} D_{ii}x_i - b_i \log(D_{ii}x_i) + \mathcal{I}_{(0,+\infty)}(D_{ii}x_i),$$

where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v})_i = \frac{\mathbf{v}_i - \tau D_{ii}}{2} + \sqrt{\tau b_i + (\tau D_{ii} - \mathbf{v}_i)^2/4}, \quad i = 1, \ldots, n.$$

The weighted variant is used to absorb diagonal linear operators into the `poisson_norm` proxable function.

Nonnegativity constraint. The `nonneg(x)` function represents the indicator $f(x) = \sum_{i=1}^{n} \mathcal{I}_{[0,+\infty)}(x_i)$, where $x \in \mathbb{R}^n$. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v})_i = \max\{\mathbf{v}_i, 0\}, \quad i = 1, \ldots, n.$$

The `weighted_nonneg(D,x)` function is a variant defined as

$$f(x) = \sum_{i=1}^{n} \mathcal{I}_{[0,+\infty)}(D_{ii}x_i),$$

where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v})_i = \max\{D_{ii}\mathbf{v}_i, 0\}/D_{ii}, \quad i = 1, \ldots, n.$$

The weighted variant is used to absorb diagonal linear operators into the `nonneg` proxable function.

Group ℓ_1-norm. The `group_norm1(x, dims)` function represents the sum of ℓ_2-norms $f(x) = \sum_{i=1}^{p} \|x_{g_i}\|_2$, where $x \in \mathbb{R}^n$ and $g_1, \ldots, g_p$ is a partition of $\{1, \ldots, n\}$ obtained by flattening x along the chosen dimensions. The proximal operator is given by

$$\mathbf{prox}_{\tau f}(\mathbf{v})_{g_i} = \mathbf{v}_{g_i} \max\{1 - \tau/\|\mathbf{v}_{g_i}\|_2, 0\}, \quad i = 1, \ldots, p,$$

also known as group soft-thresholding.

Denoising. Due to the quadratic proximity term, a proximal operator can also be interpreted as a Maximum a Posteriori (MAP) estimate for a Gaussian likelihood, i.e. a Gaussian denoiser, [Heide et al. 2014]. For a Gaussian likelihood $p(\mathbf{x}|\mathbf{v})$ and an arbitrary exponential prior $p(\mathbf{x})$

$$p(\mathbf{x}|\mathbf{v}) \propto \exp\left(-\frac{\|\mathbf{x} - \mathbf{v}\|_2^2}{2\sigma^2}\right)$$
$$p(\mathbf{x}) \propto \exp\left(-\Gamma(\mathbf{x})\right),$$

we get the following MAP estimate, that is a proximal operator

$$\mathbf{prox}_{\sigma^2\Gamma}(\mathbf{v}) = \underset{\mathbf{x}}{\text{argmin}} \left(\Gamma(\mathbf{x}) + \frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{v}\|_2^2\right)$$

Consider a family of Gaussian denoising algorithms $\{D^\Gamma_\sigma : \sigma > 0\}$ using the prior from above, which estimate $\mathbf{x}_0$ from $\mathbf{x}_0 + \sigma\mathbf{z}$ with $\mathbf{z} \sim \mathcal{N}(0, \mathbb{I})$. We can then formulate

$$\mathbf{prox}_{\tau\Gamma}(\mathbf{v}) = D^\Gamma_{\sqrt{\tau}}(\mathbf{v}),$$

Note, that it is not necessary to derive Γ in order to evaluate the proximal operator, only D^Γ_σ has to be known. Note that depending on Γ this operator can be non-convex. If Γ is the indicator function of a convex set, $D^\Gamma_{\sqrt{\tau}}$ becomes the projection onto this set. The (non-convex) proximal operatrs `patch_NLM` and `patch_BM3D` prior implement the non-local means and BM3D denoiser with algorithm with adapted weights.

2.2 Linear operators

In this section we discuss the full set of linear operators provided by ProxImaL . The inputs to and outputs from linear operators are arbitrary n-dimensional real arrays. We note when an operator or its Gram matrix is diagonal in the spatial or frequency domain.

Convolution. The `conv(k,x)` operator represents circular convolution of an n-dimensional array x with a known n-dimensional kernel k. The kernel k is automatically zero-padded to be the same size as x. The adjoint is circular convolution with the kernel $\tilde{k}$ given by

$$\tilde{k}_{i_1,\ldots,i_n} = k_{-i_1,\ldots,-i_n},$$

where k's indexing is zero-based and periodic. Non-circular boundary conditions are imposed through composition with additional linear operators, such as `mul_elemwise` to zero-pad x. The `conv` is diagonal in the frequency domain.

Subsample. The `subsample(x, steps)` operator extracts every steps_i entry of x along axis i, starting with the entry $\text{steps}_i - 1$. The adjoint inserts steps_i before each entry in axis i. The `subsample` operator is Gram diagonal.

Element-wise multiplication. The `mul_elemwise(w, x)` operator represents element-wise multiplication of x with a fixed array w. The adjoint also represents element-wise multiplication with w. The `mul_elemwise` operator is diagonal.

Scalar multiplication. The `scale(c,x)` operator represents multiplication by a fixed constant scalar $c \in \mathbb{R}$. The adjoint is also multiplication by c. The `scale` operator is diagonal.

Sum. The `sum(x1, x2, ..., xk)` operator represents summing k arrays $x_1, \ldots, x_k$. The adjoint copies a single input into k different outputs.

Vstack. The `vstack(x1, x2, ..., xk)` operator represents vertically stacking k arrays $x_1, \ldots, x_k$ into a single array. The adjoint splits a single array into k different subarrays. The `vstack` operator is diagonal.

Gradient. The `grad(x,[dims,periodic])` operator represents the discrete gradient of x across the chosen dimensions. By default the gradient is computed across all dimensions. The adjoint computes the negative divergence across the chosen dimensions. The `grad` operator is diagonal in the frequency domain if periodic boundary conditions are enabled. By default boundary conditions are non-periodic.

Warp. `warp(e, H)` operator represents Interprets `e` as a 2D image and warps it using the homography `H`. We use a bilinear resampling function. Note that the warps are not limited to homography transformation, but arbitrary pixel transforms can be supported.

