Distributed Iterative CT Reconstruction Using Multi-Agent Consensus Equilibrium

Venkatesh Sridhar ⓘ, *Student Member, IEEE*, Xiao Wang ⓘ, *Member, IEEE*, Gregery T. Buzzard ⓘ, *Member, IEEE*, and Charles A. Bouman ⓘ, *Fellow, IEEE*

Abstract—**Model-Based Image Reconstruction (MBIR) methods significantly enhance the quality of computed tomographic (CT) reconstructions relative to analytical techniques, but are limited by high computational cost. In this article, we propose a multi-agent consensus equilibrium (MACE) algorithm for distributing both the computation and memory of MBIR reconstruction across a large number of parallel nodes. In MACE, each node stores only a sparse subset of views and a small portion of the system matrix, and each parallel node performs a local sparse-view reconstruction, which based on repeated feedback from other nodes, converges to the global optimum. Our distributed approach can also incorporate advanced denoisers as priors to enhance reconstruction quality. In this case, we obtain a parallel solution to the serial framework of Plug-n-play (PnP) priors, which we call MACE-PnP. In order to make MACE practical, we introduce a partial update method that eliminates nested iterations and prove that it converges to the same global solution. Finally, we validate our approach on a distributed memory system with real CT data. We also demonstrate an implementation of our approach on a massive supercomputer that can perform large-scale reconstruction in real-time.**

Index Terms—**CT reconstruction, MBIR, multi-agent consensus equilibrium, MACE, Plug and play.**

I. INTRODUCTION

TOMOGRAPHIC reconstruction algorithms can be roughly divided into two categories: analytical reconstruction methods [1], [2] and regularized iterative reconstruction methods [3] such as model-based iterative reconstruction (MBIR) [4]–[7]. MBIR methods have the advantage that they can improve reconstructed image quality particularly when projection data are sparse and/or the X-ray dosage is low. This is because MBIR integrates a model of both the sensor and object being imaged into the reconstruction process [4], [6]–[8].

However, the high computational cost of MBIR often makes it less suitable for solving large reconstruction problems in real-time.

One approach to speeding MBIR is to precompute and store the system matrix [9]–[11]. These algorithms can dramatically speedup MBIR computation by reorganizing the elements of a precomputed system matrix in a way that improves memory bandwidth, cache efficiency and SIMD vector parallelization. In fact, the system matrix is typically precomputed in applications such as scientific imaging, non-destructive evaluation (NDE), and security scanning where the system geometry does not vary from scan to scan; so these applications can benefit from the new faster algorithms. However, for large tomographic problems, the system matrix may become too large to store on a single compute node. Therefore, there is a need for iterative reconstruction algorithms that can distribute the system matrix across many nodes in a large cluster.

More recently, advanced prior methods have been introduced into MBIR which can substantially improve reconstruction quality by incorporating machine learning approaches. For example, Plug-n-play (PnP) priors [7], [8], consensus equilibrium (CE) [12], and RED [13] allow convolutional neural networks (CNN) to be used in prior modeling. Therefore, methods that distribute tomographic reconstruction across large compute clusters should also be designed to support these emerging approaches.

In order to make MBIR methods useful, it is critical to parallelize the algorithms for fast execution. Broadly speaking, parallel algorithms for MBIR fall into two categories: fine-grain parallelization methods that are most suitable for shared-memory (SM) implementation [9], [10], [14], [15], and coarse-grain parallelization methods that are most suitable for distributed-memory (DM) implementation [16]–[18]. So for example, SM methods are best suited for implementation on a single multi-core CPU processor or a GPU, while DM methods are better suited for computation across a large cluster of compute nodes. Further, DM methods ensure that the overhead incurred due to inter-node communication and synchronization does not dominate the computation.

In particular, some DM parallel methods can handle large-scale tomographic problems by distributing the system-matrix across multiple nodes, while others do not. For example, the DM algorithm of Wang *et al.* [16] parallelizes the reconstruction across multiple nodes, but it requires that each node have a complete local copy of the system matrix. Alternatively, the DM algorithms of Linyuan *et al.* [17] and Cui *et al.* [18] could
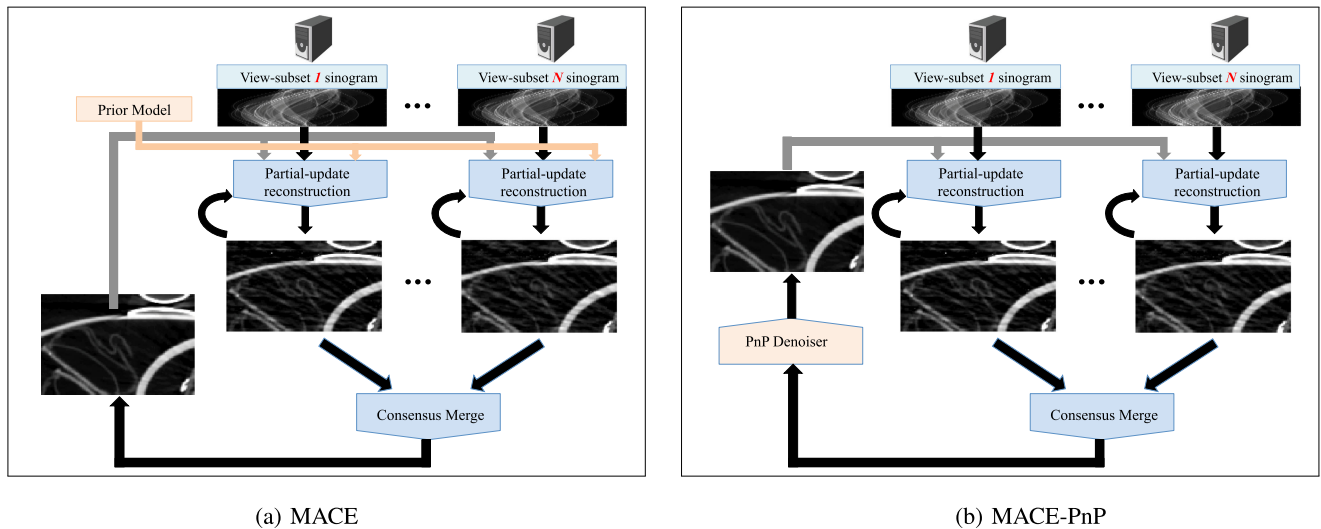
(a) MACE

(b) MACE-PnP

Fig. 1.   Illustration of the MACE algorithm for distributing CT reconstruction across a parallel cluster of compute nodes. (a) An illustration of the MACE algorithm using a conventional prior model. The MACE algorithm works by splitting the data into view subsets and reconstructing them in parallel. The individual reconstructions are them merged in an iterative loop that results in the true MAP reconstruction for the full data set. (b) An illustration of the MACE-PnP algorithm which extends the MACE framework to the use of Plug-n-Play prior models to improve reconstruction quality. In this case, a denoiser is run in the MACE loop in order to implement an advanced prior model. The MACE and MACE-PnP algorithms distribute both computation and memory across parallel clusters of computers, thereby enabling the reconstruction of large tomographic data sets.

potentially be used to parallelize reconstruction across a cluster while distributing the system matrix across the nodes. However, the method of Linyuan [17] is restricted to the use of a total variation (TV) prior [19], [20] and in experiments has required 100 s of iterations for convergence, which is not practical for large problems. Alternatively, the method of Cui [18] is for use in unregularized PET reconstruction.

It is worth noting that DM methods that distribute the system-matrix can also be extended to applications that compute the system-matrix on-the-fly, which is typically the case in medical scanners where the geometry may vary continuously based on scan parameters. However, on-the-fly system matrix computation typically both increases the complexity and reduces the speed of reconstruction algorithms.

In this paper, we build on our previous work in [21], [22] and present a Multi-agent Consensus Equilibrium (MACE) reconstruction algorithm that distributes both the computation and memory of iterative CT reconstruction across a large number of parallel nodes. The MACE approach uses the consensus equilibrium [12] framework to break the reconstruction problem into a set of subproblems which can be solved separately and then integrated together to achieve a high-quality reconstruction. By distributing computation over a set of compute nodes, MACE enables the solution of reconstruction problems that would otherwise be too large to solve.

Fig. 1 illustrates our two approaches to this distributed CT reconstruction problem. While both the approaches integrate multiple sparse-view reconstructions across a compute cluster into a high-quality reconstruction, they differ based on how the prior model is implemented. Fig. 1(a) depicts our basic MACE approach that utilizes conventional edge-preserving regularization [4], [23] as a prior model and converges to the

maximum-a-posteriori (MAP) estimate. Fig. 1(b) shows our second approach called MACE-PnP which allows for distributed CT reconstruction using plug-and-play (PnP) priors [7], [8]. These PnP priors substantially improve reconstructed image quality by implementing the prior model using a denoising algorithm based on methods such as BM3D [24] or deep residual CNNs [25]. We prove that MACE-PnP provides a parallel algorithm for computing the standard serial PnP reconstruction of [7].

A direct implementation of MACE is not practical because it requires repeated application of proximal operators that are themselves iterative. In order to overcome this problem, we introduce the concept of *partial updates*, a general approach for replacing any proximal operator with a non-iterative update. We also prove the convergence of this method for our application.

Our experiments are divided into two parts and are based on real CT datasets from synchrotron imaging and security scanning. In the first part, we use the MACE algorithm to parallelize 2D CT reconstructions across a distributed CPU cluster of 16 compute nodes. We show that MACE both speeds up reconstruction while drastically reducing the memory footprint of the system matrix on each node. We incorporate regularization in the form of either conventional priors such as Q-GGMRF, or alternatively, advanced denoisers such as BM3D that improve reconstruction quality. In the former case, we verify that our approach converges to the Bayesian estimate [4], [6], while in the latter case, we verify convergence to the PnP solution of [7], [12].

In the second part of our experiments, we demonstrate an implementation of MACE on a large-scale supercomputer that can reconstruct a large 3D CT dataset. For this problem, the MACE algorithm is used in conjunction with the super-voxel ICD (SV-ICD) algorithm [9] to distribute computation over 1200 compute nodes, consisting of a total of 81,600 cores.

Importantly, in this case the MACE algorithm not only speeds reconstruction, but it also enables reconstruction for a problem that would otherwise be impossible in real-time since the full system matrix is too large to store on a single node.

II. DISTRIBUTED CT RECONSTRUCTION USING CONVENTIONAL PRIORS

A. CT Reconstruction Using MAP Estimation

We formulate the CT reconstruction problem using the maximum-a-posteriori (MAP) estimate given by [26]

$$x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} f(x; \beta) \qquad (1)$$

where f is the MAP cost function defined by

$$f(x; \beta) = -\log p(y|x) - \log p(x) + const$$

y is the preprocessed projection data, $x \in \mathbb{R}^n$ is the unknown image of attenuation coefficients to be reconstructed, and $const$ represents any possible additive constant.

For our specific problem, we choose the forward model $p(y|x)$ and the prior model $p_\beta(x)$ so that

$$f(x; \beta) = \sum_{k=1}^{N_\theta} \frac{1}{2} \|y_k - A_k x\|_{\Lambda_k}^2 + \beta h(x) \qquad (2)$$

where $y_k \in \mathbb{R}^{N_D}$ denotes the kth view of data, N_θ denotes the number of views, $A_k \in \mathbb{R}^{N_D \times n}$ is the system matrix that represents the forward projection operator for the kth view, and $\Lambda_k \in \mathbb{R}^{N_D \times N_D}$ is a diagonal weight matrix corresponding to the inverse noise variance of each measurement. Also, the last term

$$\beta h(x) = -\log p(x) + const$$

represents the prior model we use where β can be used to control the relative amount of regularization. In this section, we will generally assume that $h(\cdot)$ is convex, so then f will also be convex. A typical choice of h which we will use in the experimental section is the Q-Generalized Gaussian Markov Random Field (Q-GGMRF) prior [27] that preserves both low contrast characteristics as well as edges.

