Compressive Imaging Using Approximate Message Passing and a Markov-Tree Prior

Subhojit Som and Philip Schniter

***Abstract*—We propose a novel algorithm for compressive imaging that exploits both the sparsity and persistence across scales found in the 2D wavelet transform coefficients of natural images. Like other recent works, we model wavelet structure using a hidden Markov tree (HMT) but, unlike other works, ours is based on loopy belief propagation (LBP). For LBP, we adopt a recently proposed "turbo" message passing schedule that alternates between exploitation of HMT structure and exploitation of compressive-measurement structure. For the latter, we leverage Donoho, Maleki, and Montanari's recently proposed approximate message passing (AMP) algorithm. Experiments with a large image database suggest that, relative to existing schemes, our turbo LBP approach yields state-of-the-art reconstruction performance with substantial reduction in complexity.**

***Index Terms*—Belief propagation, compressed sensing, hidden Markov tree, image reconstruction, structured sparsity.**

I. Introduction

In compressive imaging [1], we aim to estimate an image $\boldsymbol{x} \in \mathbb{R}^N$ from $M \leq N$ noisy linear observations $\boldsymbol{y} \in \mathbb{R}^M$,

$$\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{w} = \boldsymbol{\Phi \Psi \theta} + \boldsymbol{w}, \tag{1}$$

assuming that the image has a representation $\boldsymbol{\theta} \in \mathbb{R}^N$ in some wavelet basis $\boldsymbol{\Psi}$ (i.e., $\boldsymbol{x} = \boldsymbol{\Psi \theta}$) containing only a few (K) large coefficients (i.e., $K \ll N$). In (1), $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ is a known measurement matrix and $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ is additive white Gaussian noise. Though $M < N$ makes the problem ill-posed, it has been shown that $\boldsymbol{x}$ can be recovered from $\boldsymbol{y}$ when K is adequately small and $\boldsymbol{\Phi}$ is incoherent with $\boldsymbol{\Psi}$ [1]. The wavelet coefficients of natural images are known to have an additional structure known as persistence across scales (PAS) [2], which we now describe. For 2D images, the wavelet coefficients are naturally organized into quad-trees, where each coefficient at level j acts as a parent for four child coefficients at level $j+1$. The PAS property says that, if a parent is very small, then all of its children are likely to be very small; similarly, if a parent is large, then it is likely that some (but not necessarily all) of its children will also be large.

Several authors have exploited the PAS property for compressive imaging [3]–[6]. The so-called "model-based" approach [3] is a deterministic incarnation of PAS that leverages a restricted union-of-subspaces and manifests as a modified CoSaMP[1] [8] algorithm. Most approaches are Bayesian in nature, exploiting the fact that PAS is readily modeled by a hidden Markov tree (HMT) [9]. The first work in this direction appears to be [4], where an iteratively re-weighted ℓ_1 algorithm, generating an estimate of $\boldsymbol{x}$, was alternated with a Viterbi algorithm, generating an estimate of the HMT states. More recently, HMT-based compressive imaging has been attacked using modern Bayesian tools [10]. For example, [5] used Markov-chain Monte Carlo (MCMC), which is known to yield correct posteriors after convergence. For practical image sizes, however, convergence takes an impractically long time, and so MCMC must be terminated early, at which point its performance may suffer. Variational Bayes (VB) can sometimes offer a better performance/complexity tradeoff, motivating the approach in [6]. Our experiments indicate that, while [6] indeed offers a good performance/complexity tradeoff, it is possible to do significantly better.

In this paper, we propose a novel approach to HMT-based compressive imaging based on loopy belief propagation [11]. For this, we model the coefficients in $\boldsymbol{\theta}$ as conditionally Gaussian with variances that depend on the values of HMT states, and we propagate beliefs (about both coefficients and states) on the corresponding factor graph. A recently proposed "turbo" messaging schedule [12] suggests to iterate between exploitation of HMT structure and exploitation of observation structure from (1). For the former, we use the standard sum-product algorithm [13], [14], and for the latter, we use the recently proposed approximate message passing (AMP) approach [15]. The remarkable properties of AMP are 1) a rigorous analysis (as $M, N \to \infty$ with M/N fixed, under i.i.d. Gaussian $\boldsymbol{\Phi}$) [16] establishing that its solutions are governed by a state-evolution whose fixed points—when unique—yield the true posterior means, and 2) very low implementational complexity (e.g., AMP requires one forward and one inverse fast-wavelet-transform per iteration, and very few iterations).

We consider two types of conditional-Gaussian coefficient models: a Bernoulli–Gaussian (BG) model and a two-state Gaussian-mixture (GM) model. The BG model assumes that

Manuscript received August 12, 2011; revised January 02, 2012; accepted February 29, 2012. Date of publication March 23, 2012; date of current version June 12, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Olgica Milenkovic. This work was supported in part by NSF Grant CCF-1018368, AFOSR Grant FA9550-06-1-0324, DARPA/ONR Grant N66001-10-1-4090, and an allocation of computing time from the Ohio Supercomputer Center. This work was presented in part at the Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, November 2010.

S. Som is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30308 USA (e-mail: subhojit@gatech.edu).

P. Schniter is with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43202 USA (e-mail: schniter@ece.osu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2012.2191780

[1]We note that CoSaMP is very closely related to the subspace pursuit algorithm previously proposed by Dai and Milenkovic [7]. However, in order to compare directly with the model-based CoSaMP extensions from [3], we focus on CoSaMP rather than Subspace Pursuit in this work.

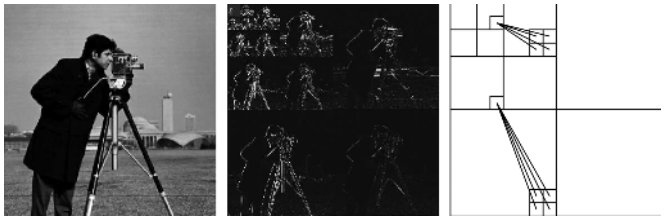

Fig. 1. Left: The cameraman image. Center: The corresponding transform coefficients, demonstrating PAS. Right: An illustration of quad-tree structure.

the coefficients are either generated from a large-variance Gaussian distribution or are exactly zero (i.e., the coefficients are exactly sparse), whereas the GM model assumes that the coefficients are generated from either a large-variance or a small-variance Gaussian distribution. Both models have been previously applied for imaging, e.g., the BG model was used in [5] and [6], whereas the GM model was used in [4] and [9].

Although our models for the coefficients $\boldsymbol{\theta}$ and the corresponding HMT states involve statistical parameters like variance and transition probability, we learn those parameters directly from the data. To do so, we take a hierarchical Bayesian approach—similar to [5] and [6]—where these statistical parameters are treated as random variables with suitable hyperpriors. Experiments on a large image database show that our turbo-AMP approach yields state-of-the-art reconstruction performance with substantial reduction in complexity.

The remainder of the paper is organized as follows. Section II describes the signal model, Section III describes the proposed algorithm, Section IV gives numerical results and comparisons with other algorithms, and Section V concludes.

Notation: Above and in the sequel, we use lowercase boldface quantities to denote vectors, uppercase boldface quantities to denote matrices, $\boldsymbol{I}$ to denote the identity matrix, $(\cdot)^T$ to denote transpose, and $\|\boldsymbol{x}\|_2 \triangleq \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$. We use $p_{\Theta\,|\,S}(\theta\,|\,s)$ to denote the probability density[2] function (pdf) of random variable Θ given the event $S = s$, where often the subscript "$_{\Theta\,|\,S}$" is omitted when there is no danger of confusion. We use $\mathcal{N}(\boldsymbol{x};\boldsymbol{m},\boldsymbol{\Sigma})$ to denote the N-dimensional Gaussian pdf with argument $\boldsymbol{x}$, mean $\boldsymbol{m}$, and covariance matrix $\boldsymbol{\Sigma}$, and we write $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{m},\boldsymbol{\Sigma})$ to indicate that random vector $\boldsymbol{x}$ has this pdf. We use $\mathrm{E}\{\cdot\}$ to denote expectation, $\Pr\{\mathcal{E}\}$ to denote the probability of event $\mathcal{E}$, and $\delta(\cdot)$ to denote the Dirac delta. Finally, we use $\propto$ to denote equality up to a multiplicative constant.