Pixel-wise Transforms. `mul_color(e, C)` operator represents Performs a blockwise 3×3 color transform using the color matrix `C`, or the predefined opponent (`opp`) and YUV (`yuv`) color spaces. The `mul_color` operator is block diagonal.

Reshape. The `reshape(x, dims)` operator reinterprets x as an array with the given dimensions (*e.g.*, flattening a matrix to a vector). The adjoint reinterprets an array with the given dimensions as an array with the original dimensions of x. The `reshape` operator is diagonal.

2.3 Adding new operators

Adding additional proximal and linear operators is straightforward. The following code defines a new proxable function. The function is defined as a class inheriting from the base class **`ProxFn`**.

```
class new_func(ProxFn):
    def __init__(...):
        # Custom initialization code.

    def prox(self, tau, v):
        # Code to compute the function's proximal operator.
        return ...

    def eval(self, v):
        # Code to evaluate the function.
        return ...
```

The `eval` method may be omitted if the function cannot be explicitly computed, as with the **`patch_NLM`** function.

The following code defines a new linear operator. The operator is defined as a class inheriting from the base class **`LinOp`**.

```
class new_linop(LinOp):
    def __init__(...):
      # Custom initialization code.

    def forward(self, inputs, outputs):
        # Read from inputs, apply operator, and write to outputs.

    def adjoint(self, inputs, outputs):
        # Read from inputs, apply adjoint, and write to outputs.
```

The class should also define the methods `is_diag`, `is_gram_diag`, and `get_diag` if the linear operator K or its Gram matrix $K^T K$ is diagonal, so that the ProxImaL compiler can exploit this fact. Similarly, the class should define the methods `is_fdiag`, `is_gram_fdiag`, and `get_fdiag` if the linear operator or its Gram matrix is diagonal in the frequency domain. If the methods are not defined, the default implementations are used, which assume that neither the operator nor its Gram matrix are diagonalizable in the spatial or frequency domain.

3 Additional Details for Results

In this section, we detail additional results that were reported in PNSR tables in the primary text and we also show larger and extended versions of other figures.

Demosaicking Table 5 reports individual PSNR measurements that were only reported in an aggregated form in the primary text. We list individual PSNRs and also the average PSNR for all 18 test images. Figures 6 and 7 show each of the test images. In particular, we show the target image, the color-coded mosaick, the reconstruction achieved by FlexISP, its absolute error, the ProxImaL reconstruction, and its absolute error. Qualitatively, both FlexISP and ProxImaL achieve almost indistinguishable results.

Burst Denoising We show the full sensor image of one of the noisy frames for each of the burst denoising examples in Figure 8. The magnified regions that are also shown in the text are highlighted and shown in addition to the noisy frames. ProxImaL achieves an image quality comparable to the commercial reference software. Figure 9 shows simulations of another dataset, were we compare several different image priors.

Poisson Deconvolution In addition to the averaged PSNR values for the test dataset shown in the primary text, we show individual PSNR values for each example image in Table 10. We compare the 12 test images shown in Figures 11 and 12 for 5 different blur kernels and for three different reconstruction methods. On average ProxImaL achieves the best results.

Image	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Avg
FlexISP, PSNR R	30.66	35.85	36.16	**41.34**	38.24	**41.07**	40.01	39.84	38.78	**40.01**	41.30	41.28	43.99	40.46	38.31	35.62	34.72	35.41	38.50
FlexISP, PSNR G	33.75	40.43	38.60	**44.68**	40.13	45.31	41.90	44.09	43.90	44.08	43.67	43.85	46.40	44.13	43.52	36.37	40.20	38.07	41.84
FlexISP, PSNR B	**28.50**	33.63	32.78	37.07	32.38	**37.93**	37.96	39.40	38.36	38.32	39.83	39.03	39.20	36.86	39.86	35.07	33.76	**37.18**	36.50
ProxImaL, PSNR R	**30.73**	**35.87**	**36.19**	41.32	38.24	41.02	**40.24**	**40.03**	**38.82**	40.00	**41.33**	**41.29**	**44.04**	**40.50**	**38.33**	35.62	**34.77**	**35.43**	**38.54**
ProxImaL, PSNR G	**33.80**	**40.50**	**38.64**	44.65	**40.20**	**45.38**	**42.14**	**44.14**	**43.97**	**44.12**	**43.72**	**43.92**	**46.49**	**44.20**	**43.55**	**36.41**	**40.23**	**38.13**	**41.90**
ProxImaL, PSNR B	28.49	**33.65**	**32.82**	37.07	**32.40**	37.89	**38.19**	**39.42**	**38.38**	**38.35**	**39.87**	**39.05**	**39.23**	**36.88**	**39.87**	**35.10**	**33.79**	37.12	**36.53**

Figure 5: *Peak signal-to-noise-ratios for demosaicking. We report PSNR values in dB for each color channel of the 18 images and compare the quality achieved with FlexISP and ProxImaL. In the primary text, we only reported the average values show in the right-most column.*

4 Additional Details for Phase Retrieval

In this section, we provide additional detail on the phase retrieval problem discussed in the main paper. We first review phase retrieval and then explain the ADMM method corresponding to the ProxImaL code in the application section. The results of this method are the ones shown in the primary text.

4.1 Background and Algorithms

Phase retrieval attempts to recover a real or complex signal, given observations of the amplitude of linear measurements. We focus on the case where the linear operator is the Fourier transform operator, *i.e.*, solve

$$\begin{array}{ll} \text{find} & \mathbf{x} \\ \text{subject to} & \mathbf{b} = |\mathcal{F}\mathbf{x}| \end{array} \tag{12}$$

where $\mathbf{b} \in \mathbb{R}^n$ is the amplitude measurement of a Fourier transformed target signal, $\mathbf{x} \in \mathbb{R}^n$ is this target signal, and $\mathcal{F} \in \mathbb{R}^{n \times n}$ is the discrete Fourier transform operator. This problem arises in various applications like X-ray crystallography, coherent diffraction imaging and phase microscopy, where we can only measure the intensity of Fourier transformed signal without the phase.