In order to parallelize our problem, we will break up the MAP cost function into a sum of auxiliary functions, with the goal of minimizing these individual cost functions separately. So we will represent the MAP cost function as

$$f(x; \beta) = \sum_{i=1}^{N} f_i(x; \beta)$$

where

$$f_i(x; \beta) = \sum_{k \in J_i} \frac{1}{2} \|y_k - A_k x\|_{\Lambda_k}^2 + \frac{\beta}{N} h(x) \qquad (3)$$

and the view subsets $J_1, \ldots, J_N$ partition the set of all views into N subsets.[1] In this paper, we will generally choose the subsets J_i to index interleaved view subsets, but the theory we develop

works for any partitioning of the views. In the case of interleaved view subsets, the view subsets are defined by

$$J_i = \{ m : m \bmod N = i, \ m \in \{1, \ldots, N_\theta\} \} .$$

B. MACE Framework

In this section, we introduce a framework, which we refer to as multi-agent consensus equilibrium (MACE) [12], for solving our reconstruction problem through the individual minimization of the terms $f_i(x)$ defined in (3).[2] Importantly, minimization of each function, $f_i(\cdot)$, has exactly the form of the MAP CT reconstruction problem but with the sparse set of views indexed by J_i. Therefore, MACE integrates the results of the individual sparse reconstruction operators, or agents, to produce a consistent solution to the full problem.

To do this, we first define the agent, or in this case the proximal map, for the ith auxiliary function as

$$F_i(x) = \operatorname{argmin}_{z \in \mathbb{R}^n} \left\{ f_i(z) + \frac{\|z - x\|^2}{2\sigma^2} \right\} \qquad (4)$$

where σ is a user selectable parameter that will ultimately affect convergence speed of the algorithm. Intuitively, the function $F_i(x)$ takes an input image x, and returns an image that reduces its associated cost function f_i and is close to x.

Our goal will then be to solve the following set of MACE equations

$$F_i(x^* + u_i^*) = x^* \ \text{ for } \ i = 1, \ldots, N \qquad (5)$$

$$\sum_{i=1}^{N} u_i^* = 0 \qquad (6)$$

where $x^* \in \mathbb{R}^n$ has the interpretation of being the consensus solution, and each $u_i^* \in \mathbb{R}^n$ represents the force applied by each agent that balances to zero.

Importantly, the solution x^* to the MACE equations is also the solution to the MAP reconstruction problem of equation (1) (see Theorem 1 of [12]). In order to see this, notice that since f_i is convex we have that

$$\partial f_i(x^*) + \frac{x^* - (x^* + u_i^*)}{\sigma^2} \ni 0, \ i = 1, \ldots, N,$$

where ∂f_i is the sub-gradient of f_i. So by summing over i and applying (6) we have that

$$\partial f(x^*) \ni 0$$

which shows that x^* is a global minimum to the convex MAP cost function f. Conversely, we can also prove that if x^* is a minimum of f, then it is a solution to the MACE equations specified by (5) and (6).

We can represent the MACE equilibrium conditions of (5) and (6) in a more compact notational form. In order to do this, first define the stacked vector

$$\mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} \qquad (7)$$

---

[1] By partition, we mean that $\cup_{i=1}^{N} J_i = \{1, 2, \ldots, N_\theta\}$ and $\cap_{i=1}^{N} J_i = \emptyset$.

[2] In this section, we suppress the dependence on β for notational simplicity.

where each component $v_i \in \mathbb{R}^n$ of the stack is an image. Then we can define a corresponding operator $\mathbf{F}$ that stacks the set of agents as

$$\mathbf{F}(\mathbf{v}) = \begin{pmatrix} F_1(v_1) \\ \vdots \\ F_N(v_N) \end{pmatrix} \qquad (8)$$

where each agent F_i operates on the i-th component of the stacked vector. Finally, we define a new operator $\mathbf{G}(\mathbf{v})$ that computes the average of each component of the stack, and then redistributes the result. More specifically,

$$\mathbf{G}(\mathbf{v}) = \begin{pmatrix} \bar{\mathbf{v}} \\ \vdots \\ \bar{\mathbf{v}} \end{pmatrix}, \qquad (9)$$

where $\bar{\mathbf{v}} = \frac{1}{N} \sum_{i=1}^{N} v_i$.

Using this notation, the MACE equations of (5) and (6) have the much more compact form of

$$\mathbf{F}(\mathbf{v}^*) = \mathbf{G}(\mathbf{v}^*) \qquad (10)$$

with the solution to the MAP reconstruction is given by $x^* = \bar{\mathbf{v}}^*$ where $\bar{\mathbf{v}}^*$ is the average of the stacked components in $\mathbf{v}^*$.

The MACE equations of (10) can be solved in many ways, but one convenient way to solve them is to convert these equations to a fixed-point problem, and then use well known methods for efficiently solving the resulting fixed-point problem [12]. It can be easily shown (see appendix A) that the solution to the MACE equations are exactly the fixed points of the operator $\mathbf{T} = (2\mathbf{F} - \mathbf{I})(2\mathbf{G} - \mathbf{I})$ given by

$$\mathbf{T}\mathbf{w}^* = \mathbf{w}^* \qquad (11)$$

where then $\mathbf{v}^* = (2\mathbf{G} - \mathbf{I})\mathbf{w}^*$.

We can evaluate the fixed-point $\mathbf{w}^*$ using the *Mann* iteration [28]. In this approach, we start with any initial guess and apply the following iteration,

$$\mathbf{w}^{(k+1)} = \rho \mathbf{T}\mathbf{w}^{(k)} + (1 - \rho)\mathbf{w}^{(k)}, k \geq 0 \qquad (12)$$

which can be shown to converge to $\mathbf{w}^*$ for $\rho \in (0, 1)$. The Mann iteration of (12) has guaranteed convergence to a fixed-point $\mathbf{v}^*$ if $\mathbf{T}$ is non-expansive. Both F_i, as well as its reflection, $2F_i - I$, are non-expansive, since each proximal map F_i belongs to a special class of operators called *resolvents* [28]. Also, we can easily show that $2\mathbf{G} - \mathbf{I}$ is non-expansive. Consequently, $\mathbf{T}$ is also non-expansive and (12) is guaranteed to converge.

In practice, $\mathbf{F}$ is a parallel operator that evaluates N agents that can be distributed over N nodes in a cluster. Alternatively, the $\mathbf{G}$ operator has the interpretation of a reduction operation across the cluster followed by a broadcast of the average across the cluster nodes.

C. Partial Update MACE Framework

A direct implementation of the MACE approach specified by (12) is not practical, since it requires a repeated application of proximal operators $F_i, i = 1, \ldots, N$, that are themselves iterative. Consequently, this direct use of (12) involves many

---

Algorithm 1: Partial-Update MACE With Conventional Priors.

1: *Initialize:*
2: $\quad \mathbf{w}^{(0)} \leftarrow$ any value $\in \mathbb{R}^{nN}$
3: $\quad \mathbf{X}^{(0)} = \mathbf{G}(\mathbf{w}^{(0)})$
4: $\quad k \leftarrow 0$
5: **while** not converged **do**
6: $\quad\quad \mathbf{v}^{(k)} = (2\mathbf{G} - \mathbf{I})\mathbf{w}^{(k)}$
7: $\quad\quad \mathbf{X}^{(k+1)} = \tilde{\mathbf{F}}(\mathbf{v}^{(k)}; \sigma, \mathbf{X}^{(k)})$ $\quad \triangleright$Approximate $\mathbf{F}(\mathbf{v}^{(k)})$
8: $\quad\quad \mathbf{w}^{(k+1)} = 2\mathbf{X}^{(k+1)} - \mathbf{v}^{(k)}$ $\triangleright \approx (2\mathbf{F} - \mathbf{I})(2\mathbf{G} - \mathbf{I})\mathbf{w}^{(k)}$
9: $\quad\quad \mathbf{w}^{(k+1)} \leftarrow \rho \mathbf{w}^{(k+1)} + (1 - \rho)\mathbf{w}^{(k)}$ $\quad \triangleright$Mann update
10: $\quad\quad k \leftarrow k + 1$
11: **end while**
12: *Solution:*
13: $\quad x^* = \bar{\mathbf{w}}^{(k)}$ $\quad\quad\quad\quad\quad\quad\quad\quad \triangleright$Consensus solution

---

nested loops of intense iterative optimization, resulting in an impractically slow algorithm.

In order to overcome the above limitation, we propose a *partial-update MACE* algorithm that permits a faster implementation without affecting convergence. In partial-update MACE, we replace each proximal operator with a fast non-iterative update in which we partially evaluate the proximal map, F_i, using only a single pass of iterative optimization.

Importantly, a partial computation of $F_i(\cdot \, \sigma)$ resulting from a single pass of iterative optimization will be dependent on the initial state. So, we use the notation $\tilde{F}_i(\cdot \, \sigma, X_i)$ to represent the partial-update for $F_i(\cdot \, \sigma)$, where $X_i \in \mathbb{R}^n$ specifies the initial state. Analogous to (8), $\tilde{\mathbf{F}}(\mathbf{v}; \sigma, \mathbf{X})$ then denotes the partial update for stacked operator $\mathbf{F}(\mathbf{v}; \sigma)$ from an initial state $\mathbf{X} \in \mathbb{R}^{nN}$.

Algorithm 1 provides the pseudo-code for the Partial-update MACE approach. Note that this algorithm strongly resembles equation (12) that evaluates the fixed-point of map $(2\mathbf{F} - \mathbf{I})(2\mathbf{G} - \mathbf{I})$, except that $\mathbf{F}$ is replaced with its partial update as shown in line 7 of the pseudo-code. Note that this framework allows a single pass of any optimization technique to compute the partial update. In this paper, we will use a single pass of the the Iterative Coordinate Descent (ICD) optimization method [6], [9], that greedily updates each voxel, but single iterations of other optimization methods can also be used.

Intuitively, step 1 of the above algorithm computes the average $\bar{\mathbf{w}}$ across N subsets and re-distributes it. Step 7 parallelizes the reconstruction across view-subsets using non-iterative proximal operator $\tilde{\mathbf{F}}$. A combination of steps 8 and 9 gives the Mann update for map $\mathbf{T}$.

In order to understand the convergence of the partial-update algorithm, we can formulate the following update with an augmented state as

$$\begin{bmatrix} \mathbf{w}^{(k+1)} \\ X^{(k+1)} \end{bmatrix} = \begin{bmatrix} \rho & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2\tilde{\mathbf{F}}\left(\mathbf{v}^{(k)}; X^{(k)}\right) - \mathbf{v}^{(k)} \\ \tilde{\mathbf{F}}\left(\mathbf{v}^{(k)}; X^{(k)}\right) \end{bmatrix}$$
$$+ \begin{bmatrix} 1 - \rho & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}^{(k)} \\ X^{(k)} \end{bmatrix}, \qquad (13)$$

Algorithm 2: ICD-based Partial-update for proximal operation $F_i(v)$ using an initial state x

---

1: Define $\varepsilon_s = [0, \ldots, 1, \ldots, 0]^t \in \mathbb{R}^n$ ▷entry 1 at s-th position
2: $z = x$ ▷Copy initial state
3: **for** $s = 1$ to n **do**
4:    $\alpha_s = \operatorname{argmin}_\alpha \{ f_i(z + \alpha\varepsilon_s) + \frac{1}{2\sigma^2}\|z + \alpha\varepsilon_s - v\|^2 \}$
5:    $z \leftarrow z + \alpha_s\varepsilon_s$
6: **end for**
7: $\tilde{F}_i(v; x) = z$ ▷Partial update

---

where $\mathbf{v}^{(k)} = (2\mathbf{G} - \mathbf{I})\mathbf{w}^{(k)}$. We specify $\tilde{F}$ more precisely in Algorithm 2. In Theorem II.1, we show that any fixed-point $(\mathbf{w}^*, \mathbf{X}^*)$ of this partial-update algorithm is a solution to the exact MACE method of (11). Further, in Theorem II.2 we show that for the specific case where f_i defined in (3) is strictly quadratic, the partial-update algorithm has guaranteed convergence to a fixed-point.