II. Signal Model

Throughout, we assume that $\boldsymbol{\Psi}$ represents a 2D wavelet transform [2], so that the transform coefficients $\boldsymbol{\theta} = [\theta_1,\ldots,\theta_N]^T$ can be partitioned into so-called "wavelet" coefficients (at indices $n \in \mathcal{W}$) and "approximation" coefficients (at indices $n \in \mathcal{A}$). The wavelet coefficients can be further partitioned into several quad-trees, each with $J \geq 1$ levels (see Fig. 1). We denote the indices of all coefficients at level $j \in \{0,\ldots,J-1\}$ of these wavelet trees by $\mathcal{W}_j$, where $j = 0$ refers to the root. In the interest of brevity, and with a slight abuse of notation, we refer to the approximation coefficients as level "-1" of the wavelet tree (i.e., $\mathcal{A} = \mathcal{W}_{-1}$).

[2]or the probability mass function (pmf), as will be clear from the context.

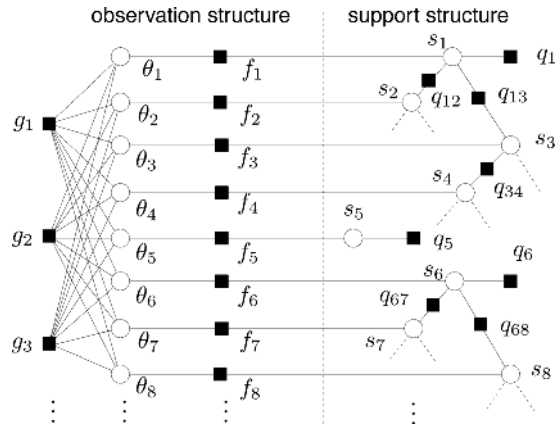

Fig. 2. Factor graph representation of the signal model. The variables s_1 and s_6 are wavelet states at the roots of two different Markov trees. The variable s_5 is an approximation state and hence is not part of any Markov tree. The remaining s_n are wavelet states at levels $j > 0$. For visual simplicity, a binary-tree is shown instead of a quad-tree, and the nodes representing the statistical parameters ρ, π^1_{-1}, π^1_0, $\{\rho_j, \pi^{11}_j, \pi^{00}_j\}$, as well as those representing their hyperpriors, are not shown. The f_n nodes represent the conditional pdfs $p(\theta_n\,|\,s_n)$, the q_{ij} nodes represent the conditional pmfs $p(s_j\,|\,s_i)$, and the q_i nodes represent the prior pmfs $p(s_i)$.

As discussed earlier, two coefficient models are considered in this paper: BG and two-state GM. For ease of exposition, we focus on the BG model until Section III-E, at which point the GM case is detailed. In the BG model, each transform coefficient θ_n is modeled using the (conditionally independent) prior pdf

$$p(\theta_n\,|\,s_n) = s_n\mathcal{N}\left(\theta_n;0,\sigma_n^2\right) + (1-s_n)\delta(\theta_n) \qquad (2)$$

where $s_n \in \{0,1\}$ is a hidden binary state. The approximation states $\{s_n\}_{n\in\mathcal{W}_{-1}}$ are assigned the *a priori* activity rate $\pi^1_{-1} \triangleq \Pr\{s_n = 1\,|\,n \in \mathcal{W}_{-1}\}$, which is discussed further below. Meanwhile, the root wavelet states $\{s_n\}_{n\in\mathcal{W}_0}$ are assigned $\pi^1_0 \triangleq \Pr\{s_n = 1\,|\,n \in \mathcal{W}_0\}$. Within each quad-tree, the states have a Markov structure. In particular, the activity of a state at level $j+1$ is determined by its parent's activity (at level j) and the transition probabilities $\{\pi^{00}_j, \pi^{11}_j\}$, where π^{00}_j denotes the probability that the child's state equals 0, given that his parent's state also equals 0, and π^{11}_j denotes the probability that the child's state equals 1, given that his parent's state also equals 1. The corresponding factor graph is shown in Fig. 2.

We take a hierarchical Bayesian approach, modeling the statistical parameters $\sigma^2, \{\sigma^2_n\}_{n=1}^N, \pi^1_{-1}, \pi^1_0, \{\pi^{11}_j, \pi^{00}_j\}_{j=0}^{J-2}$ as random variables and assigning them appropriate hyperpriors. Rather than working directly with variances, we find it more convenient to work with precisions (i.e., inverse-variances), such as $\rho \triangleq \sigma^{-2}$. We then assume that all coefficients at the same level have the same precision, so that $\rho_j = \sigma_n^{-2}$ for all $n \in \mathcal{W}_j$. To these precisions, we assign conjugate priors [17], which in this case take the form

$$\rho \sim \mathrm{Gamma}(a,b) \qquad (3)$$

$$\rho_j \sim \mathrm{Gamma}(a_j,b_j) \qquad (4)$$

where $\mathrm{Gamma}(\rho;a,b) \triangleq \frac{1}{\Gamma(a)}b^a\rho^{a-1}\exp(-b\rho)$ for $\rho \geq 0$, and where $a, b, \{a_j,b_j\}_{j=-1}^{J-1}$ are hyperparameters. (Recall that the mean and variance of $\mathrm{Gamma}(a,b)$ are given by a/b and

a/b^2, respectively [17].) For the activity rates and transition parameters, we assume

$$\pi_0^1 \sim \text{Beta}(c,d) \tag{5}$$
$$\pi_{-1}^1 \sim \text{Beta}(\underline{c},\underline{d}) \tag{6}$$
$$\pi_j^{11} \sim \text{Beta}(c_j,d_j) \tag{7}$$
$$\pi_j^{00} \sim \text{Beta}(\underline{c}_j,\underline{d}_j) \tag{8}$$

where $\text{Beta}(p;c,d) \triangleq \frac{\Gamma(c+d)}{\Gamma(c)\Gamma(d)}p^{c-1}(1-p)^{d-1}$, and where $c,d,\underline{c},\underline{d},\{c_j,d_j,\underline{c}_j,\underline{d}_j\}_{j=1}^{J-1}$ are hyperparameters. (Recall that the mean and variance of $\text{Beta}(c,d)$ are given by $\frac{c}{c+d}$ and $\frac{cd}{(c+d)^2(c+d+1)}$, respectively [17].) Our hyperparameter choices are detailed in Section IV.

III. Image Reconstruction

To infer the wavelet coefficients $\boldsymbol{\theta}$, we would ideally like to compute the posterior pdf

$$p(\boldsymbol{\theta}\,|\,\boldsymbol{y}) \propto \sum_{\boldsymbol{s}} p(\boldsymbol{y}\,|\,\boldsymbol{\theta},\boldsymbol{s})p(\boldsymbol{\theta},\boldsymbol{s}) \tag{9}$$

$$= \sum_{\boldsymbol{s}} \underbrace{p(\boldsymbol{s})}_{\triangleq\, h(\boldsymbol{s})} \prod_{n=1}^{N} \underbrace{p(\theta_n\,|\,s_n)}_{\triangleq\, f_n(\theta_n,s_n)} \prod_{m=1}^{M} \underbrace{p(y_m\,|\,\boldsymbol{\theta})}_{\triangleq\, g_m(\boldsymbol{\theta})} \tag{10}$$

where $\propto$ denotes equality up to a multiplicative constant. For the BG coefficient model, $f_n(\theta_n,s_n)$ is specified by (2). Due to the white Gaussian noise model (1), we have $g_m(\boldsymbol{\theta}) = \mathcal{N}(y_m;\boldsymbol{a}_m^T\boldsymbol{\theta},\sigma^2)$, where $\boldsymbol{a}_m^T$ denotes the mth row of the matrix $\boldsymbol{A} \triangleq \boldsymbol{\Phi}\boldsymbol{\Psi}$.

A. Loopy Belief Propagation

While exact computation of $p(\boldsymbol{\theta}\,|\,\boldsymbol{y})$ is computationally prohibitive, the marginal posteriors $\{p(\theta_n\,|\,\boldsymbol{y})\}$ can be efficiently approximated using loopy belief propagation (LBP) [11] on the factor graph of Fig. 2, which uses round nodes to denote variables and square nodes to denote the factors in (10). In doing so, we also obtain the marginal posteriors $\{p(s_n\,|\,\boldsymbol{y})\}$. For now, we treat statistical parameters $\rho,\pi_{-1}^1,\pi_0^1,\{\rho_j,\pi_j^{11},\pi_j^{00}\}$, as if they were fixed and known, and we detail the procedure by which they are learned in Section III-D.

