

Ulugbek S. Kamilov^{ID}, Charles A. Bouman^{ID},
Gregory T. Buzzard^{ID}, and Brendt Wohlberg^{ID}

Plug-and-Play Methods for Integrating Physical and Learned Models in Computational Imaging

Theory, algorithms, and applications



©SHUTTERSTOCK.COM/PAPAPIG

Plug-and-play (PnP) priors constitute one of the most widely used frameworks for solving computational imaging problems through the integration of physical models and learned models. PnP leverages high-fidelity physical sensor models and powerful machine learning methods for prior modeling of data to provide state-of-the-art reconstruction algorithms. PnP algorithms alternate between minimizing a data fidelity term to promote data consistency and imposing a learned regularizer in the form of an image denoiser. Recent highly successful applications of PnP algorithms include biomicroscopy, computerized tomography (CT), magnetic resonance imaging (MRI), and joint ptychotomography. This article presents a unified and principled review of PnP by tracing its roots, describing its major variations, summarizing main results, and discussing applications in computational imaging. We also point the way toward further developments by discussing recent results on equilibrium equations that formulate the problem associated with PnP algorithms.

Historical background

Consider the inverse problem of estimating an unknown image $\mathbf{x} \in \mathbb{R}^n$ from its noisy measurements $\mathbf{y} \in \mathbb{R}^m$. It is common to formulate this problem using the optimization

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{R}^n}{\operatorname{argmin}} f(\mathbf{x}) \quad \text{with} \quad f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}) \quad (1)$$

where g is a data fidelity term that quantifies consistency with the observed measurements $\mathbf{y}$ and h is a regularizer that enforces prior knowledge on $\mathbf{x}$. The formulation in (1) corresponds to the maximum a posteriori probability (MAP) estimator when

$$g(\mathbf{x}) = -\log(p_{\mathbf{y}|\mathbf{x}}(\mathbf{x})) \quad \text{and} \quad h(\mathbf{x}) = -\log(p_{\mathbf{x}}(\mathbf{x})) \quad (2)$$

where $p_{\mathbf{y}|\mathbf{x}}$ is the likelihood relating $\mathbf{x}$ to measurements $\mathbf{y}$ and $p_{\mathbf{x}}$ is the prior distribution. For example, given measurements of the form $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$, where $\mathbf{A}$ is the measurement operator (also known as the *forward operator*) characterizing the response of the imaging instrument and $\mathbf{e}$ is additive white Gaussian noise (AWGN), the data fidelity term reduces to the

quadratic function $g(\mathbf{x}) = (1/2) \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$. On the other hand, many popular image regularizers are based on a sparsity promoting regularizer $h(\mathbf{x}) = \tau \|\mathbf{W}\mathbf{x}\|_1$, where $\tau > 0$ is the regularization parameter and $\mathbf{W}$ is a suitable transform. Over the years, a variety of reasonable choices of h have been proposed, with examples including the total variation (TV) and Markov random field (MRF) functions. These functions have elegant analytical forms and had a major impact in applications ranging from tomography for medical imaging to image denoising for cell phone cameras.

The solution of (1) balances the requirements to be both data consistent and plausible, which can be intuitively interpreted as finding a balance between two manifolds: the sensor manifold and prior manifold. The sensor manifold is represented by small values of $g(\mathbf{x})$ and in the case of a linear forward model, is roughly an affine subspace of $\mathbb{R}^n$. Likewise, the prior manifold is represented by small values of $h(\mathbf{x})$ and includes the images that are likely to occur in our application. Importantly, real images have enormous amounts of structure, departures from which are immediately noticeable to a domain expert. Consequently, plausible images lie near a lower dimensional manifold in the higher dimensional embedding space.

Proximal algorithms are often used for solving problems of the form in (1) when g or h is nonsmooth [1]. One of the most widely used and effective proximal algorithms is the alternating direction method of multipliers (ADMM), which uses an augmented Lagrangian formulation to allow for alternating minimization of each function in turn (see [2] for an overview of ADMM). ADMM computes the solution of (1) by iterating the steps summarized in Algorithm 1 (see Figure 1) until convergence. One important property of ADMM is that it does not explicitly require knowledge of either g or h or their gradients, relying instead on the proximal operator, which is defined as

$$\text{prox}_{\tau h}(\mathbf{z}) := \underset{\mathbf{x} \in \mathbb{R}^n}{\text{argmin}} \left\{ \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \tau h(\mathbf{x}) \right\} \quad (3)$$

for any proper, closed, and convex function h [2]. Comparing (3) and (1), we see that the proximal operator can be interpreted as a MAP estimator for the AWGN denoising problem

$$\mathbf{z} = \mathbf{x}_0 + \mathbf{w} \quad \text{where} \quad \mathbf{x}_0 \sim p_{\mathbf{x}_0}, \quad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \tau \mathbf{I}) \quad (4)$$

by setting $h(\mathbf{x}) = -\log(p_{\mathbf{x}_0}(\mathbf{x}))$.

This perspective inspired the development of PnP priors in [3], where the $\text{prox}_{\gamma h}$ step in ADMM is simply replaced by a more general black-box denoiser $D: \mathbb{R}^n \rightarrow \mathbb{R}^n$, such as block-matching and 3D filtering (BM3D) [4]. That is, any black-box denoiser D can, in principle, replace (“plug”-in for) $\text{prox}_{\gamma h}$, and then ADMM algorithm can run (“play”) as before. We refer to this original algorithm as *PnP-ADMM* to distinguish it from other methods inspired by this PnP approach. In fact, there are multiple algorithms using proximal maps to minimize a sum of convex functions, and for each of these algorithms, there is a corresponding PnP version obtained by associating the proximal map with the prior term, then replacing the proximal map with a black-box denoiser. In the following, we provide more detail on PnP-ADMM and PnP-fast iterative shrinkage/thresholding algorithm (FISTA) [5] (based on the proximal gradient method) as well as extensions and variations. See [6] for the roots of FISTA, [7] for more detail on proximal splitting methods in general, and [8] for a tutorial overview of some PnP methods.

PnP integration of physical and learned models

Deep learning (DL) has emerged as a powerful paradigm for designing algorithms for various image restoration and reconstruction tasks, including denoising, deblurring, and super-resolution (the literature is vast, but see [9] for an early history). Given a set of paired data $(\mathbf{x}_i, \mathbf{z}_i)$, where $\mathbf{x}_i$ is the desired “ground truth” image and $\mathbf{z}_i$ is its noisy or corrupted observation, the traditional supervised DL strategy is to learn a mapping from $\mathbf{z}_i$ to $\mathbf{x}_i$ by training a deep convolutional neural network (CNN). Despite its empirical success in computational imaging, an important drawback of DL relative to regularized inversion is the potential need to retrain the CNN for different measurement operators and noise levels.

The success of CNNs as black-box denoisers leads naturally to their use with PnP (see “Turning an Image Denoising Network Into an Image Superresolver”). In its simplest form, PnP can be implemented by pretraining an image denoising CNN D and using D in place of $\text{prox}_{\gamma h}$ within ADMM. Remarkably, this simple heuristic of using denoisers not associated with any h exhibited great empirical success and spurred much theoretical and algorithmic work on PnP and other related methods. As a result, PnP-inspired methods are among the most widely used approaches for combining the advantages of regularized inversion, which is flexible to changes in the data fidelity term, with the powerful representation capabilities of deep CNNs.

Algorithm 1. ADMM.

```

1: input:  $\mathbf{u}^0 = \mathbf{0}$ ,  $\mathbf{x}^0$ , and  $\gamma > 0$ 
2: for  $k = 1, 2, \dots, t$  do
3:    $\mathbf{z}^k \leftarrow \text{prox}_{\gamma g}(\mathbf{x}^{k-1} - \mathbf{u}^{k-1})$ 
4:    $\mathbf{x}^k \leftarrow \text{prox}_{\gamma h}(\mathbf{z}^k + \mathbf{u}^{k-1})$ 
5:    $\mathbf{u}^k \leftarrow \mathbf{u}^{k-1} + (\mathbf{z}^k - \mathbf{x}^k)$ 
6: return  $\mathbf{x}^t$ 

```

Algorithm 2. FISTA.

```

1: input:  $\mathbf{x}^0 = \mathbf{s}^0$ ,  $\gamma > 0$ , and  $\{\theta_k\}_{k \geq 0}$ 
2: for  $k = 1, 2, \dots, t$  do
3:    $\mathbf{z}^k \leftarrow \mathbf{x}^{k-1} - \gamma \nabla g(\mathbf{x}^{k-1})$ 
4:    $\mathbf{s}^k \leftarrow \text{prox}_{\gamma h}(\mathbf{z}^k)$ 
5:    $\mathbf{x}^k \leftarrow (1 - \theta_k) \mathbf{s}^k + \theta_k \mathbf{s}^{k-1}$ 
6: return  $\mathbf{x}^t$ 

```

FIGURE 1. ADMM and FISTA are two widely used iterative algorithms for minimizing composite functions $f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$, where the regularization term h is nonsmooth. Both functions avoid differentiating h by evaluating its proximal operator.

PnP priors algorithms

The first PnP algorithm was PnP-ADMM [10], which is implemented by iterating the steps in Algorithm 3 (see Figure 2) until convergence. The operator D in PnP-ADMM is an image denoiser that approximates a solution to the problem in (4). While the original formulation of PnP relies on ADMM, PnP can be equally effective when used with other proximal algorithms, such as primal–dual splitting [11] and FISTA [5]. Algorithm 4 summarizes the steps of PnP-FISTA, which is based on the traditional FISTA summarized in Algorithm 2.