The phase retrieval problem is non-convex due to the phase constraint $\mathbf{b} = |\mathcal{F}\mathbf{x}|$. The constraint is underdeterimined, since for any diagonal matrix $\mathbf{D} \in \mathbb{C}^{n \times n}$ whose diagonal entries have unit magnitude, $\mathcal{F}^{-1}(Db)$ satisfies the constraint. Therefore, extra constraints or priors are necessary to recover the desired signal.

A common approach is to assume that the signal is non-negative with finite support D. One class of algorithms for phase retrieval alternates between projection onto the constraint that $\mathbf{x} \geq 0$ with support D and projection onto the phase constraint [Gerchberg 1972; Fienup 1982; Bauschke et al. 2002; Bauschke et al. 2003; Elser 2003; Luke 2005]. The projection P_s onto the non-negative support constraint is given by

$$(P_s\mathbf{x})_i = \begin{cases} \max\{0,\ \mathbf{x}_i\} & \text{if } i \in D \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The projection P_m onto the phase constraint is given by

$$P_m\mathbf{x} = \mathcal{F}^{-1}(\mathbf{v}(\mathbf{x})),$$

$$\text{where} \quad (\mathbf{v}(\mathbf{x}))_i = \begin{cases} \mathbf{b}_i \dfrac{(\mathcal{F}\mathbf{x})_i}{|(\mathcal{F}\mathbf{x})_i|} & \text{if } (\mathcal{F}\mathbf{x})_i \neq 0 \\ \mathbf{b}_i & \text{otherwise} \end{cases} \tag{14}$$

The hybrid input-output (HIO) algorithm is the most popular and widely used of the algorithms based on alternating projections [Fienup 1982]. The algorithm is given by the update

$$\mathbf{x}_i^{n+1} = \begin{cases} (P_sP_m\mathbf{x}^n)_i & \text{if } i \in D \\ \mathbf{x}_i^n - \beta(P_sP_m\mathbf{x}^n)_i & \text{otherwise} \end{cases} \tag{15}$$

HIO improves on simple alternating projections by including a feedback parameter β, which reduces the chance of being trapped at a local minimum.

Usually the support D is unknown, in which case algorithms like HIO must be augmented with a method for estimating the support. Given a Fourier measurement, the autocorrelation function, also known as the Patterson function, provides an initial guess

$$P(\mathbf{x}) = \mathcal{F}^{-1}(|\mathbf{x}|^2) \tag{16}$$

Binarizing the output of the autocorrelation (the threshold is usually 4%) gives a sufficiently accurate estimate of the support.

Often the initial guess is updated as the algorithm progresses. There are two common approaches to updating the support, both based on the Shrinkwrap algorithm [Marchesini et al. 2003]. The two approaches differ in the threshold used to estimate the support. One performs thresholding on the current estimate with a fixed threshold, while the other picks the threshold such that a fixed area is included in the support. The original Shrinkwrap algorithm is defined as follows.

Let $\mathbf{x}$ be the input image, $G(k_r)$ a normalized Gaussian blurring kernel with a user-defined standard deviation k_r, and η the threshold to apply. The support D is then computed by

1. $\mathbf{x_g} = |\mathbf{x}| * G(k_r)$
2. $c = \eta \max(\mathbf{x_g})$
3. $D_i = \begin{cases} 1 & \text{if } \mathbf{x}_{\mathbf{g}_i} \geq c \\ 0 & \text{if } \mathbf{x}_{\mathbf{g}_i} < c \end{cases}$

where $D = 1$ defines the region inside the support and $D = 0$ defines the region outside the support.

4.2 Our approach

HIO with Shrinkwrap gives a reasonable estimate of the true image, but artifacts are common in the reconstruction [Osherovich 2012; Fannjiang and Liao 2012]. We instead approach phase retrieval by solving the TV regularized reconstruction problem

$$\underset{\mathbf{x}}{\text{argmin}} \quad \frac{1}{2}\||\mathcal{F}\mathbf{x}| - \mathbf{b}\|_2^2 + \lambda\|\|\nabla\mathbf{x}\|_1 + I_{D\cap[0,+\infty)}(\mathbf{x}) \tag{17}$$

where ∇ is the gradient operator. The indicator function adds the support constraint and a non-negativity constraint. We solve the problem using the ADMM implementation in ProxImaL . The problem is nonconvex, so we are not guaranteed to find a global optimum. We use the output of HIO as an initial point $\mathbf{x}^0$. A fixed support estimate from this input was sufficient in our tests.

The $\mathbf{x}$-update in ADMM requires approximately solving a nonlinear least-squares problem. In particular, we must minimize the function

$$f(\mathbf{x}) = \frac{1}{2}\||\mathcal{F}\mathbf{x}| - \mathbf{b}\|_2^2 + \frac{\rho_1}{2}\|\nabla\mathbf{x} - \mathbf{v_1}\|_2^2 + \frac{\rho_2}{2}\|\mathbf{x} - \mathbf{v_2}\|_2^2 \tag{18}$$

for given $\rho_1, \rho_2, \mathbf{v_1}, \mathbf{v_2}$. We minimize $f(\mathbf{x})$ using L-BFGS. The