In LBP, messages are exchanged between the nodes of the factor graph until they converge. Messages take the form of pdfs (or pmfs), and the message flowing to/from a variable node can be interpreted as a local belief about that variable. According to the *sum-product algorithm* [13], [14] the message emitted by a variable node along a given edge is (an appropriate scaling of) the product of the incoming messages on all other edges. Meanwhile, the message emitted by a function node along a given edge is (an appropriate scaling of) the integral (or sum) of the product of the node's constraint function and the incoming messages on all other edges, where the integration (or summation) is performed over all variables other than the one directly connected to the edge along which the message travels. When the factor graph has no loops, exact marginal posteriors result from two (i.e., forward and backward) passes of the sum-product algorithm [13], [14]. When the factor graph has loops, however, exact inference is known to be NP hard [18], and so LBP is not guaranteed to produce correct posteriors. Still, LBP has shown state-of-the-art performance in many applications, such as inference on Markov random fields [19], turbo decoding [20], LDPC decoding [21], multiuser detection [22], and compressive sensing [15], [16], [23], [24].

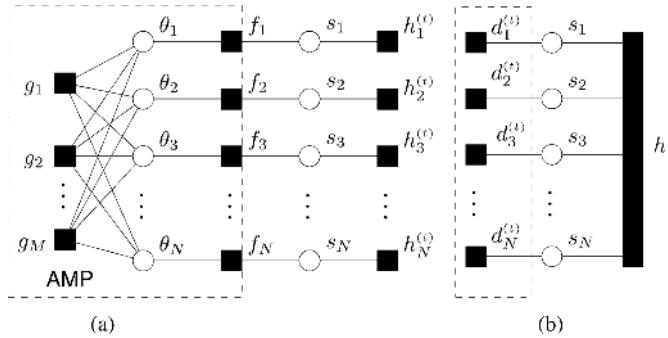

Fig. 3. The turbo approach yields a decoupled factor graph. (a) The factor graph of SSR. Node $h_n^{(t)}$ represents the prior $\Pr\{s_n = 1\}$ used by SSR during the tth turbo iteration. (b) The factor graph of SSD. Node $d_n^{(t)}$ represents the prior $\Pr\{s_n = 1\}$ used by SSD during the tth turbo iteration.

B. Message Scheduling: The Turbo Approach

With loopy belief propagation, there exists some freedom in how messages are scheduled. In this work, we adopt the "turbo" approach recently proposed in [12]. For this, we split the factor graph in Fig. 2 along the dashed line and obtain the two decoupled subgraphs in Fig. 3. We then alternate between belief propagation on each of these two subgraphs, treating the likelihoods on $\{s_n\}$ generated from belief propagation on one subgraph as priors for subsequent belief propagation on the other subgraph. We now give a more precise description of this turbo scheme, referring to one full round of alternation as a "turbo iteration." In the sequel, we use $\nu_{A\to B}^{(t)}(\cdot)$ to denote the message passed from node A to node B during the tth turbo iteration.

The procedure starts at $t = 1$ by setting the "prior" pmfs $\{h_n^{(1)}(\cdot)\}$ in accordance with the apriori activity rates $\Pr\{s_n = 1\}$ described in Section II. LBP is then iterated (to convergence) on the left subgraph in Fig. 3, finally yielding the messages $\{\nu_{f_n\to s_n}^{(1)}(\cdot)\}$. We note that the message $\nu_{f_n\to s_n}^{(1)}(s_n)$ can be interpreted as the current estimate of the likelihood[3] on s_n, i.e., $p(\boldsymbol{y}\,|\,s_n)$ as a function of s_n. These likelihoods are then treated as priors for belief propagation on the right subgraph, as facilitated by the assignment $d_n^{(1)}(\cdot) = \nu_{f_n\to s_n}^{(1)}(\cdot)$ for each n. Due to the tree structure of HMT, there are no loops in right subgraph (i.e., inside the "h" super-node in Fig. 3), and thus it suffices to perform only one forward–backward pass of the sum-product algorithm [13], [14]. The resulting leftward messages $\nu_{h\to s_n}^{(1)}(\cdot)$ are subsequently treated as priors for belief propagation on the left subgraph at the next turbo iteration, as facilitated by the assignment $h_n^{(2)}(\cdot) = \nu_{h\to s_n}^{(1)}(\cdot)$. The process then continues for turbo iterations $t = 2,3,4,\ldots$, until the likelihoods converge or

[3]In turbo decoding parlance, the likelihood $\nu_{f_n\to s_n}^{(t)}(s_n)$ would be referred to as the "extrinsic" information about s_n produced by the left "decoder", since it does not directly involve the corresponding prior $h_n^{(t)}(s_n)$. Similarly, the message $\nu_{h\to s_n}^{(t)}(s_n)$ would be referred to as the extrinsic information about s_n produced by the right decoder.

a maximum number of turbo iterations has elapsed. Formally, the turbo schedule is summarized by

$$d_n^{(t)}(s_n) = \nu_{f_n \to s_n}^{(t)}(s_n) \tag{11}$$

$$h_n^{(t+1)}(s_n) = \nu_{h \to s_n}^{(t)}(s_n). \tag{12}$$

In the sequel, we refer to inference of $\{s_n\}$ using compressive-measurement structure (i.e., inference on the left subgraph of Fig. 3) as soft support-recovery (SSR) and inference of $\{s_n\}$ using HMT structure (i.e., inference on the right subgraph of Fig. 3) as soft support-decoding (SSD). SSR details are described in the next subsection.

C. Soft Support-Recovery via AMP

We now discuss our implementation of SSR during a single turbo iteration t. Because the operations are invariant to t, we suppress the t-notation. As described above, SSR performs several iterations of loopy belief propagation per turbo iteration using the fixed priors $\lambda_n \triangleq h_n(s_n = 1)$. This implies that, over SSR's LBP iterations, the message $\nu_{f_n \to \theta_n}(\cdot)$ is fixed at

$$\nu_{f_n \to \theta_n}(\theta_n) = \lambda_n \mathcal{N}\left(\theta_n; 0, \sigma_n^2\right) + (1 - \lambda_n)\delta(\theta_n). \tag{13}$$

The dashed box in Fig. 3 shows the region of the factor graph on which messages are updated during SSR's LBP iterations. This subgraph can be recognized as the one that Donoho, Maleki, and Montanari used to derive their so-called approximate message passing (AMP) algorithm [15]. While [15] assumed an i.i.d. Laplacian prior for $\boldsymbol{\theta}$, the approach for generic i.i.d. priors was outlined in [24]. Below, we extend the approach of [24] to independent *non*-identical priors (as analyzed in [25]), and we detail the Bernoulli–Gaussian case. In the sequel, we use a superscript-i to index SSR's LBP iterations.

According to the sum-product algorithm, the fact that $\nu_{f_n \to \theta_n}(\cdot)$ is non-Gaussian implies that $\nu_{\theta_n \to g_m}^i(\cdot)$ is also non-Gaussian, which complicates the exact calculation of the subsequent messages $\nu_{g_m \to \theta_n}^i(\cdot)$ as defined by the sum-product algorithm. However, for large N, the combined effect of $\{\nu_{\theta_n \to g_m}^i(\cdot)\}_{n=1}^N$ at the g_m nodes can be approximated as Gaussian using central-limit theorem (CLT) arguments, after which it becomes sufficient to parameterize each message $\nu_{\theta_n \to g_m}^i(\cdot)$ by only its mean and variance:

$$\mu_{mn}^i \triangleq \int_{\theta_n} \theta_n \, \nu_{\theta_n \to g_m}^i(\theta_n) \tag{14}$$

$$v_{mn}^i \triangleq \int_{\theta_n} (\theta_n - \mu_{mn}^i)^2 \, \nu_{\theta_n \to g_m}^i(\theta_n). \tag{15}$$