PnP-ADMM and PnP-FISTA share the important feature of modularity; they explicitly separate the application of the physical models (the data fidelity update in line 3 of both algorithms) from that of the learned models (the image denoising in line 4 of both algorithms). This observation reveals a key strength of PnP methods: they can be easily customized for different measurement operators by changing the data fidelity term, thus enabling the use of the same learned CNN over a wide range of applications without retraining. Similarly, PnP methods provide a simple mechanism to combine different image priors on

Turning an Image Denoising Network into an Image Superresolver

Plug-and-play (PnP) can be applied to multiple imaging problems using a single convolutional neural network (CNN) denoiser simply by changing the physics-based measurement model. Consider image superresolution (ISR) with factors $2\times$ and $4\times$, where the goal is to recover a high-resolution image from its blurred, decimated, and noisy low-resolution (LR) observation. As shown in the following, a single denoising CNN can be used within PnP to address both problems, thus leveraging the implicit image model in a deep CNN over multiple problems without retraining.

Step 1: Learn a denoiser

Let $\mathcal{X} \subset \mathbb{R}^n$ denote a training data set of natural images. The denoiser is trained by updating the weights θ of a CNN D_θ to remove the noise from $z = x_0 + w$, where

$x_0 \in \mathcal{X}$ and $w \sim \mathcal{N}(0, \sigma^2 I)$. It is worth mentioning that since the prior in PnP is learned on a pretext task (image denoising) rather than on the final task (image reconstruction), PnP can be considered a self-supervised learning framework.

Step 2: Turn denoiser into superresolver

A pretrained denoiser D_θ can be used for ISR by replacing $\text{prox}_{\gamma h}$ in the alternating direction method of multipliers or the fast iterative shrinkage/thresholding algorithm by D_θ (Figure S1). Figure S2 shows the results obtained for two upsampling rates $2\times$ and $4\times$ using the same denoising CNN, either to postprocess the pseudoinverse or as an image prior within PnP. Note how PnP obtains significantly better results by integrating information both from the physical and learned models.

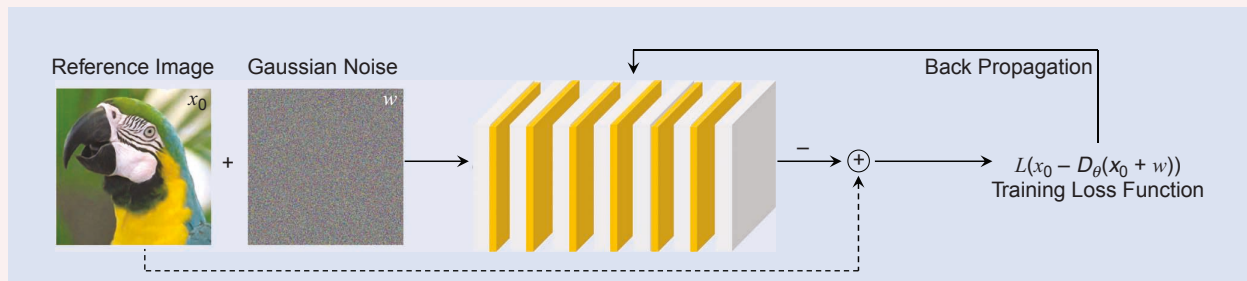


FIGURE S1. Image priors for PnP can be obtained by training CNNs to remove the additive white Gaussian noise from a set of images.

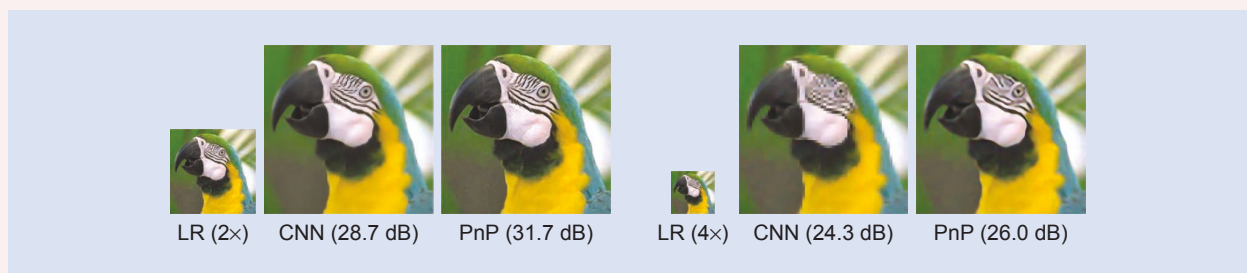


FIGURE S2. A single pretrained CNN denoiser in PnP can address different superresolution factors. The code is available at https://github.com/lan/scico-data/blob/main/notebooks/superres_ppp_dncnn_admm.ipynb.

the same problem by simply changing the denoiser D . Note that since the prior in PnP is learned on a pretext task (image denoising) rather than on the final task (image reconstruction), PnP can be considered a self-supervised learning framework.

One of the practical differences between various PnP algorithms is the treatment of the data fidelity term, g , which models the physical measurements. PnP-FISTA uses a standard (explicit) gradient descent step $z = x - \gamma \nabla g(x)$, while PnP-ADMM uses the proximal operator $\text{prox}_{\gamma g}$, which can be written as an implicit gradient step, $z = \text{prox}_{\gamma g}(x) = x - \gamma \nabla g(z)$, with ∇g evaluated at z . We assume, for simplicity, that g is differentiable; extensions to nondifferentiable g require more care. For the loss $g(x) = (1/2) \|Ax - y\|_2^2$, these updates can be computed as

$$\begin{aligned} x - \gamma \nabla g(x) &= x - A^\top (Ax - y) \quad \text{and} \\ \text{prox}_{\gamma g}(x) &= (I + \gamma A^\top A)^{-1} (x + \gamma A^\top y). \end{aligned} \quad (5)$$

PnP-ADMM is known for its fast empirical convergence and efficiency for many widely used operators in computational imaging. However, it requires the computation of the proximal map, whereas PnP-FISTA requires only the computation of the gradient, ∇g . In principle, the gradient is simpler than the proximal map, but in many applications, the proximal map can be computed or approximated efficiently using general methods, such as the conjugate gradient, or with specialized methods, such as when the forward model is a spatial blurring operator that can be computed using the fast Fourier transform (FFT) [14]. In other cases, the proximal map can be efficiently computed using partial updates [15] that avoid the explicit inversion of $(I + \gamma A^\top A)$ in (5). This is accomplished by maintaining an additional state variable that is used as initialization for the proximal minimization problem. The minimization is approximated by a few steps of an iterative solver,

Algorithm 3. PnP-ADMM [10].

```

1: input:  $u^0 = 0$ ,  $x^0$ , and  $\gamma > 0$ 
2: for  $k = 1, 2, \dots, t$  do
3:    $z^k \leftarrow \text{prox}_{\gamma g}(x^{k-1} - u^{k-1})$ 
4:    $x^k \leftarrow D(z^k + u^{k-1})$ 
5:    $u^k \leftarrow u^{k-1} + (z^k - x^k)$ 
6: return  $x^t$ 

```

Algorithm 4. PnP-FISTA [5].

```

1: input:  $x^0 = s^0$ ,  $\gamma > 0$ , and  $\theta_k \in (0, 1] \forall k$ 
2: for  $k = 1, 2, \dots, t$  do
3:    $z^k \leftarrow x^{k-1} - \gamma \nabla g(x^{k-1})$ 
4:    $s^k \leftarrow D(z^k)$ 
5:    $x^k \leftarrow (1 - \theta_k)s^k + \theta_k s^{k-1}$ 
6: return  $x^t$ 

```

FIGURE 2. The term plug-and-play (PnP) priors refers to a family of iterative algorithms that replace the proximal operator $\text{prox}_{\gamma h}: \mathbb{R}^n \rightarrow \mathbb{R}^n$ of the regularizer h (as in Figure 3) by a more general denoiser $D: \mathbb{R}^n \rightarrow \mathbb{R}^n$ in line 4. The success of DL in image restoration has led to a wide adoption of PnP for exploiting learned priors specified through pretrained deep CNNs, leading to state-of-the-art performance in a variety of applications.

starting from this initialization. As the outer loop converges, this additional state variable also converges [15], so these partial updates reduce computation without compromising the accuracy of the final solution.

An important conceptual point is that PnP algorithms with black-box denoisers do not generally solve an optimization problem. That is, the original ADMM and FISTA algorithms solve the optimization problem in (1). But once the proximal map denoising operation is replaced with a black-box denoiser, D , then there is no longer any corresponding function h to minimize. In fact, the numerical evaluation of many widely used denoisers, including BM3D and the denoising CNN, reveals that their Jacobians are not symmetric, which implies that these denoisers are neither proximal maps nor gradient descent steps [16, Th. 1].

Nonetheless, it is still possible to formulate a criterion for the converged solution for PnP using a consensus equilibrium formulation [17] (also see “Geometric Intuition for Multiagent Consensus Equilibrium”) given by

$$x = G(x - u) \quad \text{and} \quad x = D(x + u) \quad (6)$$

where $G := \text{prox}_{\gamma g}$ and x and u are the converged values of PnP-ADMM. Interestingly, in the consensus equilibrium equation of (6), x is the final reconstruction, and u can be interpreted as noise that is removed by the denoiser in $x = D(x + u)$ on the one hand and balanced by the action of the data fitting update in $x = G(x - u)$ on the other.

