Variational Bayesian Image Super-Resolution with GPU Acceleration

Giannis Chantas

Department of Informatics and Telecommunications, TEI of Larissa,
41110 Larissa, Greece

Abstract. With the term super-resolution we refer to the problem of reconstructing an image of higher resolution than that of unregistered and degraded observations. Typically, the reconstruction is based on the inversion of the observation generation model. In this paper this problem is formulated using a variational Bayesian inference framework and an edge-preserving image prior. A novel super-resolution algorithm is proposed, which is derived using a modification of the constrained variational inference methodology which infers the posteriors of the model variables and selects automatically all the model parameters. This algorithm is very intensive computationally, thus, it is accelerated by harnessing the computational power of a graphics processor unit (GPU). Examples are presented with both synthetic and real images that demonstrate the advantages of the proposed framework as compared to other state-of-the-art methods.

Keywords: Variational Inference, Bayesian Super-Resolution, Image Prior Student's-t distribution, CUDA, GPGPU.

1 Introduction

The problem of super-resolution is defined as obtaining an image with enhanced resolution from a set of lower resolution unregistered degraded images. The super-resolution problem has a long history. In this paper we will not attempt overview it; for this purpose the interested reader is referred to [1], [2] and [3]. An important category of methodologies used for this problem formulates it as an ill posed reconstruction problem. Thus, prior information is introduced (regularization) to complement the available observations and reconstruct the super-resolved image.

One powerful stochastic methodology to apply regularized reconstruction to inverse problems is Bayesian inference [4]. The main advantage of Bayesian inference is that the unknown image is treated as a random variable and the posterior pdf given the observations is found. Thus, unlike the maximum a posteriori (MAP) estimation approach, which only provides point estimates, Bayesian inference provides variance information also about the estimate [4]. However, the application of Bayesian inference is difficult when complex models and large data sets are used. Therefore, MAP estimation has been much more popular for image super-resolution problems [3].

K. Diamantaras, W. Duch, L.S. Iliadis (Eds.): ICANN 2010, Part I, LNCS 6352, pp. 480–489, 2010.

In this work the Bayesian inference framework using the variational approximation is applied for the first time to the image super-resolution problem. In this formulation a spatially varying edge-preserving image prior is used. This prior has been used previously with success for the image restoration problem in a Bayesian inference framework [8]. Bayesian inference is easier for image restoration than for super-resolution because the imaging operator in restoration is a simple convolutional operator. In contrast, in super-resolution the imaging operator is more complex and is not convolutional [3]. Thus, for the super-resolution problem a similar in spirit prior was applied only in a MAP framework [5].

Another novel aspect of this work is the use of graphics processor unit (GPU) to speed up the proposed super-resolution algorithm for large images. Specifically, a parallel CUDA [10] implementation of the linear solver in this algorithm was used to speed up the computations required.

The rest of this paper is organized as follows. In Sect. 2 and 3 the imaging model and the proposed image prior models are presented, respectively. In Sect. 4 the variational algorithm is derived. In Sect. 5 the implementation details of the GPU linear solver used to accelerate our algorithm, and the initialization method of the proposed algorithm are presented. In Sect. 6 experiments with synthetic and real data that demonstrate the properties of our algorithm are presented. Finally, in Sect. 7 conclusions and thoughts for future research are provided.

2 Imaging Model

In what follows for simplicity we use one-dimensional notation. A linear imaging model is assumed according to which P low-resolution images (observations) $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_P$ of size $N_L \times 1$ are produced by operating on the high-resolution image $\mathbf{x}$ of size $N_H \times 1$. Thus the decimation factor d can be defined as the ratio $d = N_H/N_L$. Each observation is produced by first translating and rotating the high-resolution image, then blurring and decimating it by the factor d. Lastly, a noise vector $n_k, k = 1, ..., P$ is added at each observation. This is mathematically expressed by the following P equations:

$$\mathbf{y}_k = \mathbf{B}_k(\zeta_k)\mathbf{x} + \mathbf{n}_k = \mathbf{DHW}(\zeta_k)\mathbf{x} + \mathbf{n}_k, \;\; k = 1, ..., P \;, \tag{1}$$

where $\mathbf{B}_k = \mathbf{B}_k(\theta_k) = \mathbf{DHW}(\theta_k)$, $\mathbf{D}$ is the known decimation matrix of size $N_L \times N_H$, $\mathbf{H}$ is the square $N_H \times N_H$ known convolutional blurring matrix that is assumed circulant and $\mathbf{W}(\zeta_k)$ is the $N_H \times N_H$ geometric transformation operator that translates and rotates the image. For the $k-th$ observation, $\zeta_k = [\gamma_k, \delta_k]$ is the parameter vector that contains the unknown rotation angle γ^k and the unknown translation parameter δ_k. Lastly, n_k is the $N_H \times N_H$ noise vector that is modeled as white Gaussian with the same (unknown) precision β for each observation, i.e. $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \beta\mathbf{I})$, where $\mathbf{0}$ and $\mathbf{I}$ are the $N_H \times N_H$ zero and identity matrices, respectively.