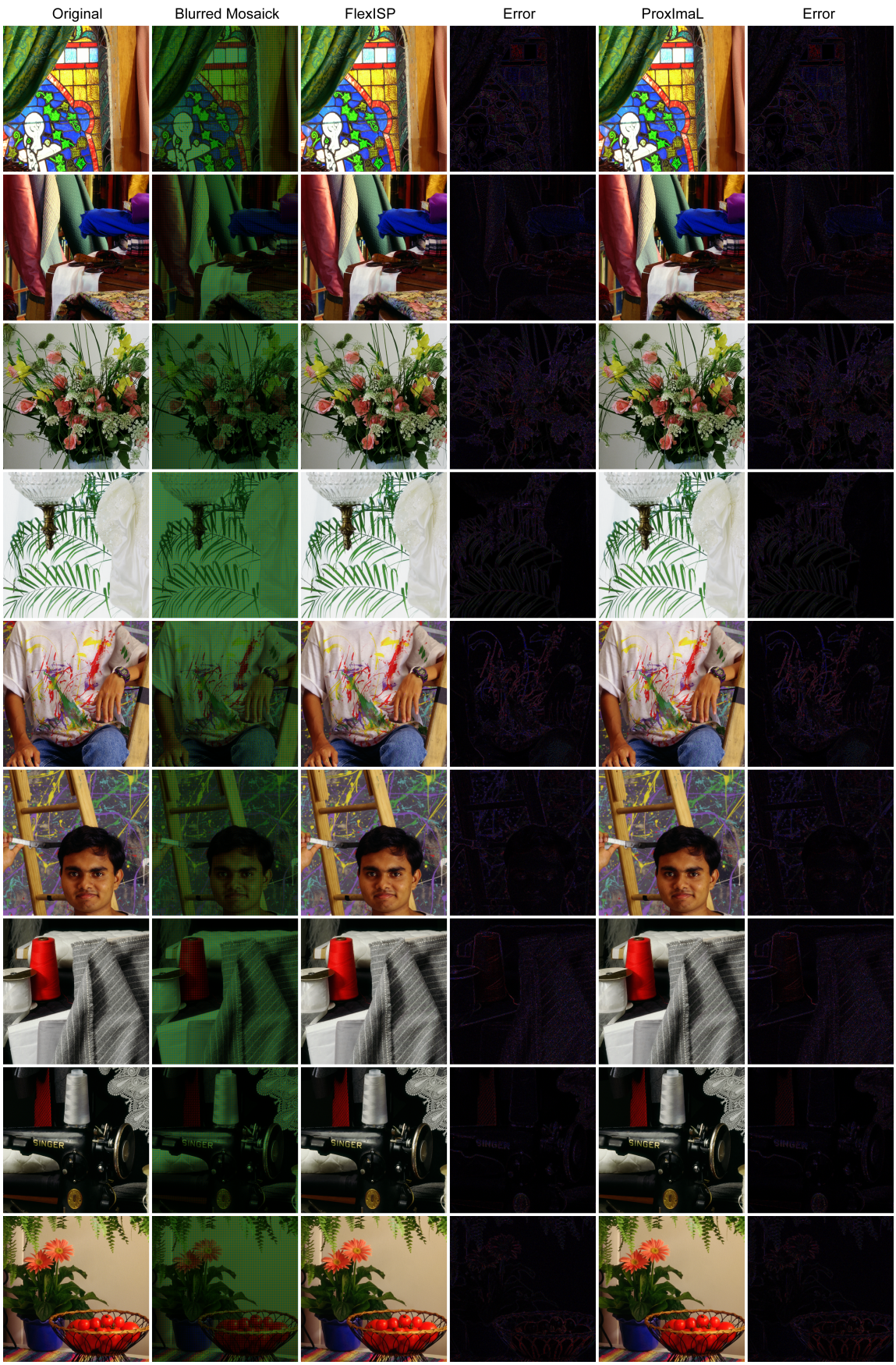

Figure 6: *Individual results for demosaicing of datasets 1–9.*

Figure 7: *Individual results for demosaicing of datasets 10–18.*

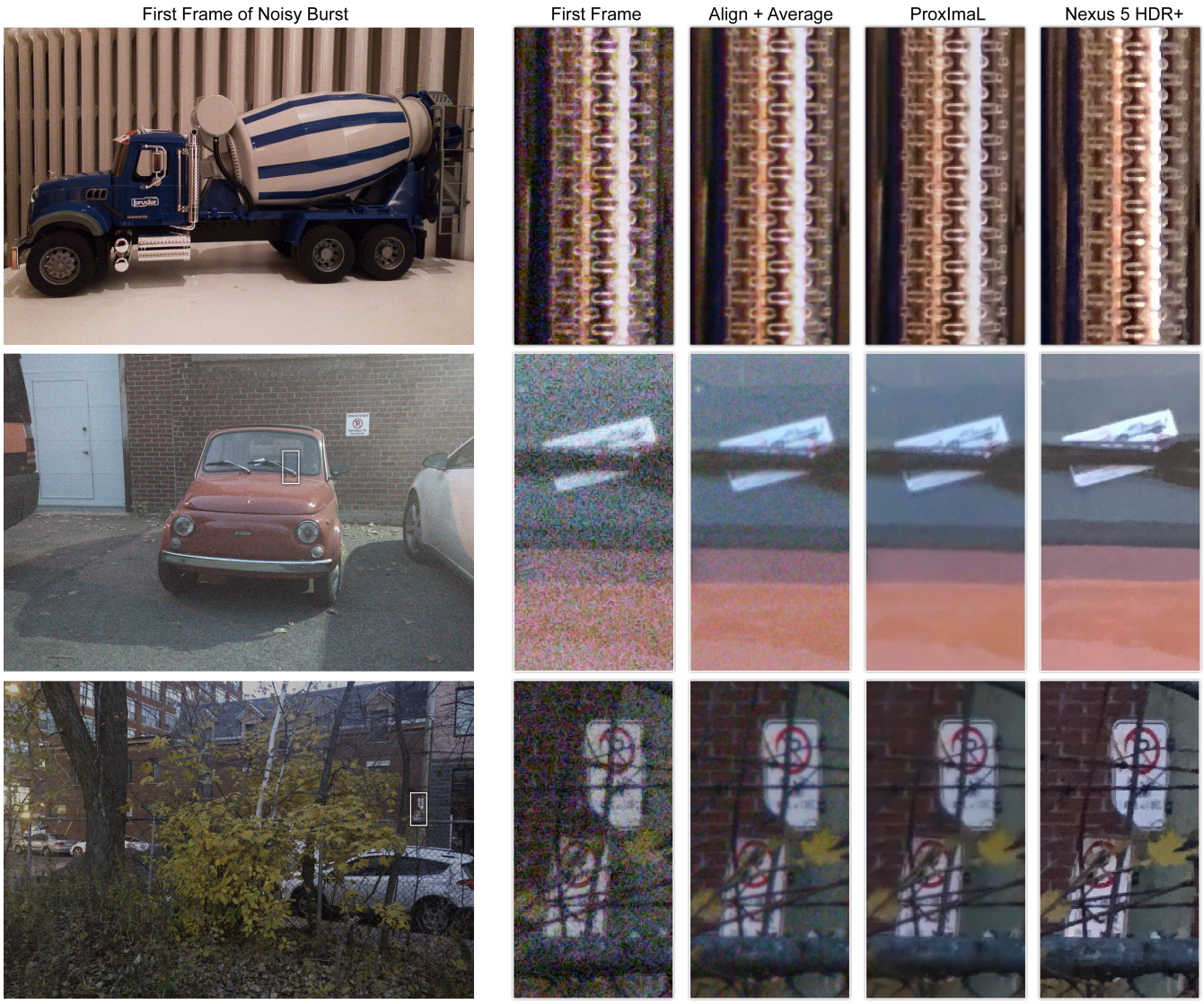

Figure 8: *Burst denoising and demosaicking from data captured with a Nexus 5 cellphone camera. The results achieved by ProxImaL are comparable to, if not better than, the HDR+ application.*

gradient is given by

$$\nabla f(\mathbf{x}) = \mathbf{x} - \mathcal{F}^{-1}\left(\mathbf{b} \cdot \frac{\mathcal{F}\mathbf{x}}{|\mathcal{F}\mathbf{x}|}\right) + \rho_1 \nabla^T(\nabla\mathbf{x} - \mathbf{v_1}) + \rho_2(\mathbf{x} - \mathbf{v_2}) \tag{19}$$

The function $f(\mathbf{x})$ is non-convex, so we are not guaranteed to find the global optimum. However, we use the current iterate $\mathbf{x}^k$ as a starting point, which lets us minimize $f(\mathbf{x})$ reliably in practice. It is important to not that different optimization algorithms and different splittings lead to different quadratic terms $f(\mathbf{x})$. In particular, we found that the slack term including ∇ regularizes the gradient for the L-BFGS substep. Splitting in a different way without including this term often causes the L-BFGS step to get stuck in local minima.

4.3 Implementation & Experiment

We compared our approach of combining HIO with ADMM (HIO+ADMM), implemented in our framework, with a more standard approach of combining HIO and error reduction (HIO+ER) [Fannjiang and Liao 2012]. For both algorithms, we ran HIO for 1000 iterations with feedback parameter $\beta = 0.9$. The initial support was generated with the Patterson function using 4% threshold. Every 20 iterations, fixed-threshold Shrinkwrap algorithm was applied to update the support. The Gaussian blur kernel started with σ equals 3 and reduced by 1% every application down to a minimum of 1.5. We tested Shrinkwrap thresholds of 0.05, 0.10, 0.15 and 0.20 and manually picked the best result. The error reduction algorithm ran for 100 iterations, which was enough to achieve a sufficiently small error in the Fourier domain and relative difference between iterations.

We used four standard test images and 2D projections of four molecules' electron density maps from the PDB protein database. Test images were resized to 128×128 and zero-padded to 256×256, resulting in 2x oversampling in Fourier domain. Fig. 14 shows the reconstruction result with a 2D projection of caffeine's electron density map. The estimated support in this case is inaccurate and the HIO+ER result is poor, with many artifacts. The HIO+ADMM result, by contrast, improves the reconstruction fidelity while reducing artifacts.