Combining

$$\prod_q \mathcal{N}(\theta; \mu_q, v_q) \propto \mathcal{N}\left(\theta; \frac{\sum_q \mu_q / v_q}{\sum_q 1/v_q}, \frac{1}{\sum_q 1/v_q}\right) \tag{16}$$

with $g_m(\boldsymbol{\theta}) = \mathcal{N}(y_m; \boldsymbol{a}_m^T \boldsymbol{\theta}, \sigma^2)$, the CLT then implies that

$$\nu_{g_m \to \theta_n}^i(\theta_n) \approx \mathcal{N}\left(\theta_n; \frac{z_{mn}^i}{A_{mn}}, \frac{c_{mn}^i}{A_{mn}^2}\right) \tag{17}$$

$$z_{mn}^i \triangleq y_m - \sum_{q \neq n} A_{mq} \mu_{qm}^i \tag{18}$$

$$c_{mn}^i \triangleq \sigma^2 + \sum_{q \neq n} A_{mq}^2 v_{qm}^i. \tag{19}$$

The updates μ_{mn}^{i+1} and v_{mn}^{i+1} can then be calculated from

$$\nu_{\theta_n \to g_m}^{i+1}(\theta_n) \propto \nu_{f_n \to \theta_n}(\theta_n) \prod_{l \neq m} \nu_{g_l \to \theta_n}^i(\theta_n) \tag{20}$$

where, using (16), the product term in (20) is

$$\propto \mathcal{N}\left(\theta_n; \frac{\sum_{l \neq m} A_{ln} z_{ln}^i / c_{ln}^i}{\sum_{l \neq m} A_{ln}^2 z_{ln}^i / c_{ln}^i}, \frac{1}{\sum_{l \neq m} A_{ln}^2 / c_{ln}^i}\right). \tag{21}$$

Assuming that the values A_{ln}^2 satisfy

$$\sum_{l \neq m} A_{ln}^2 \approx \sum_{l=1}^M A_{ln}^2 \approx 1, \tag{22}$$

which occurs, e.g., when M is large and $\{A_{ln}\}$ are generated i.i.d. with variance $1/M$, we have $c_{ln}^i \approx c_n^i \triangleq \frac{1}{M} \sum_{m=1}^M c_{mn}^i$, and thus (20) is well approximated by

$$\nu_{\theta_n \to g_m}^{i+1}(\theta_n) \propto \left(\lambda_n \mathcal{N}\left(\theta_n; 0, \sigma_n^2\right) + (1 - \lambda_n)\delta(\theta_n)\right) \times \mathcal{N}\left(\theta_n; \xi_{nm}^i, c_n^i\right) \tag{23}$$

$$\xi_{nm}^i \triangleq \sum_{l \neq m} A_{ln} z_{ln}^i. \tag{24}$$

In this case, the mean and variance of $\nu_{\theta_n \to g_m}^{i+1}(\cdot)$ become

$$\mu_{nm}^{i+1} = \alpha_n\left(c_n^i\right) \xi_{nm}^i / \left(1 + \gamma_{nm}^i\right) \tag{25}$$

$$v_{nm}^{i+1} = \gamma_{nm}^i \left(\mu_{nm}^{i+1}\right)^2 + \mu_{nm}^{i+1} c_n^i / \xi_{nm}^i \tag{26}$$

$$\gamma_{nm}^i \triangleq \beta_n\left(c_n^i\right) \exp\left(-\zeta_n\left(c_n^i\right)\left(\xi_{nm}^i\right)^2\right) \tag{27}$$

where

$$\alpha_n(c) \triangleq \left(c/\sigma_n^2 + 1\right)^{-1}$$

$$\beta_n(c) \triangleq (-1 + 1/\lambda_n)\sqrt{1 + \sigma_n^2/c}$$

$$\zeta_n(c) \triangleq \left(2c\left(1 + c/\sigma_n^2\right)\right)^{-1}.$$

According to the sum-product algorithm, $\hat{p}^{i+1}(\theta_n \,|\, \boldsymbol{y})$, the posterior on θ_n after SSR's ith-LBP iteration, obeys

$$\hat{p}^{i+1}(\theta_n \,|\, \boldsymbol{y}) \propto \nu_{f_n \to \theta_n}(\theta_n) \prod_{l=1}^M \nu_{g_l \to \theta_n}^i(\theta_n), \tag{28}$$

whose mean and variance determine the ith-iteration MMSE estimate of θ_n and its variance, respectively. Noting that the difference between (28) and (20) is only the inclusion of the mth product term, these MMSE quantities become

$$\mu_n^{i+1} = \alpha_n\left(c_n^i\right) \xi_n^i / \left(1 + \gamma_n^i\right) \tag{29}$$

$$v_n^{i+1} = \gamma_n^i \left(\mu_n^{i+1}\right)^2 + \mu_n^{i+1} c_n^i / \xi_n^i \tag{30}$$

$$\xi_n^i \triangleq \sum_{l=1}^M A_{ln} z_{ln}^i \tag{31}$$

$$\gamma_n^i \triangleq \beta_n\left(c_n^i\right)\exp\left(-\zeta_n\left(c_n^i\right)\left(\xi_n^i\right)^2\right). \tag{32}$$

Similarly, the posterior on s_n after the ith iteration obeys

$$\hat{p}^{i+1}(s_n \,|\, \boldsymbol{y}) \propto \nu^i_{f_n\to s_n}(s_n)\nu_{h_n\to s_n}(s_n) \tag{33}$$

where

$$\nu^i_{f_n\to s_n}(s_n) \propto \int_{\theta_n} f_n(\theta_n, s_n)\prod_{l=1}^{M}\nu^i_{g_l\to\theta_n}(\theta_n). \tag{34}$$

Since $f_n(\theta_n, s_n) = s_n\mathcal{N}(\theta_n; 0, \sigma_n^2) + (1-s_n)\delta(\theta_n)$, it can be seen that the corresponding log-likelihood ratio (LLR) is

$$L_n^{i+1} \triangleq \ln\frac{\nu^i_{f_n\to s_n}(s_n=1)}{\nu^i_{f_n\to s_n}(s_n=0)} = \ln\frac{1-\lambda_n^i}{\gamma_n^i\lambda_n^i}. \tag{35}$$

Clearly, the LLR L_n^{i+1} and the likelihood function $\nu^i_{f_n\to s_n}(\,\cdot\,)$ express the same information, but in different ways.

The procedure described thus far updates $\mathcal{O}(MN)$ variables per LBP iteration, which is impractical since MN can be very large. In [24], Donoho, Maleki, and Montanari proposed, for the i.i.d. case, further approximations that yield a "first-order" AMP algorithm that allows the update of only $\mathcal{O}(N)$ variables per LBP iteration, essentially by approximating the *differences* among the outgoing means/variances of the $g_m(\,\cdot\,)$ nodes (i.e., z_{mn}^i and c_{mn}^i) as well as the differences among the outgoing means/variances of the θ_n nodes (i.e., μ_n^i and v_n^i). The resulting algorithm was then rigorously analyzed by Bayati and Montanari in [16]. We now summarize a straightforward extension of the i.i.d. AMP algorithm from [24] to the case of an independent but non-identical Bernoulli–Gaussian prior (13):

$$\xi_n^i = \sum_{m=1}^{M} A_{mn}z_m^i + \mu_n^i \tag{36}$$

$$\mu_n^{i+1} = F_n\left(\xi_n^i; c^i\right) \tag{37}$$

$$v_n^{i+1} = G_n\left(\xi_n^i; c^i\right) \tag{38}$$

$$z_m^{i+1} = y_m - \sum_{n=1}^{N}A_{mn}\mu_n^i + \frac{z_m^i}{M}\sum_{n=1}^{N}F_n'\left(\xi_n^i; c^i\right) \tag{39}$$

$$c^{i+1} = \sigma^2 + \frac{1}{M}\sum_{n=1}^{N}v_n^{i+1} \tag{40}$$

where $F_n(\,\cdot;\cdot\,)$, $G_n(\,\cdot;\cdot\,)$, and $F_n'(\,\cdot;\cdot\,)$ are defined as

$$F_n(\xi; c) = \frac{\alpha_n(c)}{1+\tau_n(\xi;c)}\xi \tag{41}$$

$$G_n(\xi; c) = \tau_n(\xi;c)F_n(\xi;c)^2 + c\,\xi^{-1}F_n(\xi;c) \tag{42}$$

$$F_n'(\xi; c) = \frac{\alpha_n(c)\left[1+\tau_n(\xi;c)\left(1+2\zeta(c_n)\xi^2\right)\right]}{[1+\tau_n(\xi;c)]^2} \tag{43}$$

$$\tau_n(\xi; c) = \beta_n(c)\exp\left(-\zeta_n(c)\xi^2\right). \tag{44}$$

For the first turbo iteration (i.e., $t=1$), we initialize AMP using $z_m^{i=1} = y_m$, $\mu_n^{i=1} = 0$, and $c_n^{i=1} \gg \sigma_n^2$ for all m, n. For subsequent turbo iterations (i.e., $t>1$), we initialize AMP by setting $z_m^{i=1}, \mu_n^{i=1}, c_n^{i=1}$ equal to the final values of z_m^i, μ_n^i, c_n^i generated by AMP at the *previous* turbo iteration. We terminate the AMP iterations as soon as either $\|\boldsymbol{\mu}^i - \boldsymbol{\mu}^{i-1}\|_2 < 10^{-5}$ or a maximum of 10 AMP iterations have elapsed. Similarly, we terminate the turbo iterations as soon as either $\|\boldsymbol{\mu}^{(t)} - \boldsymbol{\mu}^{(t-1)}\|_2 < 10^{-5}$ or a maximum of 10 turbo iterations have elapsed. The final value of $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_N]^T$ is output as the signal estimate $\hat{\boldsymbol{\theta}}$.