To establish (6), note that the fixed points z , x , and u of PnP-ADMM iteration satisfy

$$z = G(x - u), \quad x = D(z + u), \quad u = u + z - x. \quad (7)$$

From the last equation, we conclude that $x = z$, which leads directly to (6). Also, the first-order optimality condition for the minimization problem $x = G(x - u) = \text{prox}_{\gamma g}(x - u)$ is $0 = x - (x - u) + \gamma \nabla g(x)$, so $u = -\gamma \nabla g(x)$. A similar analysis in [18] and [19] shows that the fixed points of PnP-FISTA satisfy the same consensus equilibrium conditions:

$$\begin{aligned} x = D(x - \gamma \nabla g(x)) &\Leftrightarrow \begin{cases} x + u = x - \gamma \nabla g(x) \\ x = D(x + u) \end{cases} \\ &\Leftrightarrow \begin{cases} x = G(x - u) \\ x = D(x + u) \end{cases}, \end{aligned} \quad (8)$$

where we again used the first-order optimality condition to convert from ∇g to G .

The convergence of PnP algorithms can be established using monotone operator theory [20]. In this approach, as in [14], [17], [18], [19], [21], [22], and [23], the problem is first expressed as finding a fixed point of some high-dimensional operator, which under appropriate hypotheses can be iterated to yield a solution. The proof of convergence of PnP-ADMM [17], [23] begins by showing that the fixed points of the PnP-ADMM are in one-to-one correspondence with the fixed points of the operator

$$T := (2G - I)(2D - I). \quad (9)$$

In fact, after a linear change of coordinates, Algorithm 3 is equivalent to the Mann iterations of T given by $v^k \leftarrow (1/2)v^{k-1} + (1/2)T(v^{k-1})$ [1], [17]. This yields linear convergence to a unique fixed point when T is a contraction, which is true when g is strongly convex and $R := I - D$ is a sufficiently strong contraction [23]. Weaker conditions establish sublinear convergence to a possibly nonunique fixed point [24]. Other well-known theoretical results on PnP-ADMM include its convergence for implicit proximal operators [21], bounded denoisers [22], and linearized Gaussian mixture model denoisers [14]. Even CNN-based denoisers can be trained to satisfy these contractive, nonexpansive, and Lipschitz conditions by using spectral normalization techniques [23], [25].

The convergence of PnP-ISTA (which is PnP-FISTA with the Nesterov acceleration parameters set to $\theta_k = 1$ for all $k \geq 1$) can be established by expressing it using operator

$F := D(I - \gamma \nabla g)$. When the data fidelity term g is strongly convex and the denoiser D is Lipschitz continuous with a sufficiently small constant, then the iteration of F converges linearly to its unique fixed point [23]. On the other hand, when g is weakly convex and the denoiser D is firmly nonexpansive, then the iteration converges sublinearly to its fixed point [19]. Related results have shown that PnP-ISTA converges to a minimizer of some global cost function when the denoiser D corresponds to a minimum mean-square error estimator applied to the denoising problem in (4) [26], and they have established recovery guarantees for PnP for the measurement operators that satisfy the restricted eigenvalue condition commonly used in compressive sensing [27].

Regularization by denoising

Regularization by denoising (RED) is an algorithm inspired by PnP that also enables the integration of denoisers as priors

Geometric Intuition for Multiagent Consensus Equilibrium

Multiagent consensus equilibrium (MACE) is a formal mechanism to compute a reconstruction that is a balance among multiple competing models or agents (Figure S3). When the agents are a forward and prior model, the MACE solution is exactly the plug-and-play solution. But MACE also works with multiple models, each promoting different desired outcomes.

An agent is a function, $x = F(v)$, that takes an input image, v , and makes it better in some way to produce a new image, x . Denoisers, gradient descent steps, and proximal maps are all examples of useful agents. Beyond that, agents can be neural networks trained to remove application-specific artifacts and noise as well as other image enhancing operations.

Algorithm S1 shows how to compute the MACE solution with n agents, shown as $F_1, \dots, F_n$. At convergence, all

agents yield the same reconstruction: $F_n(v_n) = x^*$ for all n . This is the idea of consensus. This reconstruction is also the average of the input points: $\text{Average}(v_1, \dots, v_n) = v^*$. This is the idea of equilibrium.

Algorithm S1. MACE reconstruction [17].

```

1: input: Initial Reconstruction  $x$  (an image)
2:  $v \leftarrow (x, \dots, x)$  # Form a stack of images
3: while not converged do
4:  $x \leftarrow (F_1(v_1), \dots, F_n(v_n))$  # Apply each agent
5:  $w \leftarrow 2x - v$ 
6:  $z \leftarrow \text{Average}(w_1, \dots, w_n)$ 
7:  $z \leftarrow (z, \dots, z)$  # Restack the average image
8:  $v \leftarrow v + 2\rho(z - x)$  # Update agent input
9: end while
10:  $x^* \leftarrow z$  # Get final result

```

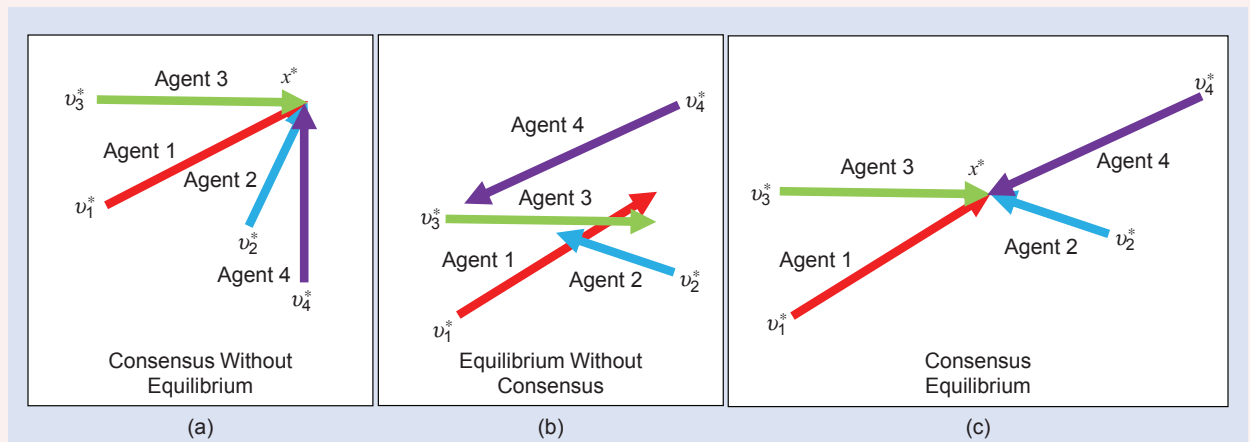


FIGURE S3. Consensus means that each agent has the same output, while equilibrium means that vector sum of the updates is $\mathbf{0}$. (a) Consensus without equilibrium. (b) Equilibrium without consensus. (c) Consensus equilibrium.

for inverse problems [28]. The steepest descent (SD) variant of RED [28] can be implemented by iterating the following steps until convergence:

$$\mathbf{x}^k \leftarrow \mathbf{x}^{k-1} - \gamma H(\mathbf{x}^{k-1}) \quad \text{with} \quad H(\mathbf{x}) := \nabla g(\mathbf{x}) + \tau(\mathbf{x} - D(\mathbf{x})) \quad (10)$$

where $\gamma > 0$ is the step size, D is a denoiser, and $\tau > 0$ is the regularization parameter. While RED was initially derived as an optimization problem [28], a subsequent analysis [16] showed that an interpretation as a fixed-point problem is more appropriate for practical denoisers. Using this approach, the fixed-point condition in (10) translates to the RED equilibrium condition given by

$$-\nabla g(\mathbf{x}) = \tau(\mathbf{x} - D(\mathbf{x})). \quad (11)$$

As noted in (2), the function g is often the negative log likelihood of the distribution $p_{\mathbf{y}|\mathbf{x}}$ relating the reconstruction $\mathbf{x}$ to the measurements $\mathbf{y}$. In this setting, $-\nabla g$ is known as the “score” of this distribution. This negative gradient describes the steepness of the log-likelihood function and hence the sensitivity to changes in $\mathbf{x}$. From this, we see that (11) balances changes in the log likelihood against the update step $\mathbf{x} - D(\mathbf{x})$. This is similar to the balance in (6), in which the same $\mathbf{u}$ is removed by the denoiser and added by the data fitting map. A much more complete discussion of RED algorithms and score matching is given in [16]. The convergence of RED algorithms can also be analyzed using monotone operator theory. In particular, it can be shown that for a convex function g and a nonexpansive denoiser D , RED-SD converges sublinearly to a set of $\mathbf{x}$ satisfying the equilibrium condition in (11) [16], [25].

Figure 3 presents results using PnP and RED on compressive sensing from random projections with 20% subsampling. The setup used in the simulation is identical to that described in [27]. The results of the traditional TV reconstruction and of the ISTA-Net+ deep unfolding (DU) architecture [29] are presented for reference (see the “DU

and DEQ Models” section for a discussion of DU). The figure considers two priors for PnP: 1) an AWGN denoiser and 2) an artifact removal (AR) operator trained to remove artifacts specific to the PnP iterations that is used in place of an AWGN denoiser. Both priors are implemented using the denoising CNN architecture, with its batch normalization layers removed to enable control of the Lipschitz constant of the network via spectral normalization. The AR operator D is trained by including it in a DU architecture that performs PnP iterations and training it end to end in a supervised fashion. This approach has the disadvantage that the prior model is no longer completely separate from the forward model, but as seen in Figure 3, it can yield significantly improved results relative to an AWGN denoiser.