Theorem II.1: Let $f_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, N$ be a strictly convex and differentiable function. Let F_i denote the proximal map of f_i. Let $\tilde{F}_i(v; x)$ denote the partial update for $F_i(v)$ as specified by Algorithm 2. Then any fixed-point $(\mathbf{w}^*, \mathbf{X}^*)$ of the Partial-update MACE approach represented by (13) is a solution to the exact MACE approach specified by (11).

Proof: Proof is in Appendix B. ∎

Theorem II.2: Let B be a positive definite $n \times n$ matrix, and let $f_i : \mathbb{R}^n \to \mathbb{R}^n, i = 1, \ldots, N$ each be given by

$$f_i(x) = \sum_{k \in J_i} \frac{1}{2}\|y_k - A_k x\|_{\Lambda_k}^2 + \frac{\beta}{N} x^T B x.$$

Let F_i denote the proximal map of f_i. Let $\tilde{F}_i(v; x, \sigma)$, $v, x \in \mathbb{R}^n$, denote the partial update for proximal operation $F_i(v; \sigma)$ as shown in Algorithm 2. Then equation (13) can be represented by a linear transform

$$\begin{bmatrix} \mathbf{w}^{(k+1)} \\ X^{(k+1)} \end{bmatrix} = \mathbf{M}_{\sigma^2} \begin{bmatrix} \mathbf{w}^{(k)} \\ X^{(k)} \end{bmatrix} + \mathbf{c}(\mathbf{y}, \mathbf{A}, \rho),$$

where $\mathbf{M}_{\sigma^2} \in \mathbb{R}^{2Nn \times 2Nn}$ and $\mathbf{c} \in \mathbb{R}^{2Nn}$. Also, for sufficiently small $\sigma > 0$, any eigenvalue of the matrix $\mathbf{M}_{\sigma^2}$ is in the range $(0,1)$ and hence the iterates defined by (13) converge in the limit $k \to \infty$ to $(\mathbf{w}^*, \mathbf{X}^*)$, the solution to the exact MACE approach specified by (11).

Proof: Proof is in Appendix B. ∎

Consequently, in the specific case of strictly convex and quadratic problems, the result of Theorem II.2 shows that despite the partial-update approximation to the MACE approach, we converge to the exact consensus solution. In practice, we use non-Gaussian MRF prior models for their edge-preserving capabilities, and so the global reconstruction problem is convex but not quadratic. However, when the priors are differentiable, they are generically locally well-approximated by quadratics, and our experiments show that we still converge to the exact solution even in such cases.

III. MACE WITH PLUG-AND-PLAY PRIORS

In this section, we generalize our approach to incorporate *Plug-n-Play* (PnP) priors implemented with advanced denoisers [7]. Since we will be incorporating the prior as a denoiser, for this section we drop the prior terms of in equation (2) by setting $\beta = 0$. So let $f(x) = f(x; \beta = 0)$ denote the CT log likelihood function of (2) with $\beta = 0$ and no prior term, and let $F(x)$ denote its corresponding proximal map. Then Buzzard *et al.* in [12] show that the PnP framework of [7] can be specified by the following equilibrium conditions

$$F(x^* - \alpha^*; \sigma) = x^* \tag{14}$$

$$H(x^* + \alpha^*) = x^*, \tag{15}$$

where $H : \mathbb{R}^n \to \mathbb{R}^n$ is the plug-n-play denoiser used in place of a prior model. This framework supports a wide variety of denoisers including BM3D and residual CNNs that can be used to improve reconstruction quality as compared to conventional prior models [7], [8].

Let $f_i(x) = f_i(x; \beta = 0)$ to be the log likelihood terms from (3) corresponding to the sparse view subsets, and let $F_i(x)$ be their corresponding proximal maps. Then in Appendix C we show that the PnP result specified by (14) and (15) is equivalent to the following set of equilibrium conditions.

$$F_i(x^* + u_i^*; \sigma) = x^*, \ i = 1, \ldots, N, \tag{16}$$

$$H(x^* + \alpha^*) = x^*, \tag{17}$$

$$\sum_{i=1}^N u_i^* + \alpha^* = 0. \tag{18}$$

Again, we can solve this set of balance equations by transforming into a fixed point problem. One approach to solving equations (16)–(18) is to add an additional agent, $F_{N+1} = H$, and use the approach of Section II-B [29]. However, here we take a slightly different approach in which the denoising agent is applied in series, rather than in parallel.

In order to do this, we first specify a rescaled parallel operator $\mathbf{F}$ and a novel consensus operator $\mathbf{G}_H$, given by

$$\mathbf{F} = \begin{pmatrix} F_1(v_1; \sqrt{N}\sigma) \\ \vdots \\ F_N(v_N; \sqrt{N}\sigma) \end{pmatrix} \text{ and } \mathbf{G}_H(\mathbf{v}) = \begin{pmatrix} H(\bar{\mathbf{v}}) \\ \vdots \\ H(\bar{\mathbf{v}}) \end{pmatrix}, \tag{19}$$

where $\bar{\mathbf{v}} = \sum_{i=1}^N v_i / N$.

In Theorem III.1 below we show that we can solve the equilibrium conditions of (16)–(18) by finding the fixed-point of the map $\mathbf{T}_H = (2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})$. In practice, we can implement the $\mathbf{G}_H$ first computing an average across a distributed cluster, then applying our denoising algorithm, followed by broadcasting back a denoised, artifact-free version of the average.

Theorem III.1: Let $F_i, i = 1, \ldots, N$, denote the proximal map of function f_i. Let maps $\mathbf{F}$ and $\mathbf{G}_H$ be defined by (19). Let $\hat{x}^*$ denote N vertical copies of x^*. Then, $(x^*, \mathbf{u}^*) \in \mathbb{R}^n \times \mathbb{R}^{nN}$ is a solution to the equilibrium conditions of (16)–(18) if and only if the point $\mathbf{w}^* = \hat{x}^* - N\mathbf{u}^*$ is a fixed-point of map $\mathbf{T}_H = (2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})$ and $\mathbf{G}_H(\mathbf{w}^*) = \hat{x}^*$.

Proof: Proof is in Appendix C. ∎

Algorithm 3: Partial-Update MACE With PnP Priors.

1:    *Initialize:*
2:    $\mathbf{w}^{(0)} \leftarrow$ any value $\in \mathbb{R}^{nN}$
3:    $\mathbf{X}^{(0)} = \mathbf{G}_H(\mathbf{w}^{(0)})$
4:    $k \leftarrow 0$
5:    **while** not converged **do**
6:       $\mathbf{v}^{(k)} = (2\mathbf{G}_H - \mathbf{I})\mathbf{w}^{(k)}$
7:       $\mathbf{X}^{(k+1)} = \tilde{\mathbf{F}}(\mathbf{v}^{(k)}; \sigma, \mathbf{X}^{(k)})$ ▷Approximate $\mathbf{F}(\mathbf{v}^{(k)})$
8:       $\mathbf{w}^{(k+1)} = 2\mathbf{X}^{(k+1)} - \mathbf{v}^{(k)}$ ▷$\approx(2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})\mathbf{w}^{(k)}$
9:       $\mathbf{w}^{(k+1)} \leftarrow \rho\mathbf{w}^{(k+1)} + (1 - \rho)\mathbf{w}^{(k)}$ ▷Mann update
10:      $k \leftarrow k + 1$
11:   **end while**
12:   *Solution:*
13:   $x^* = H\bar{\mathbf{w}}^{(k)}$ ▷Consensus solution

When $\mathbf{T}_H$ is non-expansive, we can again compute the fixed-point $\mathbf{w}^* \in \mathbb{R}^{nN}$ using the Mann iteration

$$\mathbf{w}^{(k+1)} = \rho(2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})\mathbf{w}^{(k)} + (1 - \rho)\mathbf{w}^{(k)}. \quad (20)$$

Then, we can compute the x^* that solves the MACE conditions (16)–(18), or equivalently, the PnP conditions (14)–(15), as $x^* = H\bar{\mathbf{w}}^*$. Importantly, (20) provides a parallel approach to solving the PnP framework since the parallel operator $\mathbf{F}$ typically constitutes the bulk of the computation, as compared to consensus operator $\mathbf{G}_H$.

In the specific case when the denoiser H is *firmly non-expansive*, such as a proximal map, we show in Lemma III.2 that $\mathbf{T}_H$ is non-expansive. While there is no such guarantee for any general H, in practice, we have found that this Mann iteration converges. This is consistent with previous experimental results that have empirically observed convergence of PnP [8], [12] for a wide variety of denoisers including BM3D [24], non-local means [30], or Deep residual CNNs [25], [29].

Lemma III.2: If $\mathbf{F}$ and $\mathbf{G}_H$ are defined by (19), and H is firmly non-expansive, then $\mathbf{T}_H = (2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})$ is non-expansive and the Mann iteration of (20) converges to the fixed point of $\mathbf{T}_H$.

Proof: The proof is in Appendix C. ∎

The Algorithm 3 below shows the partial update version of the Mann iteration from equation (20). This version of the algorithm is practical since it only requires a single update of each proximal map per iteration of the algorithm.

In the above algorithm, step 6 applies denoiser H to the average $\bar{\mathbf{w}}$ across the N subsets and re-distributes the denoised result. Step 7 computes the sparse-view reconstruction for each subset independently by using non-iterative proximal updates. Step 8 and 9 complete the evaluation of $\mathbf{T}_H\mathbf{w}$ and perform the Mann update for $\mathbf{T}_H$.

IV. EXPERIMENTAL RESULTS

Our experiments are divided into two parts corresponding to 2D and 3D reconstruction experiments. In each set of experiments, we analyze convergence and determine how the number of view-subsets affects the speedup and parallel-efficiency of the MACE algorithm.

TABLE I
CT DATASET DESCRIPTION

(a) 2-D Datasets

Dataset	#Views	#Channels	Image size
Low-Res. Ceramic Composite (LBNL)	1024	2560	512×512
Baggage Scan (ALERT)	720	1024	512×512

(b) 3-D Dataset

Dataset	#Views	#Channels	Volume size
High-Res. Ceramic Composite (LBNL)	1024	2560	1280×1280×1200

Table I(a) lists parameters of the 2D data sets. The first 2D data set was collected at the Lawrence Berkeley National Laboratory Advanced Light Source synchrotron and is one slice of a scan of a ceramic matrix composite material. The second 2D data set was collected on a GE Imatron multi-slice CT scanner and was reformated into parallel beam geometry. For the 2D experiments, reconstructions are done with a image size of 512×512, and the algorithm is implemented on a distributed compute cluster of 16 CPU nodes using the standard *Message Parsing Interface* (MPI) protocol. Source code for our distributed implementation can be found in [31].

Table I(b) lists parameters of the 3-D parallel-beam CT dataset used for our supercomputing experiment. Notice that for the 3D data set, the reconstructions are computed with an array size of 1280×1280, effectively doubling the resolution of the reconstructed images. This not only increases computation, but makes the system matrix larger, making reconstruction much more challenging. For the 3D experiments, MACE is implemented on the massive NERSC supercomputer using 1200 multi-core CPU nodes belonging to the Intel Xeon Phi Knights Landing architecture, with 68 cores on each node.

A. Methods

For the 2D experiments, we compare our distributed MACE approach against a single-node method. Both the single-node and MACE methods use the same ICD algorithm for reconstruction. In the case of the single-node method, ICD is run on a single compute node that stores the complete set of views and the entire system matrix. Alternatively, for the MACE approach computation and memory is distributed among N compute nodes, with each node performing reconstructions using a subset of views. The MACE approach uses partial updates each consisting of 1 pass of ICD optimization.

We specify the computation in units called *Equits* [32]. In concept, 1 equit is the equivalent computation of 1 full iteration of centralized ICD on a single node. Formally, we define an equit as

$$\# \text{ Equits} = \frac{(\# \text{ of voxel updates})}{(\# \text{ of voxels in ROI})*(\# \text{ of view subsets})}.$$

For the case of a single node, 1 equit is equivalent to 1 full iteration of the centralized ICD algorithm. However, equits can take fractional values since non-homogeneous ICD algorithms can skip pixel updates or update pixels multiple times in a single

iteration. Also notice this definition accounts for the fact that the computation of an iteration is roughly proportional to the number of views being processed by the node. Consequently, on a distributed implementation, 1 equit is equivalent to having each node perform 1 full ICD iteration using its subset of views.