D. Learning the Statistical Parameters

We now describe how the precisions $\{\rho_j\}$ are learned. First, we recall that ρ_j describes the *a priori* precision on the active coefficients at the jth level, i.e., on $\{\theta_n\}_{n\in\mathcal{S}_j}$, where the corresponding index set $\mathcal{S}_j \triangleq \{n\in\mathcal{W}_j : s_n = 1\}$ is of size $K_j \triangleq |\mathcal{S}_j|$. Furthermore, we recall that the prior on ρ_j was chosen as in (4). Thus, *if* we had access to the true values $\{\theta_n\}_{n\in\mathcal{S}_j}$, then (2) implies that

$$p(\theta_n \,|\, n\in\mathcal{S}_j) = \mathcal{N}(\theta_n; 0, \rho_j) \tag{45}$$

which implies[4] that the posterior on ρ_j would take the form of $\mathrm{Gamma}(\hat{a}_j, \hat{b}_j)$ where $\hat{a}_j = a_j + \frac{1}{2}K_j$ and $\hat{b}_j = b_j + \frac{1}{2}\sum_{n\in\mathcal{S}_j}\theta_n^2$. In practice, we do not have access to the true values $\{\theta_n\}_{n\in\mathcal{S}_j}$ nor to the set $\mathcal{S}_j$, and thus we propose to build surrogates from the SSR outputs. In particular, to update ρ_j after the tth turbo iteration, we employ

$$\mathcal{S}_j^{(t)} \triangleq \{n\in\mathcal{W}_j : L_n^{(t)} > 0\} \tag{46}$$

$$K_j^{(t)} \triangleq \left|\mathcal{S}_j^{(t)}\right|, \tag{47}$$

and $\{\mu_n^{(t)}\}_{n\in\mathcal{S}_j^{(t)}}$, where $L_n^{(t)}$ and $\mu_n^{(t)}$ denote the final LLR on s_n and the final MMSE estimate of θ_n, respectively, at the tth turbo iteration. These choices imply the hyperparameters

$$\hat{a}_j^{(t+1)} = a_j + \frac{1}{2}K_j^{(t)} \tag{48}$$

$$\hat{b}_j^{(t+1)} = b_j + \frac{1}{2}\sum_{n\in\mathcal{S}_j^{(t)}}\left(\mu_n^{(t)}\right)^2. \tag{49}$$

Finally, to perform SSR at turbo iteration $t+1$, we set the variances $\{\sigma_n^2\}_{n\in\mathcal{W}_j}$ equal to the inverse of the expected precisions, i.e., $1/\mathrm{E}\{\rho_j\} = \hat{b}_j^{(t+1)}/\hat{a}_j^{(t+1)}$. The noise variance σ^2 is learned similarly from the SSR-estimated residual.

Next, we describe how the transition probabilities $\{\pi_j^{11}\}$ are learned. First, we recall that π_j^{11} describes the probability that a child at level $j+1$ is active (i.e., $s_n = 1$) given that his parent (at level j) is active. Furthermore, we recall that the prior on π_j^{11} was chosen as in (7). Thus, *if* we knew that there were K_j active coefficients at level j, of which C_j had active children, then[5] the posterior on π_j^{11} would take the form of $\mathrm{Beta}(\hat{c}_j, \hat{d}_j)$, where $\hat{c}_j = c_j + C_j$ and $\hat{d}_j = d_j + K_j - C_j$. In practice, we do not have access to the true values of K_j and C_j, and thus we build surrogates from the SSR outputs. In particular, to update π_j^{11} after the tth turbo iteration, we approximate $s_n = 1$ by the event $L_n^{(t)} > 0$, and based on this approximation set $K_j^{(t)}$ (as

[4]This posterior results because the chosen prior is conjugate [17] for the likelihood in (45).

[5]This posterior results because the chosen prior is conjugate to the Bernoulli likelihood [17].

in (47)) and $C_j^{(t)}$. The corresponding hyperparameters are then updated as

$$\hat{c}_j^{(t+1)} = c_j + C_j^{(t)} \tag{50}$$

$$\hat{d}_j^{(t+1)} = d_j + K_j^{(t)} - C_j^{(t)}. \tag{51}$$

Finally, to perform SSR at turbo iteration $t+1$, we set the transition probabilities π_j^{11} equal to the expected value $\hat{c}_j^{(t+1)}/(\hat{c}_j^{(t+1)} + \hat{d}_j^{(t+1)})$. The parameters π_0^1, π_{-1}^1, and $\{\pi_j^{00}\}$ are learned similarly.

E. The Two-State Gaussian-Mixture Model

Until now, we have focused on the BG signal model (2). In this section, we describe the modifications needed to handle the GM model

$$p(\theta_n \,|\, s_n) = s_n \mathcal{N}\left(\theta_n; 0, \sigma_{n,\mathsf{L}}^2\right) + (1-s_n)\mathcal{N}\left(\theta_n; 0, \sigma_{n,\mathsf{S}}^2\right) \tag{52}$$

where $\sigma_{n,\mathsf{L}}^2$ denotes the variance of "large" coefficients and $\sigma_{n,\mathsf{S}}^2$ denotes the variance of "small" ones. For either the BG or GM prior, AMP is performed using the steps (36)–(40). For the BG case, the functions $F_n(\cdot;\cdot)$, $G_n(\cdot;\cdot)$, $F_n'(\cdot;\cdot)$, and $\tau_n(\cdot;\cdot)$ are given in (41)–(44), whereas for the GM case, they take the form

$$F_n(\xi; c) = \frac{\bar{\alpha}_{n,\mathsf{L}}(c) + \bar{\alpha}_{n,\mathsf{S}}(c)\bar{\tau}_n(\xi,c)}{1+\bar{\tau}_n(\xi,c)}\xi \tag{53}$$

$$G_n(\xi; c) = \frac{\bar{\tau}_n(\xi,c)\xi^2[\bar{\alpha}_{n,\mathsf{L}}(c) - \bar{\alpha}_{n,\mathsf{S}}(c)]^2}{[1+\bar{\tau}_n(\xi,c)]^2} + c\,\xi^{-1}F_n(\xi;c) \tag{54}$$

$$F_n'(\xi; c) = \frac{\bar{\tau}_n(\xi;c)\bar{\alpha}_{n,\mathsf{L}}(c)\left(1+\bar{\tau}_n(\xi;c) - 2\xi^2\bar{\zeta}_n(c)\right)}{[1+\bar{\tau}_n(\xi;c)]^2} + \frac{\bar{\alpha}_{n,\mathsf{S}}(c)\left(1+\bar{\tau}_n(\xi;c) + 2\xi^2\bar{\zeta}_n(c)\bar{\tau}_n(\xi;c)\right)}{[1+\bar{\tau}_n(\xi;c)]^2} \tag{55}$$

$$\bar{\tau}_n(\xi, c) = \bar{\beta}_n(c)\exp\left(-\bar{\zeta}_n(c)\xi^2\right) \tag{56}$$

where

$$\bar{\alpha}_{n,\mathsf{L}}(c) \triangleq \frac{\sigma_{n,\mathsf{L}}^2}{c+\sigma_{n,\mathsf{L}}^2} \tag{57}$$

$$\bar{\alpha}_{n,\mathsf{S}}(c) \triangleq \frac{\sigma_{n,\mathsf{S}}^2}{c+\sigma_{n,\mathsf{S}}^2} \tag{58}$$

$$\bar{\beta}_n(c) \triangleq \frac{1-\lambda_n}{\lambda_n}\sqrt{\frac{c+\sigma_{n,\mathsf{L}}^2}{c+\sigma_{n,\mathsf{S}}^2}} \tag{59}$$

$$\bar{\zeta}_n(c) \triangleq \frac{\sigma_{n,\mathsf{L}}^2 - \sigma_{n,\mathsf{S}}^2}{2\left(c+\sigma_{n,\mathsf{L}}^2\right)\left(c+\sigma_{n,\mathsf{S}}^2\right)}. \tag{60}$$