Online PnP algorithms

The traditional PnP methods are batch algorithms in the sense that they compute the gradient ∇g or the proximal operator $\text{prox}_{\gamma g}$ of the data fidelity term g by using the whole measurement vector $\mathbf{y} \in \mathbb{R}^m$. The per-iteration computational and memory complexity of batch PnP algorithms depends on the total number of measurements. For example, in tomography with b projections, the complexity of evaluating ∇g scales linearly with b , making it computationally expensive for a large b . This has motivated interest in online, stochastic, and/or incremental PnP algorithms that approximate the batch ∇g with an approximation $\hat{\nabla} g$ based on a single element or a small subset of the measurements [19], [24], [30].

Consider the decomposition of $\mathbb{R}^m$ into $b \geq 1$ blocks:

$$\mathbb{R}^m = \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \cdots \times \mathbb{R}^{m_b} \quad \text{with} \quad m = m_1 + m_2 + \cdots + m_b. \quad (12)$$

In this setting, the data fidelity term and the corresponding gradient vector are given by

$$g(\mathbf{x}) = \frac{1}{b} \sum_{i=1}^b g_i(\mathbf{x}) \quad \text{and} \quad \nabla g(\mathbf{x}) = \frac{1}{b} \sum_{i=1}^b \nabla g_i(\mathbf{x}) \quad (13)$$

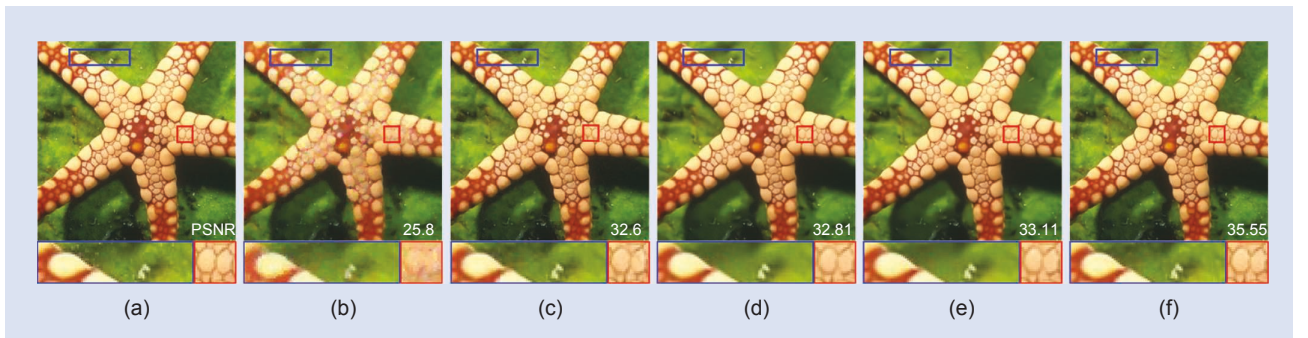


FIGURE 3. Color image recovery in compressive sensing from random projections with 20% subsampling. The results of TV and a well-known deep unfolding (DU) architecture, ISTA-Net+, are provided for reference. The PnP (denoising) and RED (denoising) methods use a pretrained AWGN denoiser as an image prior. The PnP [artifact removal (AR)] method uses a problem-dependent AR operator pretrained using DU. Note that the choice of denoiser affects the reconstruction significantly. The number on the bottom-right corner is the peak signal-to-noise ratio (PSNR) in dB. (a) Ground truth, (b) TV, (c) ISTA-Net+, (d) RED (denoising), (e) PnP (denoising), and (f) PnP (AR).

where each g_i is evaluated only on the subset $y_i \in \mathbb{R}^{m_i}$ of the full measurement vector $y \in \mathbb{R}^m$. For example, each individual term in the gradient can be set to the quadratic function $g_i(x) = (1/2) \|y_i - A_i x\|_2^2$, where A_i is the operator corresponding to the measurement block y_i .

Online PnP in Algorithm 5 and scalable iterative minibatch algorithm (SIMBA) in Algorithm 6 (Figure 4) are online extensions of PnP-FISTA and RED-SD, respectively. Both algorithms improve the scalability to large-scale measurements by using only a single component gradient $\nabla g_{i_k}(x)$, with $i_k \in \{1, \dots, b\}$, making their per-iteration complexity independent of b . Online PnP algorithms can be implemented using different block selection rules. The strategy commonly adopted for the theoretical analysis focuses on selecting indices i_k as independent identically distributed random variables distributed uniformly over $\{1, \dots, b\}$. An alternative would be to proceed in epochs of b consecutive iterations, where the set $\{1, \dots, b\}$ is reshuffled at the start of each epoch and the index ∇g_{i_k} is selected from this ordered set.

Online PnP algorithms can also be implemented in a minibatch fashion by replacing ∇g_{i_k} in step 4 of both by a minibatch gradient

$$\hat{\nabla} g(x) = \frac{1}{p} \sum_{j=1}^p \nabla g_{i_j}(x) \quad (14)$$

where p is the minibatch size and $i_1, \dots, i_p$ are indices selected from the set $\{1, \dots, b\}$. The minibatch variants of online PnP can process several blocks in parallel in every iteration, thus improving efficiency on multiprocessor hardware architectures. While online algorithms have traditionally focused on partial approximations of the gradient, recent work has also explored the approximation of the batch proximal operator $\text{prox}_{\gamma g}$ in PnP-ADMM by a partial proximal operator

Algorithm 5. Online PnP [19].

- 1: **input:** x^0 , $b \geq 1$, and $\gamma > 0$
- 2: **for** $k = 1, 2, \dots, t$ **do**
- 3: Choose an index $i_k \in \{1, \dots, b\}$
- 4: $z^k \leftarrow x^{k-1} - \gamma \nabla g_{i_k}(x^{k-1})$
- 5: $x^k \leftarrow D(z^k)$

Algorithm 6. SIMBA [30].

- 1: **input:** x^0 , $b \geq 1$, $\gamma > 0$, and $\tau > 0$
- 2: **for** $k = 1, 2, \dots, t$ **do**
- 3: Choose an index $i_k \in \{1, \dots, b\}$
- 4: $H_{i_k}(x^{t-1}) \leftarrow \nabla g_{i_k}(x^{t-1}) + \tau R(x^{t-1})$
- 5: $x^k \leftarrow x^{k-1} - \gamma H_{i_k}(x^{k-1})$

FIGURE 4. The complexity of evaluating the batch gradient of ∇g or proximal operator $\text{prox}_{\gamma g}$ is computationally expensive in some applications. This has motivated the development of online, stochastic, and incremental variants of PnP that use a single element or a small subset of the measurements at each iteration. The per-iteration complexity of such algorithms is independent of the batch size $b \geq 1$, thus making them suitable for certain large-scale applications.

$\text{prox}_{\gamma g_i}$ [24]. One can also consider block coordinate extensions of online PnP by considering decomposition of the image space $\mathbb{R}^n$ into a number of smaller image blocks [25].

The fixed-point convergence analysis of online PnP algorithms uses mathematical tools from traditional stochastic optimization and monotone operator theory. The key requirement for the analysis is that the gradient estimate in (14) is unbiased and has bounded variance,

$$\mathbb{E}[\hat{\nabla} g(x)] = \nabla g(x) \quad \text{and} \quad \mathbb{E}[\|\nabla g(x) - \hat{\nabla} g(x)\|_2^2] \leq \frac{\nu^2}{p} \quad (15)$$

for some constant $\nu > 0$ and every $x \in \mathbb{R}^n$. Note that when i_k is selected uniformly at random from $\{1, \dots, b\}$, the unbiasedness assumption is automatically satisfied. Then, for convex data fidelity terms g_i and firmly nonexpansive denoisers D , one can establish the sublinear convergence of online PnP algorithms to their fixed points [19], [24], [30].

DU and DEQ

DU (also known as *deep unrolling* or *algorithm unrolling*) is a DL paradigm with roots in sparse coding [31], [32] that has gained popularity in computational imaging, due to its ability to provide a systematic connection between iterative algorithms and CNN architectures [32], [33]. Many PnP algorithms have been turned into DU architectures by parameterizing the operator D_θ as a CNN with weights θ , truncating the PnP algorithm to a fixed number of iterations $t \geq 1$, and training the corresponding architecture end to end in a supervised fashion. For example, Figure 5 illustrates the representation of t iterations of PnP-ISTA and RED-SD as DU architectures.

Consider a set of paired data (x_i, y_i) , where x_i is the desired “ground truth” image and $y_i = Ax_i + e_i$ is its noisy

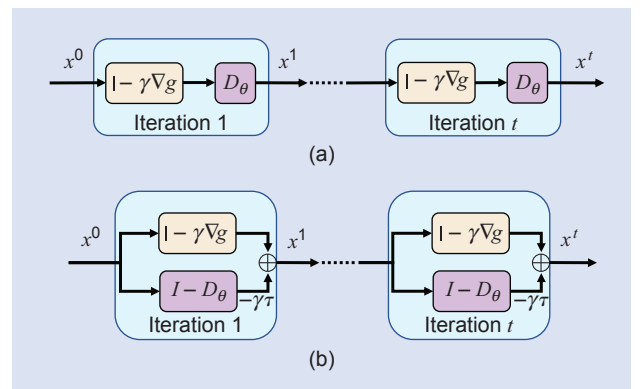


FIGURE 5. The PnP framework is related to two other popular computational imaging paradigms, DU and deep equilibrium (DEQ) models. A PnP algorithm, such as PnP-ISTA and RED-SD, can be turned into a DU architecture by truncating the algorithm to $t \geq 1$ iterations and training the weights θ of the CNN D_θ end to end. Similarly, a DEQ architecture can be obtained by running the PnP algorithm until convergence and using the implicit differentiation at the fixed point to train the weights θ . The operator D_θ in DU/DEQ is not necessarily an AWGN denoiser; instead, it is an AR operator trained to remove artifacts specific to the PnP iterations. The (a) deep unfolding of PnP-ISTA and (b) deep unfolding of RED-SD.

observation. Consider also the iterate $\mathbf{x}_i^t(\boldsymbol{\theta})$ of a PnP algorithm truncated to $t \geq 1$ iterations, where we made explicit the dependence of the PnP output on the weights $\boldsymbol{\theta}$ of the CNN parameterizing $D_{\boldsymbol{\theta}}$. DU interprets the steps required for mapping the input vector $\mathbf{y}_i$ and the initialization $\mathbf{x}_i^0$ to the output $\mathbf{x}_i^t(\boldsymbol{\theta})$ as layers of a CNN architecture. The DU training is performed by solving the optimization problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_i L(\mathbf{x}_i, \mathbf{x}_i^t(\boldsymbol{\theta})) \quad (16)$$

where L is a loss function that quantifies the discrepancy between the true and predicted solutions. Once trained using (16), the truncated PnP algorithm can be directly used for imaging [27].