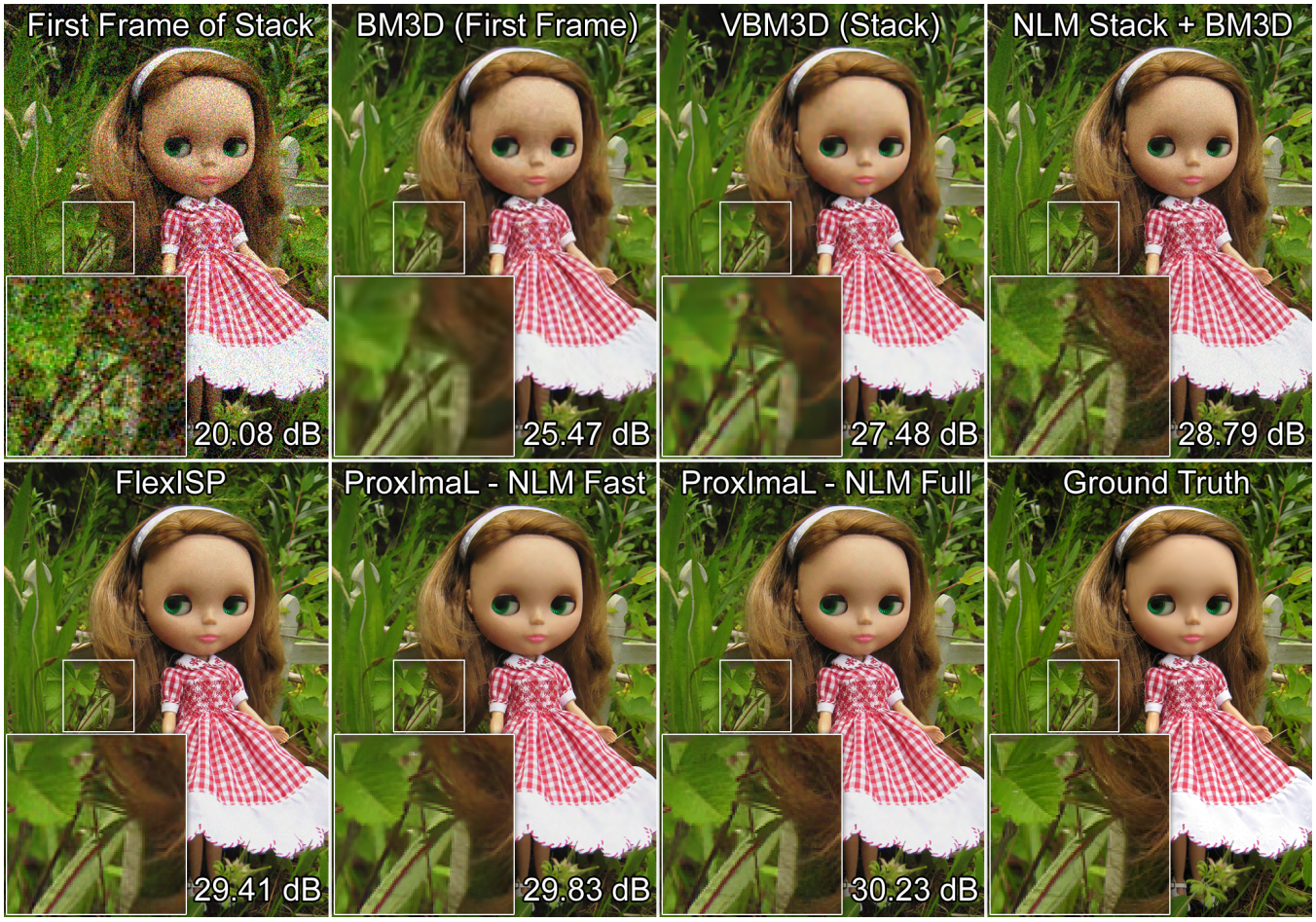

Figure 9: *Burst denoising and demosaicking for a simulation. For this example, we evaluate a range of different image priors, including BM3D, the fast approximation of non-local means (NLM) implemented by OpenCV, and a slow full implementation of NLM.*

References

ATTOUCH, H., BOLTE, J., AND SVAITER, B. F. 2011. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Mathematical Programming 137*, 1, 91–129.

BAUSCHKE, H. H., COMBETTES, P. L., AND LUKE, D. R. 2002. Phase retrieval, error reduction algorithm, and fienup variants: a view from convex optimization. *JOSA A 19*, 7, 1334–1345.

BAUSCHKE, H. H., COMBETTES, P. L., AND LUKE, D. R. 2003. Hybrid projection–reflection method for phase retrieval. *JOSA A 20*, 6, 1025–1034.

BECK, A. 2015. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *SIAM Journal on Optimization 25*, 1, 185–209.

BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. 2001. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning 3*, 1, 1–122.

BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning 3*, 1, 1–122.

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, 120–145.

DIAMOND, S., AND BOYD, S. 2016. Matrix-free convex optimization modeling. In *Optimization and Applications in Control and Data Sciences*. Springer. To appear.

ECKSTEIN, J., AND BERTSEKAS, D. 1992. On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming 55*, 1, 293–318.

ELSER, V. 2003. Phase retrieval by iterated projections. *JOSA A 20*, 1, 40–55.

ESSER, E., ZHANG, X., AND CHAN, T. 2010. A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM Journal on Imaging Sciences 3*, 4, 1015–1046.

FANNJIANG, A., AND LIAO, W. 2012. Phase retrieval with random phase illumination. *JOSA A 29*, 9, 1847–1859.

Image		1	2	3	4	5	6	7	8	9	10	11	12	Avg
Kernel 1	[Figueiredo and Jose 2010]	25.8142	21.4341	23.3527	23.7134	23.3938	18.5066	23.8740	22.3713	16.9006	23.1424	22.3405	21.0019	22.1538
Kernel 1	[Krishnan and Fergus 2009]	25.5704	20.8482	22.5548	23.4475	23.0111	18.2188	23.0683	21.9006	16.6501	22.4449	21.7963	20.5991	21.6758
Kernel 1	ProxImaL	25.8718	21.5173	23.3779	23.7933	23.4381	18.5530	23.8730	22.3358	17.0628	23.1524	22.3427	21.0172	**22.1946**
Kernel 2	[Figueiredo and Jose 2010]	25.3400	21.3473	23.7837	23.7896	23.2339	18.4321	23.6904	22.3161	16.8379	23.1683	22.0238	20.3501	**22.0261**
Kernel 2	[Krishnan and Fergus 2009]	25.0407	20.4035	21.8358	22.9868	22.5607	17.9267	22.4004	21.3340	16.3017	21.8357	20.8363	19.4776	21.0783
Kernel 2	ProxImaL, Kernel 2	25.4544	21.3226	23.1530	23.6229	23.1652	18.4075	23.4647	22.1170	16.8714	22.9302	21.7011	20.2097	21.8683
Kernel 3	[Figueiredo and Jose 2010]	27.0900	23.7496	25.4193	24.8615	25.5906	19.1358	23.9267	22.9460	18.1143	25.5309	24.0594	22.7077	23.5943
Kernel 3	[Krishnan and Fergus 2009]	26.4862	22.0279	23.5321	24.4620	24.3605	18.6870	24.1412	22.7496	17.5864	23.3817	22.9864	22.0359	22.7031
Kernel 3	ProxImaL	27.1352	24.0237	26.0872	25.4488	25.6932	19.2281	26.0142	23.8389	18.3303	25.7491	24.7727	22.9974	**24.1099**
Kernel 4	[Figueiredo and Jose 2010]	28.0897	24.5837	27.4909	26.1088	23.9506	19.2011	23.7945	23.3342	18.5090	27.3978	25.0220	21.5291	24.0843
Kernel 4	[Krishnan and Fergus 2009]	27.4198	22.7656	25.3194	25.3841	23.2999	19.0780	23.0971	22.7566	18.0080	25.2567	23.6207	21.2306	23.1030
Kernel 4	ProxImaL	28.1564	24.7609	27.8331	26.4878	23.9183	19.5558	23.9859	23.6711	18.5112	27.6335	25.1302	22.0718	**24.3097**
Kernel 5	[Figueiredo and Jose 2010]	29.4009	25.9487	28.3667	26.9152	28.9828	22.4154	27.8265	24.9435	20.5284	27.8352	28.2462	25.7322	26.4285
Kernel 5	[Krishnan and Fergus 2009]	28.8585	24.1237	25.9828	26.3871	27.4551	21.9288	27.1363	24.3077	19.7402	25.4880	26.3797	24.4814	25.7227
Kernel 5	ProxImaL	29.3980	26.1265	28.4659	27.1120	28.8667	22.6430	28.3052	25.1810	20.5318	28.0936	28.5163	25.7227	**26.5802**

Figure 10: *Poisson deconvolution. We report individual PSNR values for all 12 test images, for 5 different blur kernels, and for three reconstruction methods. On average, ProxImaL achieves the best performance.*

FIENUP, J. R. 1982. Phase retrieval algorithms: a comparison. *Applied Optics 21*, 15, 2758–2769.