Likewise, for the BG case, the extrinsic LLR is given by (35), whereas for the GM case, it becomes

$$L_n^{i+1} \triangleq \ln\frac{\nu_{f_n\to s_n}^i(s_n=1)}{\nu_{f_n\to s_n}^i(s_n=0)} = \ln\frac{1-\lambda_n^i}{\bar{\tau}_n\left(\xi_n^i; c_n^i\right)\lambda_n^i}. \tag{61}$$

IV. Numerical Results

A. Setup

The proposed turbo[6] approach to compressive imaging was compared to several other tree-sparse reconstruction algorithms: ModelCS [3], HMT+IRWL1 [4], MCMC [5], VB [6]; and to several simple-sparse reconstruction algorithms: CoSaMP [8], SPGL1 [26], and BG AMP. All numerical experiments were performed on 128×128 (i.e., $N = 16384$) grayscale images. Unless otherwise mentioned, $J = 4$-level 2D Haar wavelet decomposition was used, yielding $8^2 = 64$ approximation coefficients and $3\times8^2 = 192$ individual Markov trees. In all cases, the measurement matrix $\boldsymbol{\Phi}$ had i.i.d. Gaussian entries. Unless otherwise specified, $M = 5000$ noiseless measurements were used. We used normalized mean squared error (NMSE) $\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2/\|\boldsymbol{x}\|_2^2$ as the performance metric.

We now describe how the hyperparameters were chosen for the proposed Turbo schemes. Below, we use N_j to denote the total number of wavelet coefficients at level j, and N_{-1} to denote the total number of approximation coefficients. For both Turbo-BG and Turbo-GM, the Beta hyperparameters were chosen so that $c+d = N_0$, $\underline{c}+\underline{d} = N_{-1}$ and $c_j+d_j = N_j\,\forall j$ with $\mathrm{E}\{p_0^1\} = 1/N$, $\mathrm{E}\{p_{-1}^1\} = 1-10^{-6}$, $\mathrm{E}\{p_j^{00}\} = 1/N\,\forall j$, and $\mathrm{E}\{p_j^{11}\} = 0.5\,\forall j$. These informative hyperparameters are similar to the "universal" recommendations in [27] and, in fact, identical to the ones suggested in the MCMC work [5]. For Turbo-BG, the hyperparameters for the signal precisions $\{\sigma_n^{-2}\}_{n\in\mathcal{W}_j}$ were set to $a_j = 1\,\forall j$ and $[b_{-1},\ldots,b_3] = [10,1,1,0.1,0.1]$. This choice is motivated by the fact that wavelet coefficient magnitudes are known to decay exponentially with scale j (e.g., [27]). Meanwhile, the hyperparameters for the noise precision σ^{-2} were set to $(a,b) = (1,10^{-6})$. Although the measurements were noiseless, we allow Turbo-BG a nonzero noise variance in order to make up for the fact that the wavelet coefficients are not exactly sparse, as assumed by the BG signal model. (We note that the same was done in the BG-based work [5], [6].) For Turbo-GM, the hyperparameters $(a_{j,\mathsf{L}}, b_{j,\mathsf{L}})$ for the signal precisions $\{\sigma_{n,\mathsf{L}}^{-2}\}_{n\in\mathcal{W}_j}$ were set at the values of (a_j,b_j) for the BG case, while the hyperparameters $(a_{j,\mathsf{S}}, b_{j,\mathsf{S}})$ for $\{\sigma_{n,\mathsf{S}}^{-2}\}_{n\in\mathcal{W}_j}$ were set as $a_{j,\mathsf{S}} = 1$ and $b_{j,\mathsf{S}} = 10^{-6}b_{j,\mathsf{L}}$. Meanwhile, the noise variance σ^2 was assumed to be exactly zero, because the GM signal prior is capable of modeling non-sparse wavelet coefficients.

For MCMC [5], the hyperparameters were set in accordance with the values described in [5]; the values of $c, d, \{c_j, d_j\}$ are same as the ones used for the proposed Turbo-BG scheme, while $a = a_j = b = b_j = 10^{-6}\,\forall j$. For VB, the same hyperparameters as MCMC were used except for $a_j = 2\times10^9$ and $b_j = 10^{10}\,\forall j$, which were the default values of hyperparameters used in the publicly available code.[7] We experimented with the values for both MCMC and VB and found that the default values indeed seem to work best. For example, if one swaps the

[6]An implementation of our algorithm can be downloaded from http://www.ece.osu.edu/~schniter/turboAMPimaging

[7]http://people.ee.duke.edu/~lcarin/BCS.html

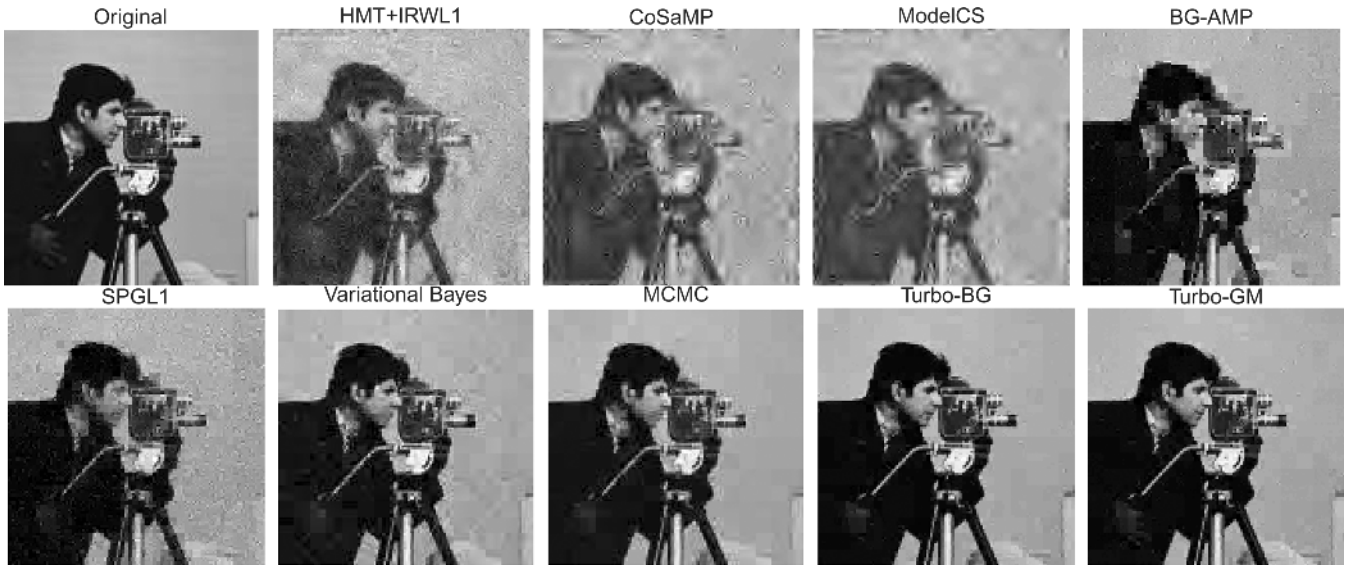

Fig. 4. Reconstruction from $M = 5000$ observations of a 128×128 (i.e., $N = 16384$) section of the cameraman image using i.i.d. Gaussian $\mathbf{\Phi}$.

TABLE I
NMSE AND RUNTIME AVERAGED OVER 591 IMAGES

Algorithm	NMSE (dB)	Computation Time (sec)
HMT+IRWL1	-14.37	363
CoSaMP	-16.90	25
ModelCS	-17.45	117
BG-AMP	-17.84	68
SPGL1	-18.06	536
VB	-19.04	107
MCMC	-20.10	742
Turbo-BG	-20.49	51
Turbo-GM	-20.74	51

(a_j, b_j) hyperparameters between VB and MCMC, then the average performance of VB and MCMC degrade by 1.69 dB and 1.55 dB, respectively, relative to the values reported in Table I.

For both the CoSaMP and ModelCS algorithms, the principal tuning parameter is the assumed number of non-zero coefficients. For both ModelCS (which is based on CoSaMP) and CoSaMP itself, we used the Rice University codes,[8] which include a genie-aided mechanism to compute the number of active coefficients from the original image. However, since we observed that the algorithms perform somewhat poorly under that tuning mechanism, we instead ran (for each image) multiple reconstructions with the number of active coefficients varying from 200 to 2000 in steps of 100, and reported the result with the best NMSE. The number of active coefficients chosen in this manner was usually much smaller than that chosen by the Rice procedure.