Using this normalized measure of computation, the speedup due to parallelization is given by

$$\text{Speedup(N)} = N \times \frac{(\text{\# of equits for centralized convergence})}{(\text{\# of equits for MACE convergence})}$$

where again N is the number of nodes used in the MACE computation. From this we can see that the speedup is linear in N when the number of equits required for convergence is constant.

In order to measure convergence of the iterative algorithms, we define the NRMSE metric as

$$\text{NRMSE}(x, x^*) = \frac{\|x - x^*\|}{\|x^*\|}.$$

where x^* is the fully converged reconstruction. For the 2-D experiments alone, we compute x^* by running the single-node reconstruction method for 20 equits.

All results using the 3D implemented on the NERSC supercomputer used the highly parallel 3-D Super-voxel ICD (SV-ICD) algorithm described in detail in [9]. The SV-ICD algorithm employees a hierarchy of parallelization and highly efficient data-caching to maximize utilization of each cluster node. More specifically, the 1200 slices in the 3D data set are processed in groups of 8 slices, with each group of 8 slices being processed by a single node. The 68 cores in each node then perform a parallelized version of ICD in which pixels are processed in blocks called super-voxels. However, even with this very high level of parallelism, the SV-ICD algorithm has two major shortcomings. First, it can only utilize 150 nodes in the super-computing cluster, but more importantly, when the system matrix is too large to store on a single node, real-time high resolution 3D reconstruction is impossible without the MACE algorithm.

For all experiments, we initialized each of the images $w_i^{(0)}, i = 1, \ldots, N$ in Algorithms 1 and 3 to be a nearly zero-valued uniform image within the region of interest (ROI). We discuss the selection of parameters ρ and σ in the next subsection.

B. MACE Reconstruction of 2-D CT Dataset

In this section, we study the convergence and parallel efficiency of the 2D data sets of Table I(a). Fig. 2 shows reconstructions of the data sets, and Fig. 3 compares the quality of the centralized and MACE reconstructions for zoomed-in regions of the image. The MACE reconstruction is computed using $N = 16$ compute nodes or equivalently $N = 16$ view-subsets. Notice that the MACE reconstruction is visually indistinguishable from the centralized reconstruction. However, for the MACE reconstruction, each node only stores less than 7% of the full sinogram, dramatically reducing storage requirements.

Table II shows the convergence of MACE for varying values of the parameter ρ, with $N=16$. Notice that for both data sets,
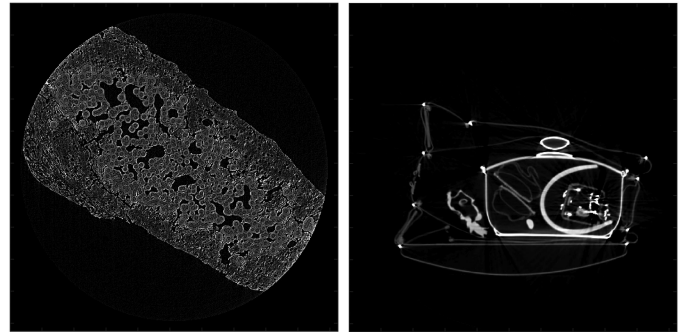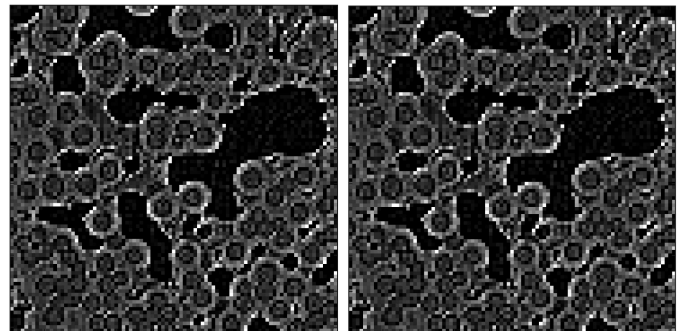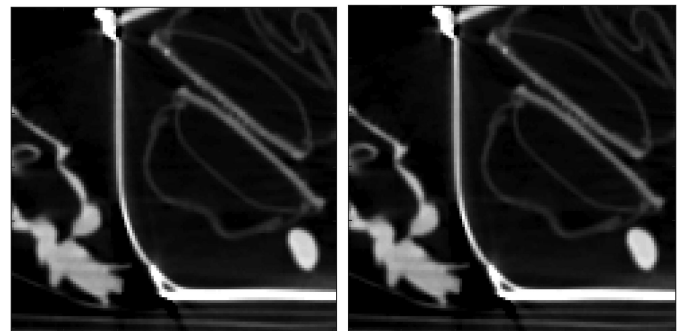

Fig. 2.   Single node reconstruction for (left) Low-Res. Ceramic Composite dataset (right) Baggage Scan dataset.

(a) Single node reconstruction   (b) MACE reconstruction, $N = 16$

(c) Single node reconstruction   (d) MACE reconstruction, $N = 16$

Fig. 3.   Comparison of reconstruction quality for MACE method using 16 parallel nodes each processing $(1/16)^{th}$ of the views, against centralized method. (a) and (c) Centralized method. (b) and (d) MACE. Notice that both methods have equivalent image quality.

TABLE II
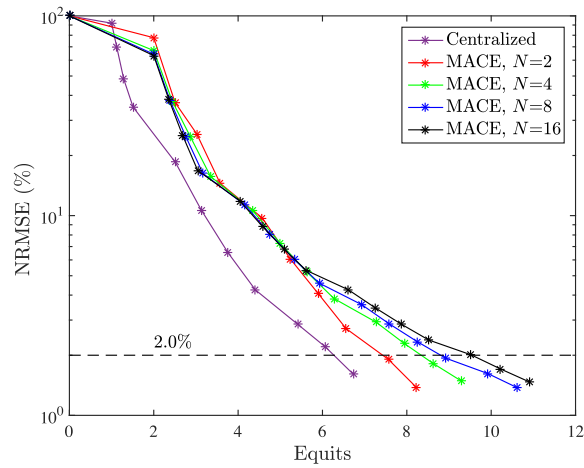MACE CONVERGENCE (EQUITS) FOR DIFFERENT VALUES OF ρ, $N=16$

Dataset	ρ				
	0.5	0.6	0.7	0.8	0.9
Low-Res. Ceramic	18.00	16.00	15.00	**14.00**	14.00
Baggage scan	12.64	10.97	10.27	**9.52**	12.40

$\rho = 0.8$ resulted in the fastest convergence. In fact, in a wide range of experiments, we found that $\rho = 0.8$ was a good choice and typically resulted in the fastest convergence.

Fig.s 4 shows the convergence of MACE using a conventional prior for varying numbers of compute nodes, N. Notice that
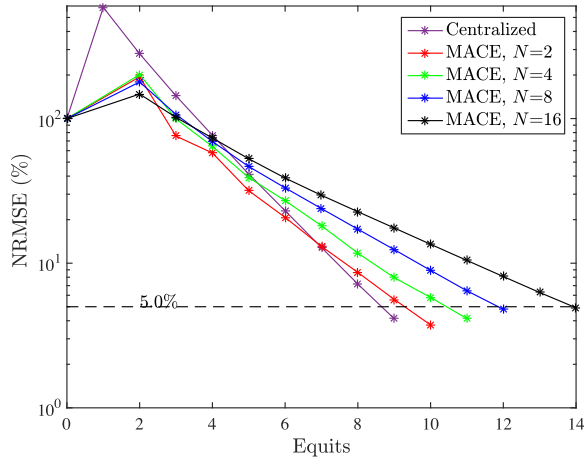
(a) Convergence for different #nodes, Low-Res. Ceramic dataset

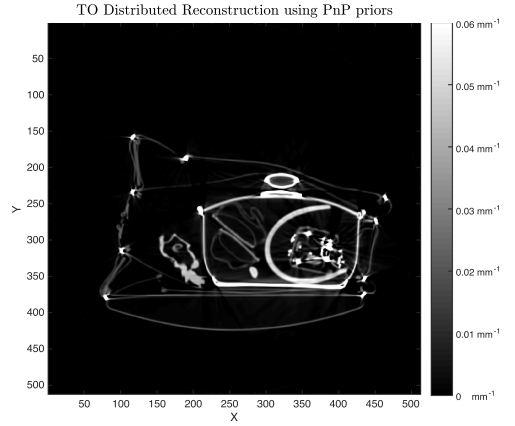(b) Convergence for different #nodes, Baggage Scan dataset

Fig. 4. MACE convergence for different number of nodes, N, using $\rho = 0.8$: (a) Low-Res. Ceramic composite dataset (b) Baggage Scan dataset. Notice that number of equits tends to gradually increase with number of parallel processing nodes, N.

for this case of the conventional QGGMRF prior model, as N increased, the number of equits required for convergence tended to increase for both data sets.

Fig. 5 shows results using the MACE with the PnP prior (MACE-PnP) with the Baggage Scan data set. Notice that the PnP prior results in improved image quality with less streaking and fewer artifacts. Also notice that with the PnP prior, the number of equits required for convergence shown in Fig. 5(c) does not significantly increase with number of compute nodes, N.

Fig. 6 summarizes this important result by plotting the parallel speed up as a function of number of nodes for both data sets using both priors. Notice that the MACE-PnP algorithm results in approximately linear speedup for $N \leq 8$ for both data sets, and near linear speedup up to $N = 16$. We conjecture that the advance prior tends to speed convergence due the stronger regularization constraint it provides.
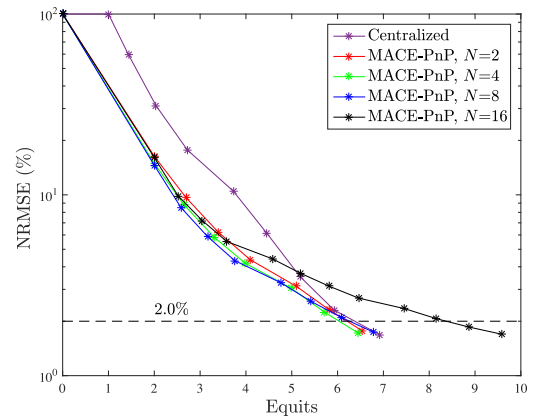
Table III shows the system matrix memory as a function of the number of nodes, N. Note that MACE

(a) 16 node MACE reconstruction with PnP prior
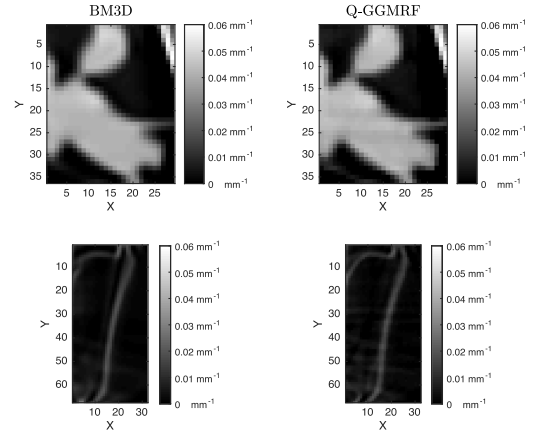
(b) Comparison of PnP prior (left) vs conventional prior (right)

(c) Convergence for different #nodes

Fig. 5. MACE-PnP reconstruction of Baggage Scan data set: (a) MACE PnP reconstruction using $N = 16$ nodes; (b) Zoomed-in regions of PnP versus conventional prior; (c) MACE-PnP convergence as a function of number of nodes N. Notice that PnP prior produces better image quality with reduced streaking and artifacts. In addition, the number of equits required for convergence of the MACE-PnP does not tend to increase significantly with number of nodes N.

drastically reduces the memory usage per node by a factor of N.