FIGUEIREDO, M., AND BIOUCAS-DIAS, J. 2009. Deconvolution of Poissonian images using variable splitting and augmented lagrangian optimization. In *SWorkshop on Statistical Signal Processing*, 733–736.

GEMAN, D., AND YANG, C. 1995. Nonlinear image recovery with half-quadratic regularization. *IEEE Trans. Image Processing 4*, 7, 932–946.

GERCHBERG, R. W. 1972. A practical algorithm for the determination of phase from image and diffraction plane pictures. *Optik 35*, 237.

HEIDE, F., ROUF, M., HULLIN, M. B., LABITZKE, B., HEIDRICH, W., AND KOLB, A. 2013. High-quality computational imaging through simple lenses. *ACM Trans. Graph. 32*, 5, 149.

HEIDE, F., STEINBERGER, M., TSAI, Y.-T., ROUF, M., PAJAK, D., REDDY, D., GALLO, O., LIU, J., HEIDRICH, W., EGIAZARIAN, K., KAUTZ, J., AND PULLI, K. 2014. FlexISP: A flexible camera image processing framework. *ACM Trans. Graph. (SIGGRAPH Asia) 33*, 6.

KRISHNAN, D., AND FERGUS, R. 2009. Fast image deconvolution using hyper-Laplacian priors. In *Advances in Neural Information Processing Systems*, 1033–1041.

LI, G., AND PONG, T. K. 2015. Global convergence of splitting methods for nonconvex composite optimization. *arXiv e-Print 1407.0753*.

LUKE, D. R. 2005. Relaxed averaged alternating reflections for diffraction imaging. *Inverse Problems 21*, 1, 37.

MARCHESINI, S., HE, H., CHAPMAN, H. N., HAU-RIEGE, S. P., NOY, A., HOWELLS, M. R., WEIERSTALL, U., AND SPENCE, J. C. 2003. X-ray image reconstruction from a diffraction pattern alone. *Physical Review B 68*, 14, 140101.

MÖLLENHOFF, T., STREKALOVSKIY, E., MOELLER, M., AND CREMERS, D. 2015. The primal-dual hybrid gradient method for semiconvex splittings. *SIAM Journal on Imaging Sciences 8*, 2, 827–857.

MOREAU, J.-J. 1965. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France 93*, 273–299.

NOCEDAL, J., AND WRIGHT, S. 2006. *Numerical Optimization*. Springer Science.

OSHEROVICH, E. 2012. Numerical methods for phase retrieval. *arXiv e-Print 1203.4756*.

PARIKH, N., AND BOYD, S. 2013. Proximal algorithms. *Foundations and Trends in Optimization 1*, 3, 123–231.

POCK, T., CREMERS, D., BISCHOF, H., AND A.CHAMBOLLE. 2009. An algorithm for minimizing the Mumford-Shah functional. In *Proceedings of the IEEE International Conference on Computer Vision*, 1133–1140.

ROBINI, M., AND ZHU, Y. 2015. Generic half-quadratic optimization for image reconstruction. *SIAM Journal on Imaging Sciences 8*, 3, 1752–1797.

WANG, Y., YANG, J., YIN, W., AND ZHANG, Y. 2008. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences 1*, 3, 248–272.