To implement BG-AMP, we used the AMP scheme described in Section III-C with the hyperparameter learning scheme described in Section III-D; HMT structure was not exploited. For this, we assumed that the priors on variance σ_n^2 and activity λ_n were identical over the coefficient index n, and assigned Gamma and Beta hyperpriors of $(a, b) = (10^{-10}, 10^{-10})$ and $(c, d) = (0.1N, 0.9N)$, respectively.

[8]http://dsp.rice.edu/software/model-based-compressive-sensing-toolbox

Fig. 5. A sample image from each of the 20 types in the Microsoft database. Image statistics were found to vary significantly from one type to another.

For HMT+IRWL1, we ran code provided by the authors with default settings. For SPGL1,[9] the residual variance was set to 0, and all parameters were set at their defaults.

B. Results

Fig. 4 shows a 128×128 section of the "cameraman" image along with the images recovered by the various algorithms. Qualitatively, we see that CoSaMP, which leverages only simple sparsity, and ModelCS, which models persistence-across-scales (PAS) through a deterministic tree structure, both perform relatively poorly. HMT+IRWL1 also performs relatively poorly, due to (we believe) the ad-hoc manner in which the HMT structure was exploited via iteratively re-weighted ℓ_1. The BG-AMP and SPGL1 algorithms, neither of which attempt to exploit PAS, perform better. The HMT-based schemes (VB, MCMC, Turbo-GM, and Turbo-GM) all perform significantly better, with the Turbo schemes performing the best.

For a quantitative comparison, we measured average performance over a suite of images in a *Microsoft Research Object Class Recognition* database[10] that contains 20 types of images (see Fig. 5) with roughly 30 images of each type. In particular, we computed the average NMSE and average runtime on a 2.5-GHz PC, for each image type. These results are reported in Figs. 6 and 7, and the global averages (over all 591 images) are reported in Table I. From the table, we observe that the

[9]http://www.cs.ubc.ca/labs/scl/spgl1/index.html

[10]We used 128×128 images extracted from the "Pixel-wise labelled image database v2" at http://research.microsoft.com/en-us/projects/objectclassrecognition. What we refer to as an "image type" is a "row" in this database.

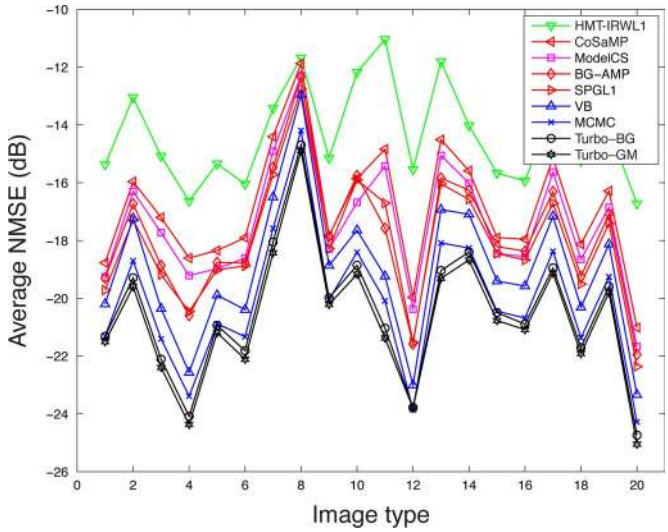

Fig. 6. Average NMSE for each image type.

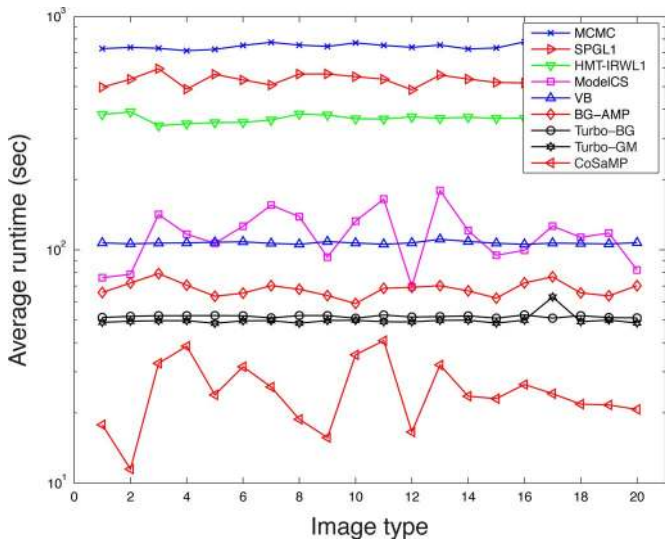

Fig. 7. Average runtime for each image type.

proposed Turbo algorithms outperform all the other tested algorithms in terms of reconstruction NMSE, but are beaten only by CoSaMP in speed.[11] Between the two Turbo algorithms, we observe that Turbo-GM slightly outperforms Turbo-BG in terms of reconstruction NMSE, while taking the same runtime. In terms of NMSE performance, the closest competitor to the Turbo schemes is MCMC,[12] whose NMSE is 0.39 dB worse than Turbo-BG and 0.65 dB worse than Turbo-GM. The good NMSE performance of MCMC comes at the cost of complexity, though: MCMC is 15 times slower than the Turbo schemes. The second closest NMSE-competitor is VB,

[11]The CoSaMP runtimes must be interpreted with caution, because the reported runtimes correspond to a single reconstruction, whereas in practice multiple reconstructions may be needed to determined the best value of the tuning parameter.

[12]The MCMC results reported here are for the default settings: 100 MCMC iterations and 200 burn-in iterations. Using 500 MCMC iterations and 200 burn-in iterations, we obtained an average NMSE of -20.22 dB (i.e., 0.12 dB better) at an average runtime of 1958 s (i.e., $2.6\times$ slower).

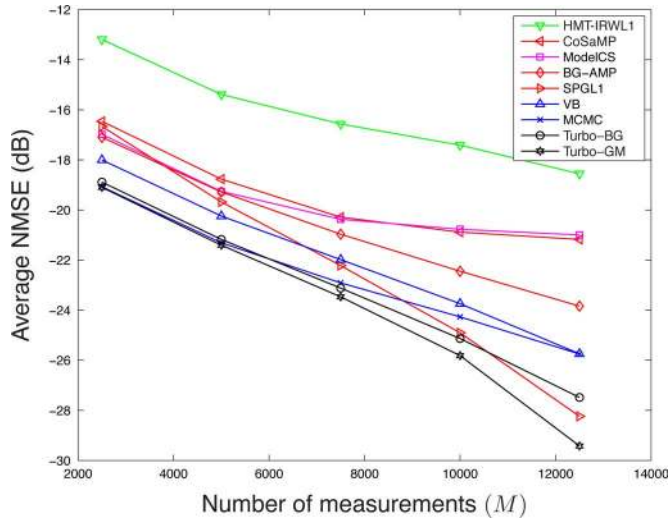

Fig. 8. Average NMSE for images of type 1.

showing performance 1.5 dB worse than Turbo-BG and 1.7 dB worse than Turbo-GM. Even with this sacrifice in performance, VB is still twice as slow as the Turbo schemes. Among the algorithms that do not exploit PAS, we see that SPGL1 offers the best NMSE performance, but is by far the slowest (e.g., 20 times slower than CoSaMP). Meanwhile, CoSaMP is the fastest, but shows the worst NMSE performance (e.g., 1.16 dB worse than SPGL1). BG-AMP strikes an excellent balance between the two: its NMSE is only 0.22 dB away from SPGL1, whereas it takes only 2.7 times as long as CoSaMP. However, by combining the AMP algorithm with HMT structure via the turbo approach, it is possible to significantly improve NMSE while simultaneously decreasing the runtime. The reason for the complexity decrease is twofold. First, the HMT structure helps the AMP and parameter-learning iterations to converge faster. Second, the HMT steps are computationally negligible relative to the AMP steps: when, e.g., $M = 5000$, the AMP portion of the turbo iteration takes approximately 6 s while the HMT portion takes 0.02 s.