Deep equilibrium (DEQ) models are a recent extension of DU to an arbitrary number of iterations [34]. DEQ can be implemented by replacing $\mathbf{x}_i^t(\boldsymbol{\theta})$ in (16) by a fixed-point $\bar{\mathbf{x}}_i(\boldsymbol{\theta})$ of a given PnP algorithm and using implicit differentiation for updating the weights $\boldsymbol{\theta}$. The benefit of DEQ over DU is that it doesn't require the storage of the intermediate variables for solving (16), thus reducing the memory complexity of training. However, DEQ requires the computation of the fixed-point $\bar{\mathbf{x}}_i(\boldsymbol{\theta})$, which can increase the computational complexity.

There are some important differences between traditional PnP and DU/DEQ. Traditional PnP relies on an AWGN denoiser as an image prior. On the other hand, the operator $D_{\boldsymbol{\theta}}$ in DU/DEQ is not an AWGN denoiser; instead, it is an AR operator trained to remove artifacts specific to the PnP iterations. As seen in Figure 3, which shows the relative performance of PnP using an AWGN denoiser and a pretrained AR operator, this problem-specific training can yield significantly improved results. However, this performance comes at a cost; while the prior in traditional PnP is fully decoupled from the measurement operator, that of DU/DEQ is trained by accounting for the measurement operator $\mathbf{A}$. Hence the DU/DEQ approach has reduced generality and higher computational/memory complexity of training since the AR prior is trained for the specific task of reconstruction from random projections rather than for AWGN denoising.

Related approaches

There is a wide variety of approaches to learning and using prior information in the context of inverse imaging, far too many to describe completely. Early work on non-CNN-learned priors includes the expected patch log likelihood [35], which uses a cost function approach on patches. Work related to the use of CNNs as priors includes [36], which describes certain empirical advantages enjoyed by CNN denoisers; [37], which uses CNN denoisers with (F)ISTA on a modified cost function with convergence/accuracy benefits [38]; and [39], which is related to RED with a motivation that comes from an analysis of denoising autoencoders [40]. In addition to DU/DEQ, another approach to improving the performance of PnP-inspired methods is to fine-tune denoisers for a specialized distribution of images. Examples of this include [41], with images from the

same class as the observed image; [14], with images from the same scene as the observed image; and [42], with training on the single observed image.

Tradeoffs and limitations

A key idea of PnP-inspired methods is to encapsulate Bayesian prior information into an algorithmic denoiser. This approach has the benefit of promoting code modularity in that data fitting updates and denoisers can be developed separately, with many possible pairings of data updates and denoisers. The downside of this generality is that some reconstruction quality is lost; under ideal conditions, an end-to-end trained system can outperform a general-purpose PnP system. DU and DEQ methods fall somewhere in the middle in that they have separate data update and denoising modules, but the denoiser is trained as part of an end-to-end system to enhance reconstruction quality.

We note also that as with any inversion method, particularly one with learned priors, PnP methods involve a number of hyperparameters to be tuned. For PnP-ADMM and PnP-FISTA, one of the most important of these is the distribution of images used to train a denoiser, most especially the assumed noise level. In practice, the noise level needed for optimal reconstruction may not be known at training time. Approaches to address this include training a denoiser for multiple noise levels [43] and reconciling multiple denoisers using multiagent consensus equilibrium (MACE) [17]. An additional factor is the architecture of the NN, which can play an important role in the quality of results and the time required for reconstructions. However, this is a complex design problem with much ongoing work and many problem-specific considerations. Finally, the use of learned priors introduces the possibility of mismatch between training data and application data. Some work on the effect of such mismatch is described in [42], [44], and [45].

MACE

MACE, introduced in [17], is a framework that extends PnP-ADMM to more than two update terms and provides an equilibrium interpretation to the problem solved by PnP-ADMM. As noted in the preceding, PnP-ADMM with a black-box denoiser does not generally solve an optimization problem but instead solves the equilibrium problem in (6). MACE takes this equilibrium condition as a starting point and extends it to allow for multiple types of models, including physics-based, data-driven, and application-specific models. When a physics-based forward model and a denoiser-based prior model are used, then the MACE solution is the same as the PnP solution. But MACE is more general and offers more flexibility in the choice of models as well as algorithms for computing the solution.

A MACE agent is a function $F: \mathbb{R}^n \rightarrow \mathbb{R}^n$ that takes in an image $\mathbf{v}$ and produces an "improved" output image, $\mathbf{x} = F(\mathbf{v})$. So, a denoiser and proximal maps are examples of agents, but other agents might implement AR and other heuristic improvements. We denote the agents by $F_1, \dots, F_\ell$, each of which maintains its own version $\mathbf{v}_j$ of the input image. We can then

stack input images as $\mathbf{v} = [\mathbf{v}_1, \dots, \mathbf{v}_\ell]$ and the output images as $\mathbf{F}(\mathbf{v}) = [F_1(\mathbf{v}_1), \dots, F_\ell(\mathbf{v}_\ell)]$. At the same time, we define an averaging operator $\mathbf{G}(\mathbf{v}) = (\bar{\mathbf{v}}, \dots, \bar{\mathbf{v}})$, where $\bar{\mathbf{v}}$ is the average of the $\{\mathbf{v}_j\}$. With this notation, the MACE equation is

$$\mathbf{F}(\mathbf{v}^*) = \mathbf{G}(\mathbf{v}^*) \quad (17)$$

and the final reconstruction is the average of the vectors $\mathbf{v}_j^*$.

As seen in “Geometric Intuition for Multiagent Consensus Equilibrium,” the MACE equation in (17) has an interpretation as the consensus equilibrium. Since each entry in $\mathbf{G}$ is identical, all agents must output the same reconstruction, so $F_j(\mathbf{v}_j) = F_k(\mathbf{v}_k)$ for all j and k : this is consensus. Moreover, since this consensus point is the same as the average of the input points encoded in $\mathbf{v}^*$, the updates $\mathbf{v}_j - F_j(\mathbf{v}_j)$ must sum to zero: this is equilibrium. As with PnP and RED, the MACE equations can be converted to a fixed-point problem by noting that the averaging operator $\mathbf{G}$ is a linear projection and so

has the property that $\mathbf{G}^2 = \mathbf{G}$. This implies that $(2\mathbf{G} - \mathbf{I})$ is its own inverse, which means that $(2\mathbf{G} - \mathbf{I})(2\mathbf{G} - \mathbf{I}) = \mathbf{I}$. Then, from (17), we see that $2\mathbf{F}(\mathbf{v}^*) - \mathbf{v}^* = 2\mathbf{G}(\mathbf{v}^*) - \mathbf{v}^*$, so multiplying both sides by $(2\mathbf{G} - \mathbf{I})$, we have that (17) is equivalent to

$$\mathbf{T}(\mathbf{v}^*) = \mathbf{v}^*, \quad \text{where } \mathbf{T} := (2\mathbf{G} - \mathbf{I})(2\mathbf{F} - \mathbf{I}).$$

As in the preceding, we can solve this fixed point problem via Mann iterations [20]. For MACE, this takes the form of $\mathbf{v} \leftarrow (1 - \rho)\mathbf{v} + \rho\mathbf{T}(\mathbf{v})$, with $\rho \in (0, 1)$. This is guaranteed to converge when $\mathbf{T}$ is nonexpansive and has a fixed point but converges in practice for a wide variety of agents. “Geometric Intuition for Multiagent Consensus Equilibrium” gives additional intuition and a pseudocode implementation of this algorithm for solving the MACE equations.