Table IV shows our choice of proximal parameter σ which controls convergence rate of MACE, as a function of number of nodes, N. We manually tune σ to achieve optimal convergence rate. Intuitively, a suitable choice of σ must achieve a balance
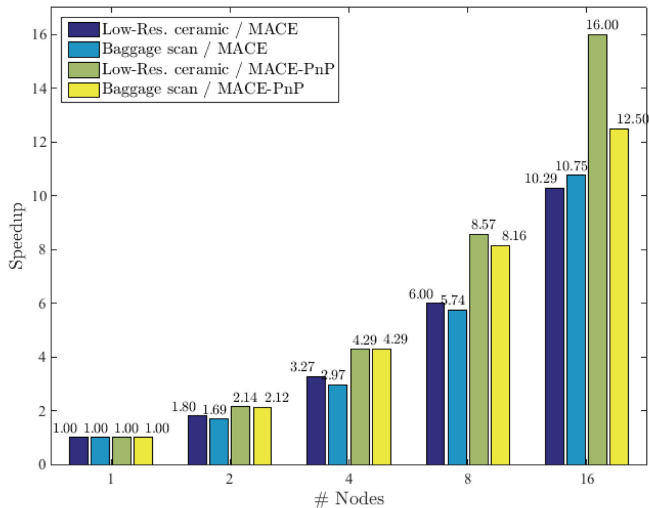
Fig. 6. MACE speedup as a function of the number of nodes, for different datasets and prior models. Importantly, note that in the case of PnP priors, we achieve a *near linear* speedup for both datasets.

TABLE III
MACE MEMORY USAGE FOR THE SYSTEM-MATRIX (GIGABYTES) AS A FUNCTION OF NUMBER OF NODES, N [1]

Dataset	N				
	1	2	4	8	16
Low-Res. Ceramic	14.83	7.42	3.71	1.86	0.93
Baggage scan	5.04	2.52	1.26	0.63	0.32

[1] System-matrix represented in sparse matrix format and floating-point precision

TABLE IV
MACE PARAMETER σ (UNITS MM^{-1}) AS A FUNCTION OF NUMBER OF NODES N [1]

| Dataset	N			
	2	4	8	16
Low-Res. Ceramic	9.0	12.0	14.0	16.0
Baggage scan	0.08	0.10	0.12	0.15

[1] σ affects speed of convergence but not the converged result

between the data-model term $f_i(\cdot)$ and the quadratic penalty term within the proximal map F_i defined by (4).

C. MACE Reconstruction for Large 3D Datasets

In this section, we study the convergence and parallel efficiency of the 3D data sets of Table I(b) using the MACE algorithm implemented on the NERSC supercomputer.

All cases use the SV-ICD algorithm for computation of the reconstructions at individual nodes with a Q-GGMRF as prior model and $\rho = 0.8$ as value of Mann parameter. As noted previously, for this case the system matrix is so large that it is not possible to compute the reconstruction with $N = 1$ view subset. So in order to produce our reference reconstruction, we ran 40 equits of MACE with $N = 2$ view subsets, which appeared to achieve full convergence.
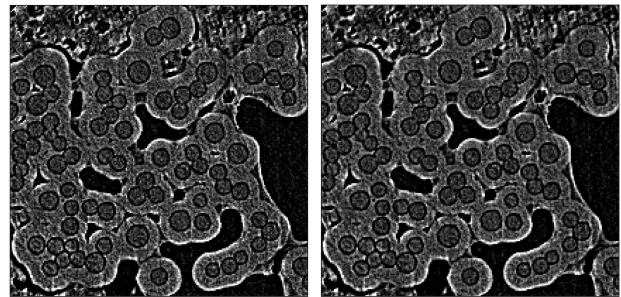
(a) Fully converged   (b) MACE Reconstruction, $N = 8$

Fig. 7. Comparison of quality between (a) fully converged result and (b) the MACE reconstruction using 8 view-subsets on the NERSC supercomputer. Notice that both have equivalent image quality.
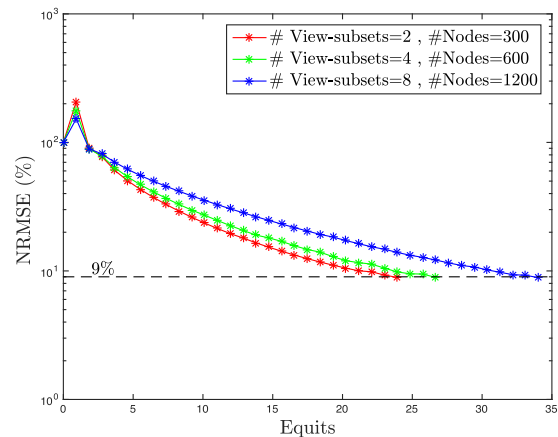

Fig. 8. MACE convergence for the High-Res. ceramic reconstruction on the NERSC supercomputer as a function of view-subsets (reconstruction size $1280 \times 1280 \times 1200$).

TABLE V
COMPUTATIONAL PERFORMANCE FOR 3-D MACE RECONSTRUCTION ON THE NERSC SUPERCOMPUTER AS A FUNCTION OF #VIEW-SUBSETS

#View-subsets	1	2	4	8
#Nodes	150	300	600	1200
#Cores	10,200	20,400	40,800	81,600
Memory usage [2](GB)	-	25.12	12.56	6.28
#Equits	-	23.91	26.67	34.03
#Time (s)	-	275	154	121
MACE Algorithmic Speedup	-	2	3.58	5.62
Machine-time Speedup	-	2	3.57	4.55

[2] rough estimate when system-matrix is represented in conventional sparse matrix format, floating-point precision

Fig. 7 compares zoomed in regions of the fully converged result with MACE reconstructions using $N = 8$ view subsets. Notice both have equivalent image quality.

Fig. 8 shows the convergence of MACE as a function of number of equits. Notice that, as in the 2D case using the Q-GGMRF prior, the number of equits tends to increase as the number of view subsets, N, increases.

Table V summarizes the results of the experiment as a function of the number of view subsets, N. Notice that the memory

requirements for reconstruction drop rapidly with the number of view subsets. This makes parallel reconstruction practical for large tomographic data sets. In fact, the results are not listed for the case of $N = 1$, since the system matrix is too large to store on the nodes of the NERSC supercomputer. Also, notice that parallel efficiency is good up until $N = 4$, but it drops off with $N = 8$.

V. CONCLUSION

In this paper, we proposed a novel MACE algorithm that distributes both the memory and computation of iterative CT reconstruction across a large number of parallel nodes. MACE integrates multiple sparse-view reconstructions across the compute cluster into a high-quality reconstruction. Further, we introduced a variant of MACE called MACE-PnP that supports the use of advanced PnP denoisers that improve reconstruction quality. We proved that MACE converges to the MAP estimate, while MACE-PnP converges to the solution of the serial PnP framework. In order to make MACE feasible for practical applications, we proposed a partial-update implementation that utilizes non-iterative proximal updates and proved its convergence. We analyzed the convergence and speedup of our method on a distributed memory system with real CT data. Further, we demonstrated real-time reconstruction from large CT datasets on a massive supercomputer using MACE.

APPENDIX A

We show that the MACE equations of (10) can be formulated as a fixed-point problem represented by (11). For a more detailed explanation see Corollary 3 of [12].

Proof: A simple calculation shows that for any $\mathbf{v} \in \mathbb{R}^{nN}$, operator $\mathbf{G}$ defined in (9) follows

$$\mathbf{GGv} = \mathbf{Gv}, \text{ and so } (2\mathbf{G} - \mathbf{I})(2\mathbf{G} - \mathbf{I})\mathbf{v} = \mathbf{v}.$$

Thus $2\mathbf{G} - \mathbf{I}$ is self-inverse. We define $\mathbf{w}^*$ as

$$\mathbf{w}^* = (2\mathbf{G} - \mathbf{I})\mathbf{v}^*,$$

in which case $\mathbf{v}^* = (2\mathbf{G} - \mathbf{I})\mathbf{w}^*$ due to the above self-inverse property. Additionally, (10) gives

$$(2\mathbf{F} - \mathbf{I})\mathbf{v}^* = (2\mathbf{G} - \mathbf{I})\mathbf{v}^*.$$

Note that the RHS of the above is merely $\mathbf{w}^*$. So, plugging $\mathbf{v}^*$ in terms of $\mathbf{w}^*$ on the LHS, we get

$$(2\mathbf{F} - \mathbf{I})(2\mathbf{G} - \mathbf{I})\mathbf{w}^* = \mathbf{w}^*.$$

Hence $\mathbf{w}^*$ fixed-point of a specific map $\mathbf{T} : \mathbb{R}^{nN} \to \mathbb{R}^{nN}$, where $\mathbf{T} = (2\mathbf{F} - \mathbf{I})(2\mathbf{G} - \mathbf{I})$. Finding $\mathbf{w}^*$ gives us $\mathbf{v}^*$, since $\mathbf{v}^* = (2\mathbf{G} - \mathbf{I})\mathbf{w}^*$. ∎

APPENDIX B

Lemma B.1: Let $F_i : \mathbb{R}^n \to \mathbb{R}^n$, denote the proximal map of a strictly convex and continuously differentiable function $f_i : \mathbb{R}^n \to \mathbb{R}$. Let $\varepsilon_s \in \mathbb{R}^n$ be defined as $\varepsilon_s = [0, \ldots, 1, \ldots, 0]^t$ where entry 1 is at s-th index. Let $\tilde{F}_i(v; x)$ denote the partial-update for $F_i(v)$ from an initial state x as shown in Algorithm 2. Then $F_i(v) = x$ if and only if $\tilde{F}_i(v; x) = x$.

Proof: We first assume $\tilde{F}_i(v; x) = x$. Since f_i is strictly convex and continuously differentiable, line 4 of Algorithm 2 can be re-written as

$$\alpha_s = \left\{ \alpha \mid \varepsilon_s^t \left[\nabla f_i(z + \alpha\varepsilon_s) + \frac{1}{\sigma^2}(z + \alpha\varepsilon_s - v) \right] = 0 \right\}. \tag{21}$$

Since $\tilde{F}_i(v; x) = x$, from line 5 of Algorithm 2 and the fact that $\varepsilon_s, s = 1, \ldots, n$ are independent, it follows that $\alpha_s = 0, s = 1, \ldots, n$. Applying $\alpha_s = 0$ repeatedly to lines 4-5 of Algorithm 2 and using (21), we get

$$\frac{\partial f_i(x)}{\partial x_s} + \frac{1}{\sigma^2}(x_s - v_s) = 0, \quad s = 1, \ldots, n.$$

Stacking the above result vertically, we get

$$\nabla f_i(x) + \frac{1}{\sigma^2}(x - v) = 0.$$

Since f_i is strictly convex and continuously differentiable the above gives

$$x = \operatorname{argmin}_z \left\{ f_i(z) + \frac{\|z - v\|^2}{2\sigma^2} \right\}$$

and so, $x = F_i(v)$. Therefore, $\tilde{F}_i(v; x) = x$ gives $F_i(v) = x$.