We also studied NMSE and compute time as a function of the number of measurements, M. For this study, we examined images of Type 1 at $M = 2500$, 5000, 7500, 10 000, 12 500. In Fig. 8, we see that Turbo-GM offers the uniformly best NMSE performance across M. However, as M decreases, there is little difference between the NMSEs of Turbo-GM, Turbo-BG, and MCMC. As M increases, though, we see that the NMSEs of MCMC and VB converge, but that they are significantly outperformed by Turbo-GM, Turbo-BG, and—somewhat surprisingly—SPGL1. In fact, at $M = 12\,500$, SPGL1 outperforms Turbo-BG, but not Turbo-GM. However, the excellent performance of SPGL1 at these M comes at the cost of very high complexity, as evident in Fig. 9.

We have used $J = 4$-level wavelet decomposition so far. We have also studied the reconstruction performance as a function of wavelet decomposition depth for images of Type 1. Fig. 10 shows the reconstruction error from $M = 5000$ observations for the Bayesian algorithms. We observe minimal improvement in performance of the algorithms beyond $J = 4$.

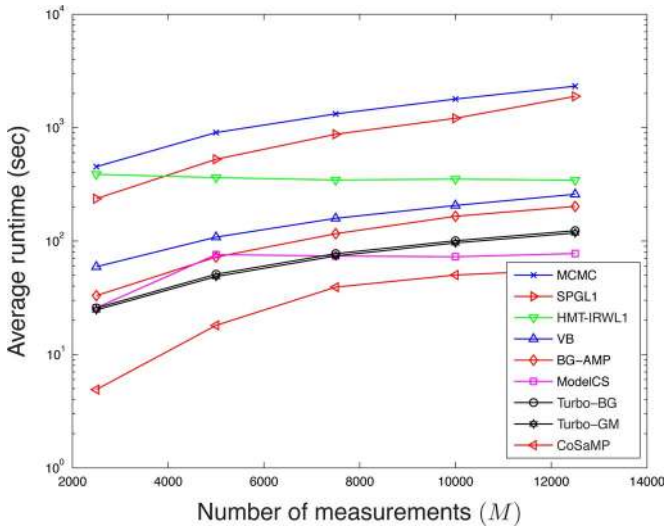

Fig. 9. Average runtime for images of type 1.

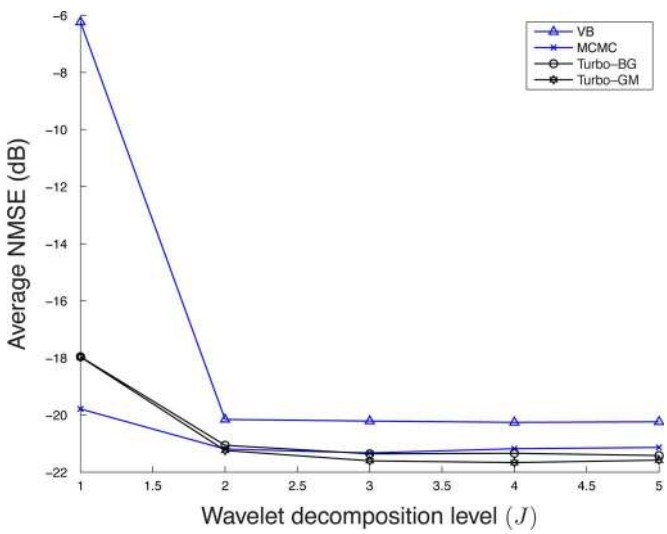

Fig. 10. Average NMSE for images of type 1.

V. Conclusion

We proposed a new approach to HMT-based compressive imaging based on loopy belief propagation, leveraging a turbo message passing schedule and the AMP algorithm of Donoho, Maleki, and Montanari. We then tested our algorithm on a suite of 591 natural images and found that it outperformed the state-of-the-art approach (i.e., variational Bayes) while halving its runtime.

References

[1] J. Romberg, "Imaging via compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 14–20, Mar. 2008.

[2] S. Mallat, *A Wavelet Tour of Signal Processing*, 3rd ed. San Diego, CA: Academic, 2008.

[3] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1982–2001, Apr. 2010.

[4] M. F. Duarte, M. B. Wakin, and R. G. Baraniuk, "Wavelet-domain compressive signal reconstruction using a hidden Markov tree model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Las Vegas, NV, Apr. 2008, pp. 5137–5140.

[5] L. He and L. Carin, "Exploiting structure in wavelet-based Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 57, no. 9, pp. 3488–3497, Sep. 2009.

[6] L. He, H. Chen, and L. Carin, "Tree-structured compressive sensing with variational Bayesian analysis," *IEEE Signal Process. Lett.*, vol. 17, no. 3, pp. 233–236, Mar. 2010.

[7] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009.

[8] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, 2009.

[9] M. S. Crouse, R. D. Nowak, and R. G. Baraniuk, "Wavelet-based statistical signal processing using hidden Markov models," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 886–902, Apr. 1998.

[10] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, 2nd ed. New York: Springer, 2004.

[11] B. J. Frey and D. J. C. MacKay, "A revolution: Belief propagation in graphs with cycles," in *Adv. Neural Inf. Process. Syst.*, M. Jordan, M. S. Kearns, and S. A. Solla, Eds. Cambridge, MA: MIT Press, 1998, pp. 479–485.

[12] P. Schniter, "Turbo reconstruction of structured sparse signals," in *Proc. Conf. Inf. Sci. Syst.*, Princeton, NJ, Mar. 2010, pp. 1–5.

[13] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufman, 1988.

[14] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[15] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," in *Proc. Nat. Acad. Sci.*, Nov. 2009, vol. 106, no. 45, pp. 18 914–18 919.

[16] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 764–785, Feb. 2011.

[17] , A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, Eds., *Bayesian Data Analysis*, 2nd ed. Boca Raton, FL: CRC Press, 2003.

[18] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Art. Intell.*, vol. 42, pp. 393–405, 1990.

[19] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *Int. J. Comput. Vis.*, vol. 40, no. 1, pp. 25–47, Oct. 2000.

[20] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's 'belief propagation' algorithm," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 140–152, Feb. 1998.

[21] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. New York: Cambridge Univ. Press, 2003.

[22] J. Boutros and G. Caire, "Iterative multiuser joint decoding: Unified framework and asymptotic analysis," *IEEE Trans. Inf. Theory*, vol. 48, no. 7, pp. 1772–1793, Jul. 2002.

[23] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Trans. Signal Process.*, vol. 58, no. 1, pp. 2269–2280, Jan. 2010.

[24] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. Inf. Theory Workshop*, Cairo, Egypt, Jan. 2010, pp. 1–5.

[25] S. Som, L. C. Potter, and P. Schniter, "On approximate message passing for reconstruction of non-uniformly sparse signals," in *Proc. Nat. Aerosp. Electron. Conf.*, Dayton, OH, Jul. 2010, pp. 223–229.

[26] E. van den Berg and M. P. Friedlander, "Probing the Pareto frontier for basis pursuit solutions," *SIAM J. Sci. Comput.*, vol. 31, no. 2, pp. 890–912, 2008.

[27] J. K. Romberg, H. Choi, and R. G. Baraniuk, "Bayesian tree-structured image modeling using wavelet-domain hidden Markov models," *IEEE Trans. Image Process.*, vol. 10, no. 7, pp. 1056–1068, Jul. 2001.

[28] S. Som, L. C. Potter, and P. Schniter, "Compressive imaging using approximate message passing and a Markov-tree prior," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 2010, pp. 243–247.

Subhojit Som received the B.Tech. (Hons.) degree from the Indian Institute of Technology, Kharagpur, in 2002 and the M.S. and Ph.D. degrees from The Ohio State University, Columbus, in 2006 and 2010, respectively.

From 2002 to 2004, he was a Design Engineer at Texas Instruments in the Broadband Communications Group. Presently, he is a Postdoctoral Fellow at the School of Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta. His research interests include signal processing and machine learning.

Dr. Som was a recipient of the National Talent Search scholarship from NCERT, India, and Distinguished University Fellowship from The Ohio State University.

Philip Schniter received the B.S. and M.S. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 1992 and 1993, respectively, and the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY, in 2000.

From 1993 to 1996, he was a Systems Engineer with Tektronix Inc., Beaverton, OR. After receiving the Ph.D. degree in 2000, he joined the Department of Electrical and Computer Engineering at The Ohio State University, Columbus, where he is currently an Associate Professor and a member of the Information Processing Systems (IPS) Lab. In 2008–2009, he was a Visiting Professor at Eurecom, Sophia Antipolis, France, and Supélec, Gif-sur-Yvette, France. His areas of interest include statistical signal processing, wireless communications and networks, and machine learning

Dr. Schniter received the National Science Foundation CAREER Award in 2003.