Figure 6 illustrates the benefits of MACE for fusing the outputs of three separate 2D image denoisers to regularize the result of a 4D reconstruction problem in space and time

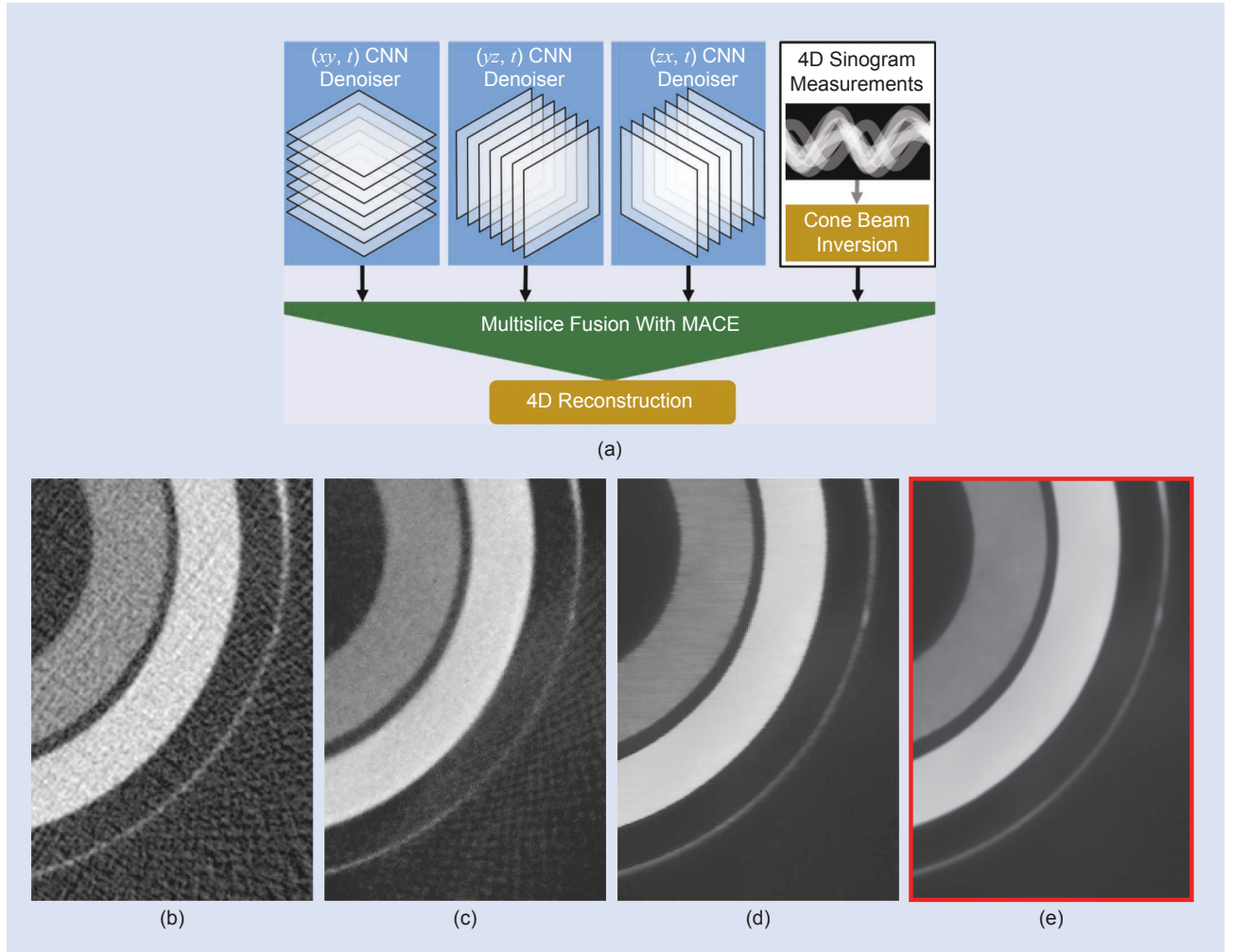


FIGURE 6. The application of MACE from [46], using multiple 2D denoisers to regularize a time-varying volume (4D). (a) The combination of 2D denoisers in multiple orientations with CT measurements using MACE. (b) and (c) The reference reconstructions using filtered backprojection (FBP) and model-based iterative reconstruction (MBIR). (d) The reconstruction using 2D denoisers in two out of three possible spatial orientations, which leads to streaking artifacts in the complementary direction. (e) The multislice fusion (MSF) MACE reconstruction using 2D denoisers in all three orientations.

[46]. Each denoiser operates along only two of the three dimensions, and MACE integrates these three denoisers along with a physical model of tomographic projections. The image using all four agents, labeled *multislice fusion*, has the best quality, while the images reconstructed with one missing denoising agent contain streaking artifacts aligned with the orientation of the missing agent. The use of multiple 2D denoisers has a number of important advantages over 3D/4D denoising. First, 2D processing is more efficient since memory access is more local and the number of nearest neighbors is smaller. Second, 2D denoisers implemented as deep NNs can be trained on widely available 2D images, whereas 3D/4D data are currently very limited.

PnP in practice

Implementing PnP algorithms

While PnP methods are generally easy to implement, there are several reference implementations that can be used as sources of inspiration and as utilities. Some open source libraries that include implementations of PnP algorithms include Sparse Optimization Research Code (<https://github.com/bwohlberg/porco>), PnP-MACE (<https://github.com/gbuzzard/PnP-MACE>), and, most recently, SCICO [49], providing a wide array of computational imaging tools in Python. SCICO is built on the Python package JAX, which provides support for seamless code transition between CPU and GPU, acceleration via just-in-time compilation, and automatic differentiation. In particular, the superresolution demonstration in “Turning an Image Denoising Network Into an Image Superresolver” was implemented using SCICO (see “Implementing PnP-ADMM Superresolution in SCICO”).

com/bwohlberg/porco), PnP-MACE (<https://github.com/gbuzzard/PnP-MACE>), and, most recently, SCICO [49], providing a wide array of computational imaging tools in Python. SCICO is built on the Python package JAX, which provides support for seamless code transition between CPU and GPU, acceleration via just-in-time compilation, and automatic differentiation. In particular, the superresolution demonstration in “Turning an Image Denoising Network Into an Image Superresolver” was implemented using SCICO (see “Implementing PnP-ADMM Superresolution in SCICO”).

Applications of PnP

PnP has been applied to a very wide range of problems, including superresolution [47] and blind deconvolution, various forms of tomographic imaging [46], MRI [13], and synthetic aperture radar [48], to name but a few. In this section, we present two applications of PnP in computational imaging: MRI and intensity diffraction tomography (IDT).

Figure 7 presents an application of PnP to a free-breathing 3D MRI problem described in [13]. The experimentally collected in vivo k -space measurements were acquired using T1-weighted stack-of-stars 3D spoiled gradient echo

Implementing PnP-ADMM Superresolution in SCICO

Scientific Computational Imaging Code (SCICO) [49] is an open source library for computational imaging that includes implementations of plug-and-play (PnP) algorithms. Since SCICO is built on the python package JAX, it provides seamless support for learned deep priors. In the following, we show the steps for implementing the PnP alternating direction method of multipliers (ADMM) for the example in Figure S2. We assume that a reference image has been loaded as the variable `img`, then set up problem parameters, such as the downsampling rate and noise level, and construct a downsampled and noise-corrupted measurement that will be superresolved:

```
rate = 4 # downsampling rate
σ = 2e-2 # noise standard deviation
Afn = lambda x: downsample_image(x, rate=rate) # forward operator
s = Afn(img) # downsample reference image
noise, key = scico.random.randn(s.shape, seed = 0)
sn = s + σ * noise # downsampled and noise-corrupted measurement
```

We next set up the inverse problem of form (1), where g is the least-squares function and h is used to invoke a denoising convolutional neural network (DnCNN) as the black-box denoiser:

```
A = linop.LinearOperator(input_shape=img.shape, output_shape=s.shape, eval_fn=Afn)
```

```
g = loss.SquaredL2Loss(y=sn, A = A)
C = linop.Identity(input_shape=img.shape)
h = functional.DnCNN("17M")
```

We obtain a baseline solution (and initializer for PnP) by denoising the pseudoinverse of the forward operator:

```
xpinv, info = solver.cg(A.T @ A, A.T @ sn, snp.zeros(img.shape))
dncnn = denoiser.DnCNN("17M") # construct denoiser object
xden = dncnn(xpinv) # denoised pseudo inverse solution
```

Finally, we set up ADMM to solve the inverse problem:

```
ρ = 3.4e-2 # ADMM penalty parameter
solver = ADMM(
    f=g, g_list=[h],
    C_list=[C], rho_list=[ρ],
    x0=xden, maxiter=12,
    itstat_options={"display": True},
    subproblem_solver=Linear SubproblemSolver(
        cg_kwargs={
            "tol": 1e-3,
            "maxiter": 10,
        }
    ),
)
xppp = solver.solve() # PnP solution.
```

sequence with fat suppression. The first three images in Figure 7 are reconstructions from 800 k -space radial lines, corresponding to scans of about 2 min. *Multicoil nonuniform inverse FFT (MCNUFFT)* refers to a simple inversion of the measurement operator without any regularization. *Deformation-compensating learning (DeCoLearn)* refers to a CNN trained in a self-supervised fashion to remove artifacts from MCNUFFT images obtained from 1,600 radial lines (scans of about 4 min) [12]. While DeCoLearn offers excellent reconstruction performance when applied to 2,000-line data (see *Reference* in Figure 7), its performance degrades when applied without retraining to 800-line data. Regularization by AR (RARE) is a variant of PnP that is obtained by simply replacing the AWGN denoiser with DeCoLearn [13]. Note that RARE successfully adapts DeCoLearn to 800-line data without retraining, which is due to its ability to leverage DeCoLearn as an image prior.

Figure 8 presents an application of PnP to biomicroscopy using the IDT instrument described in [30]. RED-SD and SIMBA are both used to reconstruct a 3D algae sample of $1,024 \times 1,024 \times 25$ voxels from $b = 89$ high-resolution intensity measurements. Both algorithms use exactly the same forward model and the same denoising CNN AWGN. The per-iteration memory complexity of RED-SD is about 75 GB, which includes the storage of the 3D complex-valued transfer functions for each illumination $\{A_i\}$, all the measured intensity images $\{y_i\}$, and the estimate of the desired image x . By using minibatches of size $p = 10$, SIMBA significantly reduces the per-iteration memory complexity to about 11 GB. Additionally, SIMBA has significant per-iteration computational advantage over RED-SD, due to its usage of minibatch gradients. Despite these memory and computational advantages

of SIMBA, the results in Figure 4 clearly show its comparable performance to RED-SD in terms of imaging quality.