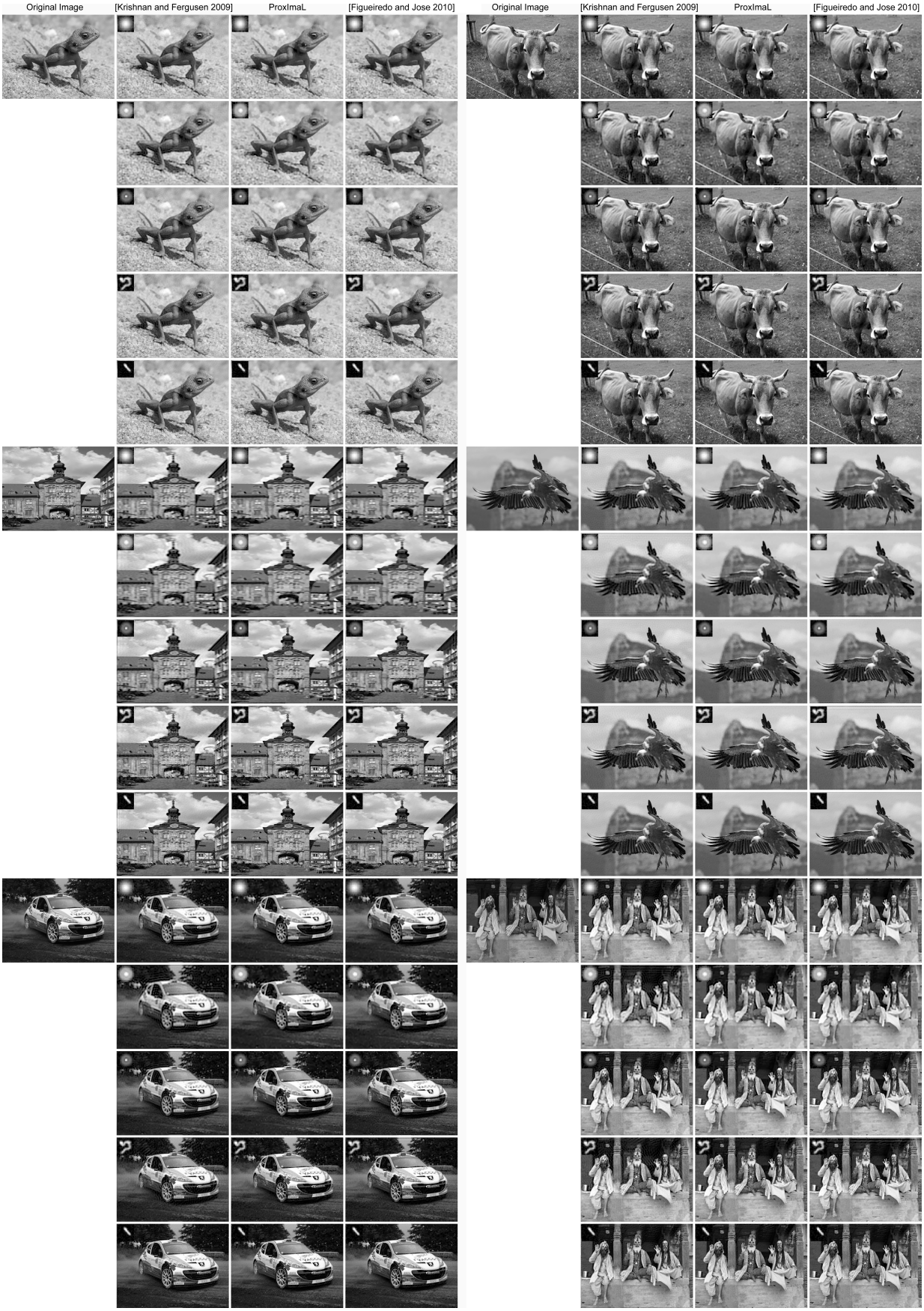

Figure 11: *Poisson deconvolution results for test images 1–6. Note that all test images are scaled to the same aspect ratio for convenience of visualization, but each of them actually has a slightly different aspect ratio and resolution.*

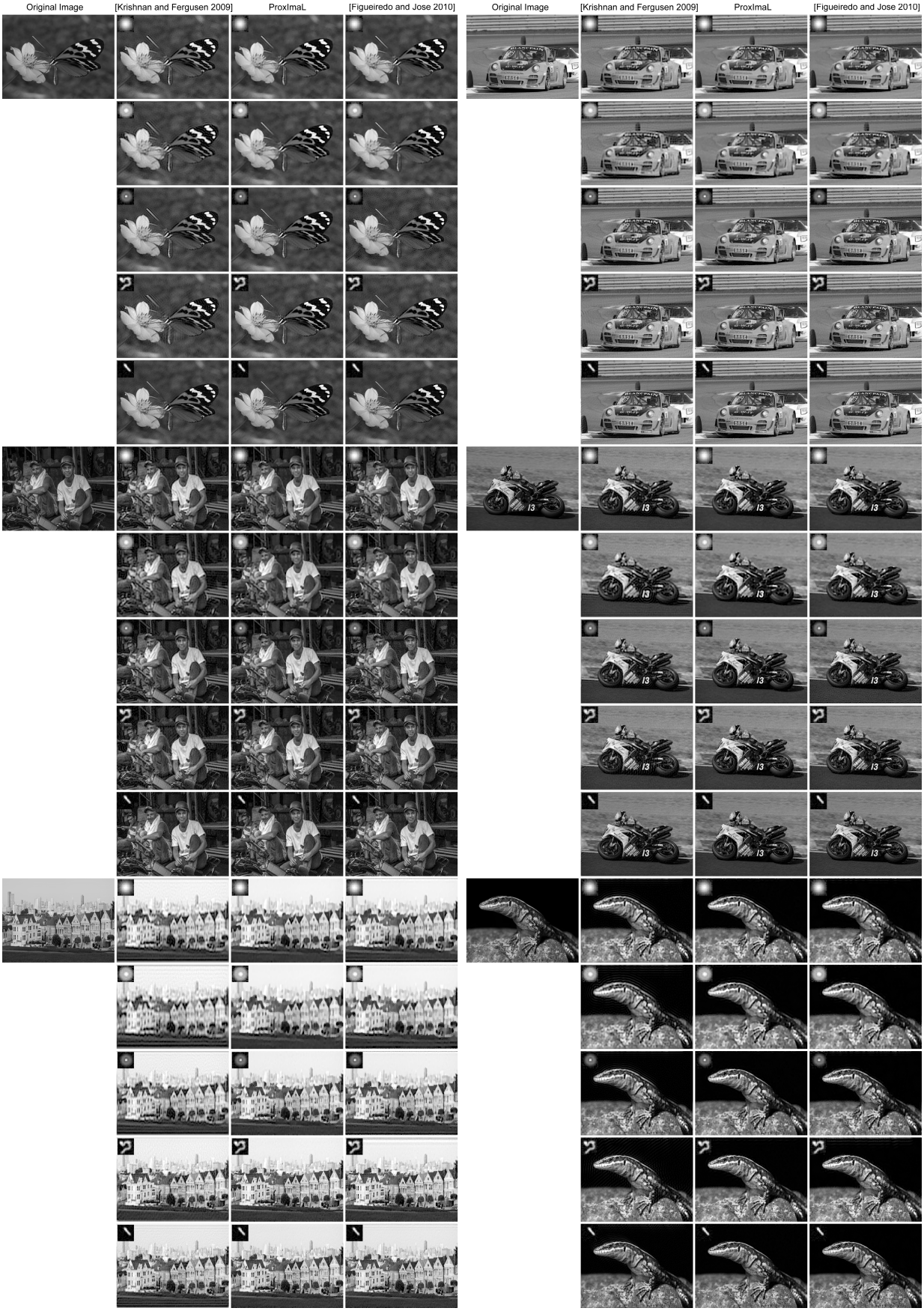

Figure 12: *Poisson deconvolution for test images 7–12. Note that all test images are scaled to the same aspect ratio for convenience of visualization, but each of them actually has a slightly different aspect ratio and resolution.*

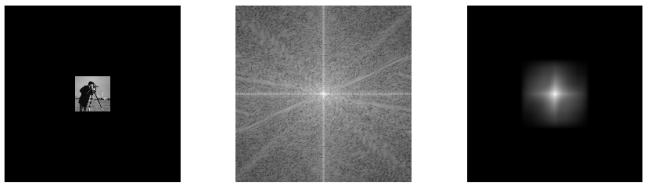

Figure 13: *(left) the cameraman with extra padding;(middle) Fourier amplitude of padded image in log space;(right) output of autocorrelation function.*

Figure 14: *(left)* 128×128 *2D projection of caffeine's electron density map;(left middle)* 256×256 *groundtruth support;(middle) HIO+ER output(29.27db);(right middle)HIO+ADMM output(34.13db);(right)Estimated support*