The converse can be proved by reversing the above steps. ∎

Proof: Proof of Theorem II.1

Assume Partial-update MACE algorithm has a fixed-point $(\mathbf{w}^*, \mathbf{X}^*)$. Then from (13) we get,

$$\mathbf{X}^* = \tilde{\mathbf{F}}(\mathbf{v}^*; \mathbf{X}^*, \sigma) \text{ and} \tag{22}$$

$$\mathbf{w}^* = 2\mathbf{X}^* - \mathbf{v}^*, \tag{23}$$

where $\mathbf{v}^* = (2\mathbf{G} - \mathbf{I})\mathbf{w}^*$. So, (23) can be re-written as

$$\mathbf{w}^* = 2\mathbf{X}^* - (2\mathbf{G} - \mathbf{I})\mathbf{w}^*, \text{ which gives}$$

$$\mathbf{X}^* = \mathbf{Gw}^*.$$

So, $X_i^* = \bar{\mathbf{w}}^*$ for $i = 1, \ldots, N$, and consequently, (22) can be expressed as

$$\bar{\mathbf{w}}^* = \tilde{F}_i(v_i^* \, \bar{\mathbf{w}}^*, \sigma), \; i = 1, \ldots N.$$

Applying Lemma VI.1 to the above we get

$$\bar{\mathbf{w}}^* = F_i(v_i^*; \sigma), i = 1, \ldots, N.$$

By stacking the above result vertically, we get

$$\mathbf{Gw}^* = \mathbf{F}(\mathbf{v}^*).$$

Based on definition of $\mathbf{v}^*$, the above gives

$$\mathbf{Gw}^* = \mathbf{F}(2\mathbf{G} - \mathbf{I})\mathbf{w}^*.$$

Multiplying both LHS and RHS by 2 and further subtracting $\mathbf{w}^*$ from both sides, we get

$$(2\mathbf{F} - \mathbf{I})(2\mathbf{G} - \mathbf{I})\mathbf{w}^* = \mathbf{w}^*.$$

Therefore, any fixed-point of the Partial-update MACE algorithm, $(\mathbf{w}^*, \mathbf{X}^*)$, is a solution to the exact MACE approach specified by (11). ∎

Proof: Proof of Theorem II.2 (convergence of the Partial-update MACE algorithm)

We can express the function f_i defined in Theorem II.2 more compactly. For this purpose, let subset J_i be represented as $J_i = \{k_1, k_2, \cdots k_M\}$. Then, f_i can be compactly written as

$$f_i(x) = \frac{1}{2}\|\tilde{y}_i - \tilde{A}_i x\|_{\tilde{\Lambda}_i}^2 + \frac{\beta}{N}x^T Bx. \quad (24)$$

where $\tilde{y}_i, \tilde{A}_i$ and $\tilde{\Lambda}_i$ are defined as

$$\tilde{y}_i = \begin{bmatrix} y_{k_1} \\ \vdots \\ y_{k_M} \end{bmatrix}, \tilde{A}_i = \begin{bmatrix} A_{k_1} \\ \vdots \\ A_{k_M} \end{bmatrix} \text{ and } \tilde{\Lambda}_i = \begin{bmatrix} \Lambda_{k_1} & & \\ & \ddots & \\ & & \Lambda_{k_M} \end{bmatrix}.$$

From (24), we can express F_i, the proximal map of f_i, as

$$F_i(v) = \text{argmin}_{z \in \mathbb{R}^n} \left\{ f_i(z) + \frac{\|z - v\|^2}{2\sigma^2} \right\}$$

$$= \text{argmin}_{z \in \mathbb{R}^n} \left\{ \frac{1}{2}z^t\left(H_i + \frac{I}{\sigma^2}\right)z - z^t\left(b_i + \frac{v}{\sigma^2}\right) \right\}, \quad (25)$$

where $H_i \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}^n$ are defined as

$$H_i = \tilde{A}_i^t \tilde{\Lambda}_i \tilde{A}_i + (\beta/N)B \text{ and } b_i = \tilde{A}_i^t \tilde{y}_i.$$

We can obtain $\tilde{F}_i(v; x)$, the partial-update for $F_i(v)$ defined in (25), by using the convergence analysis of [26], [33] for ICD optimization. This gives

$$\tilde{F}_i(v; x) = -(L_i + D_i + \sigma^{-2}I)^{-1}(L_i^t x - b_i - \sigma^{-2}v), \quad (26)$$

where matrices $L_i, D_i \in \mathbb{R}^{n \times n}$, are defined as

$L_i = $ Lower triangular sub-matrix of H_i (excluding diag.)

$D_i = $ Diagonal sub-matrix of H_i.

Further we define $\tilde{L}_i \in \mathbb{R}^{n \times n}$ by $\tilde{L}_i = L_i + D_i$. We can re-write equation (26) as

$$\tilde{F}_i(v; x) = -(\tilde{L}_i + \sigma^{-2}I)^{-1}(L_i^t x - b_i - \sigma^{-2}v)$$

$$= -\sigma^2(I + \sigma^2 \tilde{L}_i)^{-1}(L_i^t x - b_i - \sigma^{-2}v) \quad (27)$$

Let $\varepsilon = \sigma^2$. For sufficiently small ε^2, we can approximate $(I + \varepsilon\tilde{L}_i)^{-1}$ in (27) as

$$(I + \varepsilon\tilde{L}_i)^{-1} = I - \varepsilon\tilde{L}_i + \mathcal{O}(\varepsilon^2).$$

Plugging the above approximation into equation (27), simplifying, and dropping the $\mathcal{O}(\varepsilon^2)$ term, we get

$$\tilde{F}_i(v; x) = (I - \varepsilon\tilde{L}_i)v - \varepsilon L_i^t x + \varepsilon b_i \quad (28)$$

and hence

$$2\tilde{F}_i(v; x) - v = (I - 2\varepsilon\tilde{L}_i)v - 2\varepsilon L_i^t x + 2\varepsilon b_i. \quad (29)$$

Define block matrices $\mathbf{L} \in \mathbb{R}^{Nn \times Nn}$ and $\tilde{\mathbf{L}} \in \mathbb{R}^{Nn \times Nn}$ with L_i and $\tilde{L}_i$ along the diagonal, respectively.

Using (28), (29), $\mathbf{L}$, and $\tilde{\mathbf{L}}$ in (13), we can express the Partial-update MACE update up to terms involving $\mathcal{O}(\varepsilon^2)$ as

$$1\begin{bmatrix} \mathbf{w}^{(k+1)} \\ X^{(k+1)} \end{bmatrix} = \begin{bmatrix} \rho & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} - 2\varepsilon\tilde{\mathbf{L}} & -2\varepsilon\mathbf{L}^t \\ \mathbf{I} - \varepsilon\tilde{\mathbf{L}} & -\varepsilon\mathbf{L}^t \end{bmatrix} \begin{bmatrix} \mathbf{v}^{(k)} \\ X^{(k)} \end{bmatrix}$$

$$\times \begin{bmatrix} 1-\rho & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}^{(k)} \\ X^{(k)} \end{bmatrix} + \mathbf{c}$$

$$= \begin{bmatrix} \rho & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} - 2\varepsilon\tilde{\mathbf{L}} & -2\varepsilon\mathbf{L}^t \\ \mathbf{I} - \varepsilon\tilde{\mathbf{L}} & -\varepsilon\mathbf{L}^t \end{bmatrix} \begin{bmatrix} 2\mathbf{G} - \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

$$\times \begin{bmatrix} \mathbf{w}^{(k)} \\ X^{(k)} \end{bmatrix} + \begin{bmatrix} 1-\rho & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}^{(k)} \\ X^{(k)} \end{bmatrix} + \mathbf{c}$$

$$= \begin{bmatrix} \rho(\mathbf{I} - 2\varepsilon\tilde{\mathbf{L}})(2\mathbf{G} - \mathbf{I}) + (1-\rho)\mathbf{I} & -2\rho\varepsilon\mathbf{L}^t \\ (\mathbf{I} - \varepsilon\tilde{\mathbf{L}})(2\mathbf{G} - \mathbf{I}) & -\varepsilon\mathbf{L}^t \end{bmatrix}$$

$$\times \begin{bmatrix} \mathbf{w}^{(k)} \\ X^{(k)} \end{bmatrix} + \mathbf{c}, \quad (30)$$

where $\mathbf{c} \in \mathbb{R}^{2nN}$ is a constant term based on variables ρ and b_i. Define $\mathbf{M}_\varepsilon \in \mathbb{R}^{2nN \times 2nN}$ and $\mathbf{z} \in \mathbb{R}^{2nN}$ as follows

$$\mathbf{M}_\varepsilon = \begin{bmatrix} \rho(\mathbf{I} - 2\varepsilon\tilde{\mathbf{L}})(2\mathbf{G} - \mathbf{I}) + (1-\rho)\mathbf{I} & -2\rho\varepsilon\mathbf{L}^t \\ (\mathbf{I} - \varepsilon\tilde{\mathbf{L}})(2\mathbf{G} - \mathbf{I}) & -\varepsilon\mathbf{L}^t \end{bmatrix} \quad (31)$$

$$\mathbf{z} = \begin{bmatrix} \mathbf{w} \\ X \end{bmatrix} \quad (32)$$

Then we can re-write (30) as follows

$$\mathbf{z}^{(k+1)} = \mathbf{M}_\varepsilon \mathbf{z}^{(k)} + \mathbf{c} \quad (33)$$

For $\mathbf{z}^{(k)}$ to converge in the limit $k \to \infty$, the absolute value of any eigenvalue of $\mathbf{M}_\varepsilon$ must be in the range $(0,1)$. We first determine the eigenvalues of $\mathbf{M}_0$, where $\mathbf{M}_0 = \text{limit}_{\varepsilon \to 0}\mathbf{M}_\varepsilon$, and then apply a 1st order approximation in ε to obtain the eigenvalues of $\mathbf{M}_\varepsilon$. We can express $\mathbf{M}_0$ as

$$\mathbf{M}_0 = \begin{bmatrix} 2\rho\mathbf{G} + (1-2\rho)\mathbf{I} & 0 \\ 2\mathbf{G} - \mathbf{I} & 0 \end{bmatrix}$$

Let $\lambda_0 \in \mathbb{R}$ and $\mathbf{z}_0 = [\mathbf{w}_0^t X_0^t]^t \in \mathbb{R}^{2nN}$ represent eigenvalue and eigenvector of $\mathbf{M}_0$. Then $\mathbf{M}_0\mathbf{z}_0 = \lambda_0\mathbf{z}_0$, and so

$$\mathbf{G}\mathbf{w}_0 = \frac{1}{2\rho}(\lambda_0 + 2\rho - 1)\mathbf{w}_0 \quad (34)$$

$$(2\mathbf{G} - \mathbf{I})\mathbf{w}_0 = \lambda_0 X_0 \quad (35)$$

Since $\mathbf{G}$ is an orthogonal projection onto a subspace, all of its eigenvalues are 0 or 1. This with (34) implies that $\lambda_0 + 2\rho - 1$ is 0 or 2ρ. In the first case, $\lambda_0 = 1 - 2\rho$, which lies in the open interval $(-1, 1)$ for ρ in $(0,1)$. In the second case, $\lambda_0 = 1$. Applying this in (34) and (35), we get $X_0 = \mathbf{w}_0 = \mathbf{G}\mathbf{w}_0$, so that each eigenvector for $\lambda_0 = 1$ has $X_0 = \mathbf{w}_0$ with all subvectors identical.

Let $\lambda_\varepsilon \in \mathbb{R}$ and $\mathbf{z}_\varepsilon = [\mathbf{w}_\varepsilon^t X_\varepsilon^t]^t \in \mathbb{R}^{nN}$ represent eigenvalue and eigenvector of $\mathbf{M}_\varepsilon$ respectively. Let a be the derivative of λ_ε

with respect to ε, and let $\mathbf{u}_1 = \nabla_\varepsilon \mathbf{w}_\varepsilon$ and $\mathbf{u}_2 = \nabla_\varepsilon X_\varepsilon$. Applying a 1st order approximation in ε, λ_ε and $\mathbf{z}_\varepsilon$ are given by

$$\lambda_\varepsilon = \lambda_0 + a(\varepsilon - 0) = 1 + a\varepsilon$$

$$\mathbf{z}_\varepsilon = \begin{bmatrix} \mathbf{w}_0 + (\varepsilon - 0)\mathbf{u}_1 X_0 + (\varepsilon - 0)\mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{w}_0 + \varepsilon\mathbf{u}_1 \\ \mathbf{w}_0 + \varepsilon\mathbf{u}_2 \end{bmatrix}.$$

If we can prove that a is negative when ε is infinitely small positive, then consequently $|\lambda_\varepsilon| < 1$, and so, the system of equations specified by equation (30) converges (note that the case of $\lambda_0 = 1 - 2\rho$ gives $|\lambda_\varepsilon| < 1$ by continuity for small ε). Since $\mathbf{M}_\varepsilon \mathbf{z}_\varepsilon = \lambda_\varepsilon \mathbf{z}_\varepsilon$, the first component of equation (31) gives

$$\Big[\rho(\mathbf{I} - 2\varepsilon\tilde{\mathbf{L}})(2\mathbf{G} - \mathbf{I}) + (1 - \rho)\mathbf{I}\Big](\mathbf{w}_0 + \varepsilon\mathbf{u}_1)$$
$$- 2\rho\varepsilon\mathbf{L}^t(\mathbf{w}_0 + \varepsilon\mathbf{u}_2) = (1 + a\varepsilon)(\mathbf{w}_0 + \varepsilon\mathbf{u}_1).$$