Future directions

The idea of encapsulating prior information using algorithmic updates is a fertile area with much room for growth. For PnP-ADMM and related methods, there are questions about which denoisers provide guaranteed convergence, how to accelerate convergence, and how to manage the tradeoffs between modularity and reconstruction quality. These questions also apply to MACE, with additional questions about how to select hyperparameters for each agent, how to balance the contributions of multiple agents, and how to use agents that work in different spaces (e.g., the sinogram domain and space domain). And of course, there are many new applications to explore.

Conclusions

Since their introduction in 2013, PnP methods have become a standard tool for computational imaging. They have been used in a remarkably diverse range of applications in which they provide state-of-the-art performance. When they were introduced, they provided what was arguably the first practical approach to integrating learned models with imaging physics to solve inverse imaging problems. A significant factor in their rapid growth in popularity was the ease with which they can be implemented. Alternative approaches to achieving this goal have since emerged, and in some cases, they provide better reconstruction performance, but this is achieved at the expense of a potentially time-consuming and data-dependent application-specific training process. PnP and the multiagent extension MACE are particularly powerful for contexts in which the forward model is not fixed and in which there is insufficient labeled problem-specific training data.

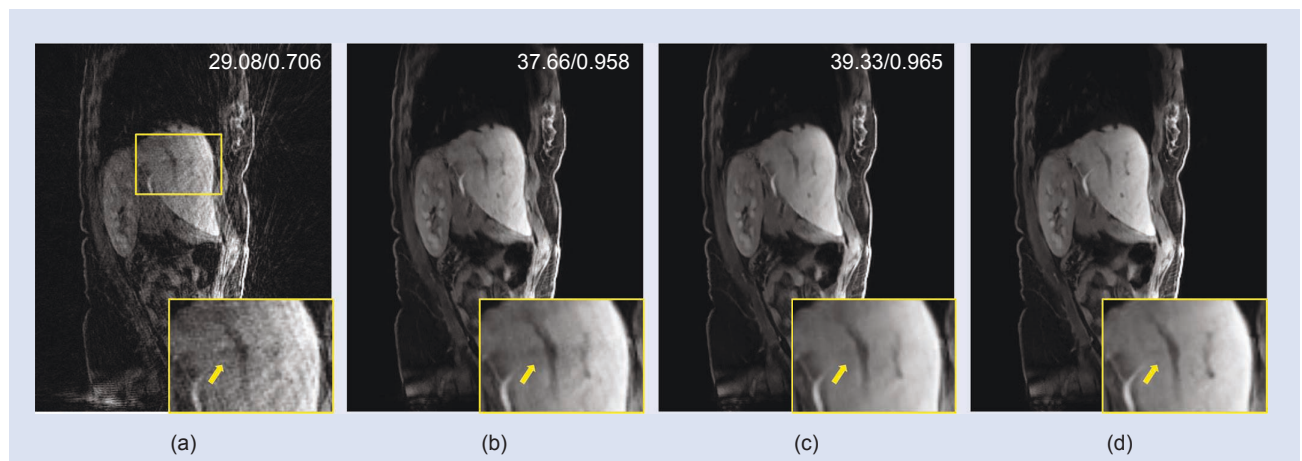


FIGURE 7. PnP algorithms explicitly separate the application of the forward model from that of the learned prior, enabling the adaptation of trained CNNs to new sensor configurations. This is illustrated on experimentally collected 3D MRI data corresponding to 800 radial spokes (scans of about 2 min). *Multicoil nonuniform inverse FFT (MCNUFFT)* refers to a simple inversion of the measurement operator without any regularization. *Deformation-compensating learning (DeCoLearn)* [12] is a CNN that was trained under a mismatched sensor configuration corresponding to 1,600 lines (scans of about 4 min). A variant of PnP called *regularization by AR (RARE)* [13] is used to adapt DeCoLearn to the desired 800-line data. The results of the DeCoLearn reconstruction using all the available 2,000 lines is shown as the reference image. The numbers on the top-right corner correspond to the relative PSNR/structural similarity index measure (SSIM) values with respect to the reference image. Note the ability of RARE to successfully adapt DeCoLearn to 800-line data. (a) MCNUFFT. (b) DeCoLearn. (c) RARE with DeCoLearn. (d) Reference image.

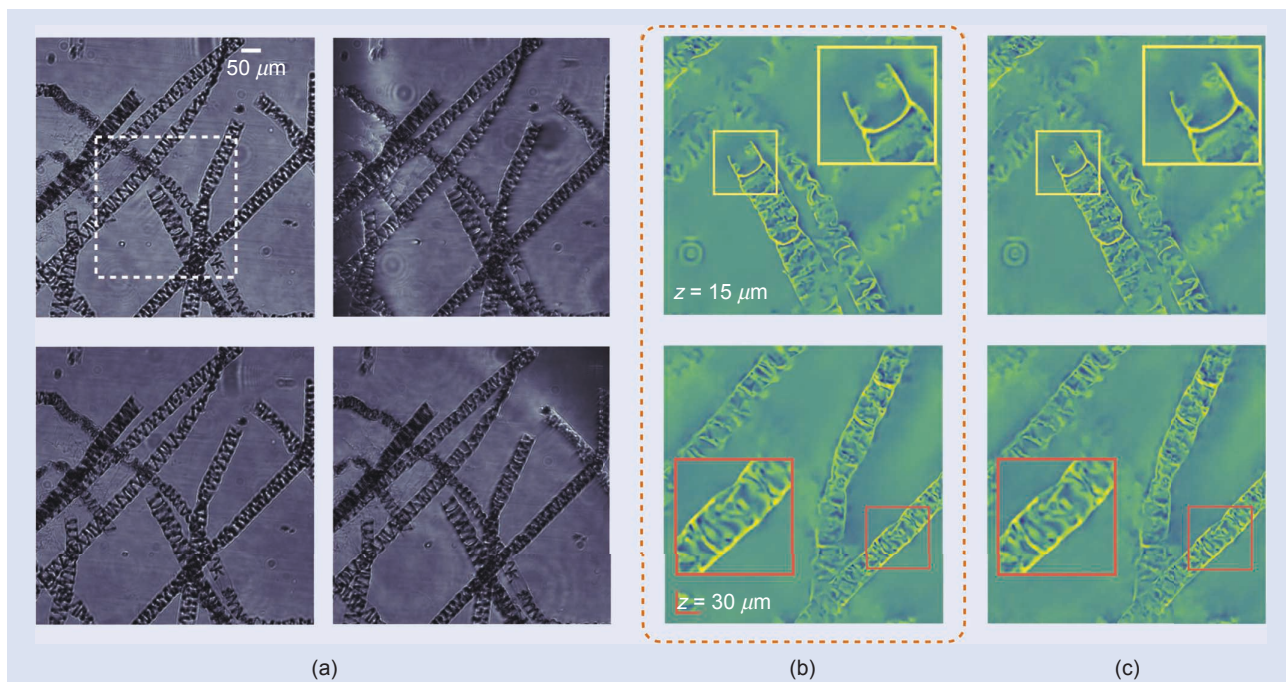


FIGURE 8. Online PnP algorithms, such as SIMBA, in Algorithm 6, can reduce the computational and memory complexity of PnP. (a) The reconstruction of a 3D algae sample from 89 experimentally collected intensity diffraction tomography (IDT) measurements. (b) SIMBA, which uses minibatches of size $p = 10$, is compared against (c) RED-SD, which uses all $b = 89$ measurements at each iteration. Both algorithms use exactly the same measurement model and the same denoising CNN AWGN. Note how the results of SIMBA are indistinguishable from RED-SD even though the per-iteration complexity of SIMBA is only a fraction of that of RED-SD.

Authors

Ulugbek S. Kamilov (kamilov@wustl.edu) received his Ph.D. degree in electrical engineering from the Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland, in 2015. He is currently an assistant professor and the director of the Computational Imaging Group, Washington University, St. Louis, MO 63130 USA. He is the recipient of a National Science Foundation CAREER Award and the IEEE Signal Processing Society's 2017 Best Paper Award. His research interests include computational imaging and photography, computer vision, machine learning and optimization. He is a Senior Member of IEEE.

Charles A. Bouman (bouman@purdue.edu) received his Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA. He is currently the Showalter Professor of Electrical and Computer Engineering and Biomedical Engineering, Purdue University, West Lafayette, IN 47907 USA. He is an honorary member of the Society for Imaging Science and Technology and a fellow of the National Academy of Inventors. He is a recipient of the IEEE Signal Processing Society's 2021 Claude Shannon–Harry Nyquist Technical Achievement Award and a corecipient of the first-place award for the 2022 American Association of Physicists in Medicine Truth in Computerized Tomography Reconstruction Challenge. His research interests include computational imaging and sensing using a mix of signal processing, statistical modeling, physics, and computation. He is a Fellow of IEEE.

Gregory T. Buzzard (buzzard@purdue.edu) received his Ph.D. degree in mathematics from the University of Michigan,

Ann Arbor, MI, USA, in 1995. He is a member of the mathematics faculty, Purdue University, West Lafayette, IN 47907 USA. He is a corecipient of the first-place award for the 2022 American Association of Physicists in Medicine Truth in Computerized Tomography Reconstruction Challenge. His research interests include formulation and analysis of methods for computational inverse imaging and optimal sampling. He is a Senior Member of IEEE.