Neglecting terms $\mathcal{O}(\varepsilon^2)$, expanding, and using $(2\mathbf{G} - \mathbf{I})\mathbf{w}_0 = \mathbf{w}_0$, the above simplifies for $\varepsilon > 0$ to

$$2\rho(\mathbf{G} - \mathbf{I})\mathbf{u}_1 = \Big[2\rho(\tilde{\mathbf{L}} + \mathbf{L}^t) + a\mathbf{I}\Big]\mathbf{w}_0.$$

Applying $\mathbf{G}$ to both sides, using $\mathbf{G}(\mathbf{G} - \mathbf{I}) = \mathbf{G}^2 - \mathbf{G} = 0$ and $\mathbf{G}\mathbf{w}_0 = \mathbf{w}_0$, we get

$$\mathbf{0} = 2\rho\mathbf{G}(\tilde{\mathbf{L}} + \mathbf{L}^t)\mathbf{w}_0 + a\mathbf{w}_0$$

and so,

$$\mathbf{G}(\tilde{\mathbf{L}} + \mathbf{L}^t)\mathbf{w}_0 = -\frac{a}{2\rho}\mathbf{w}_0. \tag{36}$$

Since $\tilde{L}_i + L_i^t, i = 1, \ldots, N$ is positive definite for each i, so is $\mathbf{H} = \tilde{\mathbf{L}} + \mathbf{L}^t$. Further, $\mathbf{G} \in \mathbb{R}^{nN \times nN}$ is an orthogonal projection matrix with n-dimensional range. Hence $\mathbf{G}$ can be expressed as $\mathbf{G} = PP^t$, where $P \in \mathbb{R}^{nN \times n}$ is orthogonal basis of the range of $\mathbf{G}$ (i.e $P^t P = I$). Since $\mathbf{w}_0 = \mathbf{G}\mathbf{w}_0$, equation (36) can be written as

$$PP^t\mathbf{H}PP^t\mathbf{w}_0 = -\frac{a}{2\rho}PP^t\mathbf{w}_0$$

Multiply both LHS and RHS by P^t, and define $\tilde{\mathbf{w}}_0 = P^t\mathbf{w}_0$. Since $P^t P = I$, we get

$$P^t\mathbf{H}P\tilde{\mathbf{w}}_0 = -\frac{a}{2\rho}\tilde{\mathbf{w}}_0$$

This implies that $-a/(2\rho)$ is an eigenvalue of $P^t\mathbf{H}P$. Since $\rho > 0$ and $P^t\mathbf{H}P$ is positive definite, we have $a < 0$, and consequently, $|\lambda_\varepsilon| < 1$.

Since all eigenvalues of $\mathbf{M}_\varepsilon$ have absolute value less than 1, the system of equations specified by (13) converges to a point $\mathbf{z}^* = (\mathbf{w}^*, \mathbf{X}^*)$ in the limit $k \to \infty$. From Theorem II.1, $\mathbf{w}^*$ is a solution to the exact MACE approach specified by (11). ∎

APPENDIX C

Theorem C.1: Let $F_i, i = 1, \ldots, N$, be the proximal map of a closed, convex, differentiable function, f_i, let $\sum_{i=1}^N f_i = f$, let F be the proximal map for f, and let H be any denoiser. Then, the MACE framework specified by equilibirum conditions (16)–(18) is exactly equivalent to the standard PnP framework specified by (14) and (15).

Proof: Assume (16)–(18) hold. Then, as per (16),

$$x^* = F_i(x^* + u_i^*; \sigma), i = 1, \ldots, N.$$

Since f_i is convex, differentiable, and F_i is defined as

$$F_i(x; \sigma) = \operatorname{argmin}_{v \in \mathbb{R}^n} \left\{ f_i(v) + \frac{\|v - x\|^2}{2\sigma^2} \right\},$$

it follows from the above stated equilibrium condition that

$$\nabla f_i(x^*) + \frac{x^* - (x^* + u_i^*)}{\sigma^2} = 0, \text{ or,}$$

$$\nabla f_i(x^*) - \frac{u_i^*}{\sigma^2} = 0. \tag{37}$$

Summing the above equation over $i = 1, \ldots, N$ we get

$$\sum_{i=1}^N \nabla f_i(x^*) - \frac{N\bar{u}^*}{\sigma^2} = 0,$$

where $\bar{\mathbf{u}} = \sum_{i=1}^N u_i/N$. Since $f = \sum_{i=1}^N f_i$, the above can be re-written as

$$\nabla f(x^*) + \frac{x^* - (x^* + N\bar{u}^*)}{\sigma^2} = 0.$$

Since f is convex, the above equation implies that

$$x^* = \operatorname{argmin}_{v \in \mathbb{R}^n} \left\{ f(v) + \frac{\|v - (x^* + N\bar{u}^*)\|^2}{2\sigma^2} \right\}, \text{ and so,}$$

$$x^* = F(x^* + N\bar{u}^*; \sigma).$$

From (17) and (18), we additionally get $x^* = H(x^* - N\bar{u}^*)$. Therefore, we get (14) and (15), where $\alpha^* = -N\bar{u}^*$.

For the converse, assume (14) and (15) hold. Then, as per (14), $F(x^* - \alpha^*; \sigma) = x^*$. So, we get

$$\nabla f(x^*) + \frac{x^* - (x^* - \alpha^*)}{\sigma^2} = 0.$$

Since $f = \sum_{i=1}^N f_i$, we can re-write the above as

$$\alpha^* = -\sum_{i=1}^N \sigma^2 \nabla f_i(x^*).$$

We define u_i^* as $u_i^* = \sigma^2 \nabla f_i(x^*)$. So, from the above equation, we get $\alpha^* + \sum_{i=1}^N u_i^* = 0$, which gives (18). Further from the defintion of u_i^* we get

$$\nabla f_i(x^*) + \frac{x^* - (x^* + u_i^*)}{\sigma^2} = 0, \text{ and so,}$$

$$F_i(x^* + u_i^*; \sigma) = x^*,$$

which gives (16). Also, as per (15), $H(x^* + \alpha^*) = x^*$, which gives (17). Therefore, we obtain (16)–(18), where $u_i^* = \sigma^2 \nabla f_i(x^*), i = 1, \ldots, N$. ∎

Remark: As in [12], the theorem statement and proof can be modified to allow for nondifferentiable, but still convex functions, f_i.

Proof: Proof of Theorem III.1

Assume (16)–(18) hold. We define $\mathbf{t}^*$ as $\mathbf{t}^* = N\mathbf{u}^*$. So, (18) gives $\alpha^* + (\sum_{i=1}^N t_i^*)/N = 0$, or, $\alpha^* = -\bar{\mathbf{t}}^*$. Consequently, we can express (17) as

$$H(x^* - \bar{\mathbf{t}}^*) = x^*. \tag{38}$$

Further, (16) specifies $F_i(x^* + u_i^*; \sigma) = x^*$. We showed earlier in (37) that this gives $\nabla f_i(x) - u_i/\sigma^2 = 0$. So, we get

$$\nabla f_i(x^*) - \frac{N u_i^*}{N \sigma^2} = 0, \quad \text{or,}$$

$$\nabla f_i(x^*) + \frac{x^* - (x^* + t_i^*)}{(\sqrt{N}\sigma)^2} = 0, \quad \text{or,}$$

$$F_i(x^* + t_i^*; \sqrt{N}\sigma) = x^*, \quad i = 1, \ldots, N. \tag{39}$$

Define $\mathbf{w}^*$ as $\mathbf{w}^* = \hat{x}^* - \mathbf{t}^*$, where $\hat{x}^*$ is N vertical copies of x^*. We write (38) as $\mathbf{G}_H \mathbf{w}^* = \hat{x}^*$. So, based on definition of $\mathbf{w}^*$, we have $\mathbf{t}^* = \hat{x}^* - \mathbf{w}^* = \mathbf{G}_H \mathbf{w}^* - \mathbf{w}^* = (\mathbf{G}_H - \mathbf{I})\mathbf{w}^*$. We can write (39) as $\mathbf{F}(\hat{x}^* + \mathbf{t}^*) = \hat{x}^*$ according to (19), and so, by plugging in $\hat{x}^*$ and $\mathbf{t}^*$ in terms of $\mathbf{w}^*$ we get

$$\mathbf{F}(\mathbf{G}_H \mathbf{w}^* + (\mathbf{G}_H - \mathbf{I})\mathbf{w}^*) = \mathbf{G}_H \mathbf{w}^*, \quad \text{or,}$$

$$\mathbf{F}(2\mathbf{G}_H - \mathbf{I})\mathbf{w}^* = \mathbf{G}_H \mathbf{w}^*.$$

Multiplying by 2 and adding $\mathbf{w}^*$ on both sides we get

$$(2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})\mathbf{w}^* = \mathbf{w}^*.$$

Therefore, $\mathbf{w}^*$ is a fixed-point of $\mathbf{T}_H = (2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})$, and, $\mathbf{G}_H(\mathbf{w}^*) = \hat{x}^*$, where $\mathbf{w}^*$ is given by $\mathbf{w}^* = \hat{x}^* - N\mathbf{u}^*$.

For the converse, assume $(2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})\mathbf{w}^* = \mathbf{w}^*$ and $\mathbf{G}_H \mathbf{w}^* = \hat{x}^*$ hold. The former gives $\mathbf{F}(2\mathbf{G}_H - \mathbf{I})\mathbf{w}^* = \mathbf{G}_H \mathbf{w}^*$. Applying the latter, we get $\mathbf{F}(2\hat{x}^* - \mathbf{w}^*) = \hat{x}^*$. Define $\mathbf{t}^*$ as $\mathbf{t}^* = \hat{x}^* - \mathbf{w}^*$. So, we have $\mathbf{F}(\hat{x}^* + \mathbf{t}^*) = \hat{x}^*$, or,

$$F_i(x^* + t_i^*; \sqrt{N}\sigma) = x^*, \quad i = 1, \ldots, N.$$

A calculation with the definition of F_i shows that the above gives

$$F_i(x^* + t_i^*/N; \sigma) = x^*, \quad i = 1, \ldots, N.$$

Define $\mathbf{u}^* = \mathbf{t}^*/N$. So, from the above, $F_i(x^* + u_i^*; \sigma) = x^*$, which gives (16). Since $\mathbf{G}_H \mathbf{w}^* = \hat{x}^*$, we have $x^* = H\bar{\mathbf{w}}^*$. Combining this with $\mathbf{w}^* = \hat{x}^* - \mathbf{t}^*$, we get $x^* = H(x^* - \bar{\mathbf{t}}^*)$. Define $\alpha^* = -\bar{\mathbf{t}}^*$. So we have, $x^* = H(x^* + \alpha^*)$, which gives (17). Also from definition of α^* and $\mathbf{u}^*$, we get $\alpha^* + \sum_{i=1}^{N} u_i^* = 0$, which gives (18). Therefore, we obtain (16) to (18), where $\mathbf{u}^*$ is given by $N\mathbf{u}^* = \hat{x}^* - \mathbf{w}^*$. ∎

Proof: Proof of Lemma III.2

First we show that $2\mathbf{G}_H - \mathbf{I}$ is non-expansive when H is firmly non-expansive. This proof also applies to the case where H is a proximal map, since proximal maps are firmly non-expansive [28]. Consider any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{nN}$. Then

$$\|(2\mathbf{G}_H - \mathbf{I})\mathbf{x} - (2\mathbf{G}_H - \mathbf{I})\mathbf{y}\|^2$$

$$= 4\|\mathbf{G}_H \mathbf{x} - \mathbf{G}_H \mathbf{y}\|^2 + \|\mathbf{x} - \mathbf{y}\|^2 - 4\langle \mathbf{G}_H \mathbf{x} - \mathbf{G}_H \mathbf{y}, \mathbf{x} - \mathbf{y}\rangle$$