Brendt Wohlberg (brendt@ieee.org) received his Ph.D. degree in electrical engineering from the University of Cape Town, South Africa, in 1996. He is currently a staff scientist in Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545 USA. He was previously Editor-in-Chief of *IEEE Transactions on Computational Imaging* and is currently Editor-in-Chief of *IEEE Open Journal of Signal Processing*. He is a corecipient of the 2020 Society for Industrial and Applied Mathematics Activity Group on Imaging Science Best Paper Prize. His research interests include sparse representations, signal and image processing inverse problems, and computational imaging. He is a Senior Member of IEEE.

References

- [1] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 123–231, Jan. 2014, doi: 10.1561/24000000003.
- [2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011, doi: 10.1561/22000000016.
- [3] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," Purdue Univ., West Lafayette, IN, USA,

ECE Tech. Rep. 448, 2013. [Online]. Available: <http://docs.lib.purdue.edu/ecetr/448>

[4] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007, doi: 10.1109/TIP.2007.901238.

[5] U. S. Kamilov, H. Mansour, and B. Wohlberg, "A plug-and-play priors approach for solving nonlinear imaging inverse problems," *IEEE Signal Process. Lett.*, vol. 24, no. 12, pp. 1872–1876, Dec. 2017, doi: 10.1109/LSP.2017.2763583.

[6] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009, doi: 10.1137/080716542.

[7] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, H. Bauschke, R. Burachik, P. Combettes, V. Elser, D. Luke, and H. Wolkowicz, Eds. New York, NY, USA: Springer Science & Business Media, 2011, p. 18.

[8] C. A. Bouman, *Foundations of Computational Imaging: A Model-Based Approach*. Philadelphia, PA, USA: SIAM, 2022.

[9] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/j.neunet.2014.09.003.

[10] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global Conf. Signal Process. Inf. Process. (GlobalSIP)*, 2013, pp. 945–948, doi: 10.1109/GlobalSIP.2013.6737048.

[11] S. Ono, "Primal-dual plug-and-play image restoration," *IEEE Signal Process. Lett.*, vol. 24, no. 8, pp. 1108–1112, May 2017, doi: 10.1109/LSP.2017.2710233.

[12] W. Gan, Y. Sun, C. Eldeniz, J. Liu, H. An, and U. S. Kamilov, "Deformation-compensated learning for image reconstruction without ground truth," *IEEE Trans. Med. Imag.*, early access, 2022, doi: 10.1109/TMI.2022.3163018.

[13] J. Liu, Y. Sun, C. Eldeniz, W. Gan, H. An, and U. S. Kamilov, "RARE: Image reconstruction using deep priors learned without ground truth," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 6, pp. 1088–1099, May 2020, doi: 10.1109/JSTSP.2020.2998402.

[14] A. M. Teodoro, J. M. Bioucas-Dias, and M. Figueiredo, "A convergent image fusion algorithm using scene-adapted Gaussian-mixture-based denoising," *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 451–463, Jan. 2019, doi: 10.1109/TIP.2018.2869727.

[15] V. Sridhar, X. Wang, G. T. Buzzard, and C. Bouman, "Distributed iterative CT reconstruction using multi-agent consensus equilibrium," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 1153–1166, Jul. 2020, doi: 10.1109/TCI.2020.3008782.

[16] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Trans. Comput. Imag.*, vol. 5, no. 1, pp. 52–67, Mar. 2019, doi: 10.1109/TCI.2018.2880326.

[17] G. T. Buzzard, S. H. Chan, S. Sreehari, and C. A. Bouman, "Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium," *SIAM J. Imag. Sci.*, vol. 11, no. 3, pp. 2001–2020, Sep. 2018, doi: 10.1137/17M1122451.

[18] T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers, "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1799–1808, doi: 10.1109/ICCV.2017.198.

[19] Y. Sun, B. Wohlberg, and U. S. Kamilov, "An online plug-and-play algorithm for regularized image reconstruction," *IEEE Trans. Comput. Imag.*, vol. 5, no. 3, pp. 395–408, Sep. 2019, doi: 10.1109/TCI.2019.2893568.

[20] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 2nd ed. Cham: Springer International Publishing AG, 2017.

[21] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, G. T. Buzzard, L. F. Drummy, J. P. Simmons, and C. A. Bouman, "Plug-and-play priors for bright field electron tomography and sparse interpolation," *IEEE Trans. Comput. Imag.*, vol. 2, no. 4, pp. 408–423, Dec. 2016, doi: 10.1109/TCI.2016.2599778.

[22] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 84–98, Mar. 2017, doi: 10.1109/TCI.2016.2629286.

[23] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, vol. 97, pp. 5546–5557.

[24] Y. Sun, Z. Wu, X. Xu, B. Wohlberg, and U. S. Kamilov, "Scalable plug-and-play ADMM with convergence guarantees," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 849–863, Jul. 2021, doi: 10.1109/TCI.2021.3094062.

[25] Y. Sun, J. Liu, and U. S. Kamilov, "Block coordinate regularization by denoising," in *Proc. 33rd Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2019, pp. 382–392.

[26] X. Xu, Y. Sun, J. Liu, and U. S. Kamilov, "Provable convergence of plug-and-play priors with MMSE denoisers," *IEEE Signal Process. Lett.*, vol. 27, pp. 1280–1284, Jul. 2020, doi: 10.1109/LSP.2020.3006390.

[27] J. Liu, S. Asif, B. Wohlberg, and U. S. Kamilov, "Recovery analysis for plug-and-play priors using the restricted eigenvalue condition," in *Proc. 35th Adv. Neural Inf. Process. Syst.*, Dec. 6–14, 2021, pp. 1–13.

[28] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, Oct. 2017, doi: 10.1137/16M1102884.

[29] J. Zhang and B. Ghanem, "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 1828–1837, doi: 10.1109/CVPR.2018.00196.

[30] Z. Wu, Y. Sun, A. Matlock, J. Liu, L. Tian, and U. S. Kamilov, "SIMBA: Scalable inversion in optical tomography using deep denoising priors," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 6, pp. 1163–1175, Jun. 2020, doi: 10.1109/JSTSP.2020.2999820.

[31] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, Haifa, Israel, Jun. 21–24, 2010, pp. 399–406.

[32] X. Chen, J. Liu, Z. Wang, and W. Yin, "Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds," in *Proc. 31st Adv. Neural Inf. Process. Syst.*, 2018, pp. 9079–9089.

[33] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021, doi: 10.1109/MSP.2020.3016905.

[34] D. Gilton, G. Ongie, and R. Willett, "Deep equilibrium architectures for inverse problems in imaging," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 1123–1133, Oct. 2021, doi: 10.1109/TCI.2021.3118944.

[35] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Barcelona, Spain, Nov. 2011, pp. 479–486, doi: 10.1109/ICCV.2011.6126278.

[36] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2808–2817, doi: 10.1109/CVPR.2017.300.

[37] T. Ttirer and R. Giryes, "Image restoration by iterative denoising and backward projections," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1220–1234, 2019, doi: 10.1109/TIP.2018.2875569.

[38] T. Ttirer and R. Giryes, "Back-projection based fidelity term for ill-posed linear inverse problems," *IEEE Trans. Image Process.*, vol. 29, pp. 6164–6179, Apr. 2020, doi: 10.1109/TIP.2020.2988779.

[39] S. A. Bigdeli, M. Jin, P. Favaro, and M. Zwicker, "Deep mean-shift priors for image restoration," in *Proc. 30th Adv. Neural Inf. Process. Syst.*, Dec. 4–9, 2017, pp. 763–772.

[40] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3563–3593, Jan. 2014.

[41] A. M. Teodoro, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "Image restoration and reconstruction using variable splitting and class-adapted image priors," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 25–28, 2016, pp. 3518–3522.

[42] T. Ttirer and R. Giryes, "Super-resolution via image-adapted denoising CNNs: Incorporating external and internal learning," *IEEE Signal Process. Lett.*, vol. 26, no. 7, pp. 1080–1084, May 2019, doi: 10.1109/LSP.2019.2920250.

[43] A. Gnanasambandam and S. Chan, "One size fits all: Can we train one denoiser for all noise levels?" in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, Jul. 13–18, 2020, vol. 119, pp. 3576–3586.

[44] A. Shocher, N. Cohen, and M. Irani, "Zero-shot super-resolution using deep internal learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 3118–3126, doi: 10.1109/CVPR.2018.00329.

[45] S. A. Hussein, T. Ttirer, and R. Giryes, "Correction filter for single image super-resolution: Robustifying off-the-shelf deep super-resolvers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 1425–1434, doi: 10.1109/CVPR42600.2020.00150.

[46] S. Majee, T. Balke, C. A. J. Kemp, G. T. Buzzard, and C. A. Bouman, "Multislice fusion for sparse-view and limited-angle 4D CT reconstruction," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 448–462, Apr. 2021, doi: 10.1109/TCI.2021.3074881.

[47] E. J. Reid, L. F. Drummy, C. A. Bouman, and G. T. Buzzard, "Multi-resolution data fusion for super resolution imaging," *IEEE Trans. Comput. Imag.*, vol. 8, pp. 81–95, Jan. 2022, doi: 10.1109/TCI.2022.3140551.

[48] C. J. Pellizzari, M. F. Spencer, and C. A. Bouman, "Coherent plug-and-play: Digital holographic imaging through atmospheric turbulence using model-based iterative reconstruction and convolutional neural networks," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 1607–1621, Dec. 2020, doi: 10.1109/TCI.2020.3042948.

[49] T. Balke, F. Davis, C. Garcia-Cardona, M. McCann, L. Pfister, and B. Wohlberg, "Scientific Computational Imaging Code (SCICO)," *J. Open Source Softw.*, vol. 7, no. 78, p. 4722, Oct. 2022, doi: 10.21105/joss.04722.