By writing $\mathbf{G}_H$ in terms of H, we simplify the last term as

$$\langle \mathbf{G}_H \mathbf{x} - \mathbf{G}_H \mathbf{y}, \mathbf{x} - \mathbf{y}\rangle$$

$$= \sum_{i=1}^{N} \langle H\bar{\mathbf{x}} - H\bar{\mathbf{y}}, x_i - y_i\rangle$$

$$= \sum_{i=1}^{N} \langle H\bar{\mathbf{x}} - H\bar{\mathbf{y}}, \bar{\mathbf{x}} - \bar{\mathbf{y}} + (x_i - \bar{\mathbf{x}}) - (y_i - \bar{\mathbf{y}})\rangle$$

$$= N\langle H\bar{\mathbf{x}} - H\bar{\mathbf{y}}, \bar{\mathbf{x}} - \bar{\mathbf{y}}\rangle.$$

Since H is *firmly non-expansive*, [34, Prop. 4.2] implies that $\langle H\bar{\mathbf{x}} - H\bar{\mathbf{y}}, \bar{\mathbf{x}} - \bar{\mathbf{y}}\rangle \geq \|H\bar{\mathbf{x}} - H\bar{\mathbf{y}}\|^2$. This gives

$$\langle \mathbf{G}_H \mathbf{x} - \mathbf{G}_H \mathbf{y}, \mathbf{x} - \mathbf{y}\rangle \geq N\|H\bar{\mathbf{x}} - H\bar{\mathbf{y}}\|^2 = \|\mathbf{G}_H \mathbf{x} - \mathbf{G}_H \mathbf{y}\|^2.$$

Plugging the above into the first equation of this proof, we get

$$\|(2\mathbf{G}_H - \mathbf{I})\mathbf{x} - (2\mathbf{G}_H - \mathbf{I})\mathbf{y}\|^2 \leq \|\mathbf{x} - \mathbf{y}\|^2.$$

Therefore, $(2\mathbf{G}_H - \mathbf{I})$ is a non-expansive map. Also, since $F_i, i = 1, \ldots, N$ is the proximal map of a convex function, F_i is a resolvent operator, so $2F_i - I$ is a reflected resolvent operator, hence non-expansive. This means $2\mathbf{F} - \mathbf{I}$ is non-expansive, so $(2\mathbf{F} - \mathbf{I})(2\mathbf{G}_H - \mathbf{I})$ is non-expansive, since it is the composition of two non-expansive maps. ∎

REFERENCES

[1] J. Fessler, "Fundamentals of ct reconstruction in 2D and 3D," *Comprehensive Biomed. Phys.*, vol. 2, pp. 263–295, 2014.

[2] H. Turbell, "Cone-beam reconstruction using filtered backprojection," Ph.D. dissertation, Dept. Elect. Eng., Linkopins Univ., Linköping, Sweden 2001.

[3] M. Beister and W. Kolditz, and D. Kalender, "Iterative reconstruction methods in X-ray CT," *Physica Medica*, vol. 28, pp. 94–108, 2012.

[4] S. J. Kisner, E. Haneda, C. A. Bouman, S. Skatter, M. Kourinny, and S. Bedford, "Model-based CT reconstruction from sparse views," in *Proc. Int. Conf. Image Formation X-Ray Comput. Tomography*, 2012, pp. 444–447.

[5] B. De Mann *et al.*, "A study of four minimization approaches for iterative reconstruction in X-ray CT," in *Proc. IEEE Nucl. Sci. Symp. Conf. Rec.*, 2005, pp. 2708–2710.

[6] C. A. Bouman and K. D. Sauer, "A unified approach to statistical tomography using coordinate descent optimization," *IEEE Trans. Image Process.*, vol. 5, no. 3, pp. 480–492, Mar. 1996.

[7] S. Sreehari *et al.*, "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.

[8] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2013, pp. 945–948.

[9] X. Wang, A. Sabne, S. J. Kisner, A. Raghunathan, S. Midkiff, and C. A. Bouman, "High performance model based image reconstruction," in *Proc. ACM PPoPP Conf.*, 2016, pp. 1–12.

[10] A. Sabne, X. Wang, S. J. Kisner, A. Raghunathan, S. Midkiff, and C. A. Bouman, "Model-based iterative CT image reconstruction on GPUs," in *Proc. ACM PPoPP Conf.*, 2016, pp. 207–220.

[11] D. Matenine, G. Cote, J. Mascolo-Fortin, Y. Goussard, and P. Despres, "System matrix computation vs storage on GPU: A comparative study in cone beam CT," *Med. Phys.*, vol. 45, no. 2, pp. 579–588, 2018.

[12] G. T. Buzzard, S. 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. 1–20, 2018.

[13] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, 2017.

[14] J. Zheng, S. Saquib, K. Sauer, and C. A. Bouman, "Parallelizable Bayesian tomography algorithm with rapid, guarenteed convergence," *IEEE Trans. Med. Imag.*, vol. 9, no. 10, pp. 1745–1759, Oct. 2000.

[15] J. Fessler and D. Kim, "Axial block coordinate descent (abcd) algorithm for X-ray CT image reconstruction," in *Proc. 11th Int. Meeting Fully Three-Dimensional Image Reconstruction Radiol. Nucl. Medicine*, 2011, pp. 262–265.

[16] X. Wang, A. Sabne, P. Sakdhnagool, S. J. Kisner, C. A. Bouman, and S. P. Midkiff, "Massively parallel 3D image reconstruction," in *Proc. ACM SC Conf.*, 2017, pp. 1–12.

[17] W. Linyuan *et al.*, "Distributed CT image reconstruction algorithm based on the alternating direction method," *J. X-Ray Sci. Technol.*, vol. 23, pp. 83–98, 2014.

[18] J. Cui, G. Pratx, B. Meng, and C. S. Levin, "Distributed MLEM: An iterative tomographic image reconstruction algorithm for distributed memory architectures," *IEEE Trans. Med. Imag.*, vol. 32, no. 5, pp. 957–967, May 2013.

[19] K. Sauer and C. Bouman, "Bayesian estimation of transmission tomograms using segmentation-based optimization," *IEEE Trans. Nucl. Sci.*, vol. 39, no. 4, pp. 1144–1152, Aug. 1992.

[20] A. Chambolle, V. Caselles, M. Novaga, D. Cremers, and T. Pock, "An introduction to total variation for image analysis," *HAL Archives*, pp. 1–49 2009.

[21] V. Sridhar, G. T. Buzzard, and C. A. Bouman, "Distributed framework for fast iterative CT reconstruction from View-subsets," in *Proc. Conf. Comput. Imag., IS&T Electron. Imag.*, 2018, pp. 102-1–102-7.

[22] X. Wang *et al.*, "Consensus equilibrium framework for super-resolution and extreme-scale CT reconstruction," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage, Anal.*, 2019, pp. 1–13.

[23] J. B. Thibault, K. D. Sauer, C. A. Bouman, and J. Hsieh, "A three-dimensional statistical approach to improved image quality for multislice helical CT," *Med. Phys.*, vol. 34, no. 11, pp. 4526–44, 2007.

[24] 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.

[25] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[26] K. Sauer and C. A. Bouman, "A local update strategy for iterative reconstruction from projections," *IEEE Trans. Signal Process.*, vol. 41, no. 2, pp. 534–548, Feb. 1993.

[27] C. A. Bouman and K. Sauer, "A generalized Gaussian image model for edge-preserving map estimation," *IEEE Trans. Image Process.*, vol. 2, no. 3, pp. 296–310, Jul. 1993.

[28] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations Trends Optim.*, vol. 1, no. 3, pp. 123–231, 2013.

[29] S. Majee, T. Balke, C. A. Kemp, G. T. Buzzard, and C. A. Bouman, "4D X-ray CT reconstruction using multi-slice fusion," in *Proc. Int. Conf. Comput. Photography*, 2019, pp. 1–8.

[30] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, 2005, pp. 60–65.

[31] V. Sridhar, G. T. Buzzard, and C. A. Bouman, "MPI implementation of distributed memory approach to CT reconstruction," 2018. [Online]. Available: https://github.rcac.purdue.edu/vsridha?tab=repositories

[32] Z. Yu, J. B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, "'fast model-based X-ray CT reconstruction using spatially non-homogeneous icd optimization," *IEEE Trans. Image Process.*, vol. 20, no. 1, pp. 161–175, Jan. 2011.

[33] C. A. Bouman, "Model based image processing," 2013. [Online]. Available: https://engineering.purdue.edu/ bouman/publications/pdf/MBIP-book.pdf

[34] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. : Berlin, Germany: Springer, 2011.

Xiao Wang (Member, IEEE) received the B.A. degrees in mathematics and computer science with honor from Saint John's University, MN, in 2012, and M.S.E.E. and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, in 2016 and 2017, respectively. Currently, he works as a Postdoctoral Research Fellow at Harvard Medical School and Boston Children's Hospital under the supervision of Professor Simon Warfield. Xiao Wang's research interest focuses on applying medical physics, image processing and high performance computing to computed tomographic (CT) and positron emission tomography (PET) reconstruction, so that the reconstruction algorithm tightly integrates with the scanner detector sensor design and data acquisition physics, making it possible to observe phenomena at a super-resolution previously difficult or impossible to measure with traditional approaches.

Gregery T. Buzzard (Member, IEEE) received degrees in violin performance, computer science, and mathematics from Michigan State University and received the Ph.D. degree in mathematics from the University of Michigan in 1995. He held Postdoctoral positions at Indiana University, Bloomington, and Cornell University before joining the mathematics faculty at Purdue University, West Lafayette, in 2002, where he served as head of the Department of Mathematics from 2013-2020. His research interests include dynamical systems, experiment design, uncertainty quantification, and inverse problems. He is a Member of the IEEE.

Charles A. Bouman (Fellow, IEEE) received the B.S.E.E. degree from the University of Pennsylvania in 1981, and the M.S. degree from the University of California at Berkeley in 1982, and the Ph.D. degree in electrical engineering from Princeton University in 1989. He joined the faculty of Purdue University in 1989 where he is currently the Showalter Professor of Electrical and Computer Engineering and Biomedical Engineering. Prof. Bouman's research is in the area of computational imaging and sensing where he is focused on the integration of signal processing, statistical modeling, physics, and computation to solve difficult sensing problems with applications in healthcare, material science, physics, chemistry, and commercial imaging. His research resulted in the first commercial model-based iterative reconstruction (MBIR) system for medical X-ray computed tomography (CT), and he is co-inventor on over 50 issued patents that have been licensed and used in millions of consumer imaging products. Professor Bouman is a member of the National Academy of Inventors, a Fellow of the IEEE, a Fellow of the American Institute for Medical and Biological Engineering (AIMBE), a Fellow of the society for Imaging Science and Technology (IS&T), a Fellow of the SPIE professional society. He is the recipient of the 2014 Electronic Imaging Scientist of the Year award, and the IS&T's Raymond C. Bowman Award. He has been a Purdue University Faculty Scholar and received the College of Engineering Engagement/Service Award, and Team Award. He was also founding co-director of Purdue's Magnetic Resonance Imaging Facility from 2007-2016, and led of Purdue's Integrated Imaging Cluster from 2012-2016. He was previously the Editor-in-Chief for the IEEE TRANSACTIONS ON IMAGE PROCESSING; a Distinguished Lecturer for the IEEE Signal Processing Society; and a Vice President of Technical Activities for the IEEE Signal Processing Society, during which time he led the creation of the IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING. He has been an associate editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, and the SIAM Journal on Mathematical Imaging. He has also been a Vice President of Publications and a member of the Board of Directors for the IS&T Society, and he is the founder and Co-Chair of the SPIE/IS&T conference on Computational Imaging.

Venkatesh Sridhar (Student Member, IEEE) received the bachelor's degree in electronics & instrumentation engineering from the Birla Institute of Technology and Science, Pilani, India, in 2012, and the master's and Ph.D. degrees in electrical & computer engineering from Purdue University, West Lafayette, IN, USA, in 2014 and 2020, respectively. He worked at Intel Corporation, Oregon, USA as an Analog Engineer from 2014-2015. His research interests include computational imaging, tomography, radar, inverse problems, machine learning and parallel computing.