Let $\mathbf{y}$ be a $PN_L \times 1$ vector, containing the P low-resolution images: $\mathbf{y} = \left[\mathbf{y}_1^T, \mathbf{y}_2^T, \ldots, \mathbf{y}_P^T\right]^T$. Using this notation, the observations are given by:

$$\mathbf{y} = \mathbf{B}\mathbf{x} + \mathbf{n} \,, \tag{2}$$

where $\mathbf{n} = \left[\mathbf{n}_1^T, \mathbf{n}_2^T, \ldots, \mathbf{n}_P^T\right]^T$ and $\mathbf{B}$ is the $PN_L \times N_H$ imaging operator:

$$\mathbf{B} = \left[\mathbf{B}_1^T, \mathbf{B}_2^T, \ldots, \mathbf{B}_P^T\right]^T \,.$$

Lastly, in this work, to model the geometric transformation (rotation-translation) operation the Shannon (sinc) interpolator is used, which is linear and thus can be represented by the matrix $\mathbf{W}$.

3 Image Model

In what follows we introduce the image prior for the high-resolution image $\mathbf{x}$ of the imaging model described in Sect. 2. We first define K linear convolutional operators (filters) $\mathbf{Q}_1, \ldots, \mathbf{Q}_K$ of size $N_H \times N_H$. These filters are high-pass, such as first order differences in the vertical and horizontal direction. The filter outputs $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}_1^T, \ldots, \boldsymbol{\epsilon}_K^T)^T$ are produced according to the following K equations:

$$\boldsymbol{\epsilon}_l = \mathbf{Q}_l \mathbf{x}, \;\; l = 1, \ldots, K \,. \tag{3}$$

Then, it is assumed that all $\epsilon_l(i)$ for every i are iid zero mean Student's-t distributed with parameters λ_l and ν_l:

$$p(\epsilon_l(i)) = St(0; \lambda_l, \nu_l) = \frac{\Gamma(\nu_l/2 + 1/2)}{\Gamma(\nu_l/2)} \left(\frac{\lambda_l}{\nu_l}\right)^{\nu_l/2} \left(1 + \frac{\lambda_l \epsilon_l(i)^2}{\nu_l}\right)^{-\nu_l/2 - 1/2} ,$$

for $l = 1, \ldots, K$, where the parameters λ_l and ν_l are different for every filter but remain the constant as the spatial location i varies. To analyze the properties of the Student's-t distribution we write it down as the integral:

$$p(\epsilon_l(i)) = \int_{a_l(i)} p(\epsilon_l(i)|a_l(i)) p(a_l(i)) da_l(i) \tag{4}$$

where $a_l(i)$'s are random variables that are iid Gamma distributed $p(a_l(i)) = Gamma(\nu_l, \nu_l)$, and $p(\epsilon_l(i)|a_l(i)) = \mathcal{N}\left(0, a_l(i)^{-1}\right)$. The Student's-t distribution can be viewed as an *infinite mixture* of zero mean Gaussians [4] with different precisions. Thus, it can be heavy-tailed. Therefore, when used as a prior on the outputs of the $\mathbf{Q}_l$ high-pass filters it allows reconstructed images to have sharp edges. In contrast Gaussian based models have the tendency to smooth out edges.

4 Variational Inference

The variational methodology for Bayesian inference proposed in [8] for the imaging model in equation (2) is applicable only when the imaging operator $\mathbf{B}$ is convolutional and thus commutes with $\mathbf{Q}_l$, which is not the case for the super-resolution problem. In this section we present a modification of the variational algorithm in [8] which overcomes this difficulty, it is more general and can be applied to any linear imaging model of the form in (2).

As in [8], to perform Bayesian inference we introduce an alternative imaging model, which is derived by applying the operators $\mathbf{Q}_l$ to (2):

$$\mathbf{y} = \mathbf{B}\mathbf{Q}_l^{-1}\boldsymbol{\epsilon}_l + \mathbf{n},\ l = 1, \ldots, K, \tag{5}$$

where we have used the relationships $\mathbf{x} = \mathbf{Q}_l^{-1}\boldsymbol{\epsilon}_l, l = 1, \ldots, K$, stemming from the definitions of the $\boldsymbol{\epsilon}_l$'s in 3. Here, the key difference from [8] is that we avoided the multiplication with the operators $\mathbf{Q}_l$ but we embedded directly the relationships between the image and the filter outputs. This is the main reason for which the following derivation of the variational algorithm is novel and differs from that in [8].

With this imaging model, we work in the field of the filter outputs, and we treat $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}_1^T, \ldots, \boldsymbol{\epsilon}_K^T)^T$, where $\boldsymbol{\epsilon}_l = (\epsilon_l(1), \ldots, \epsilon_k(N_H))^T$, for $l = 1, \ldots, K$ and $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_K)$, where $\mathbf{a}_l = (a_l(1), \ldots, a_l(N_H))$, for $l = 1, \ldots, K$, as hidden variables. Then, according to Bayesian inference we find the posterior distributions for the hidden variables and estimate the parameters $\theta = \lambda_k, \nu_k$. The marginal of the observations $p(\mathbf{y}; \theta)$, which is required to find the posteriors of the hidden variables is hard to compute [4]. More specifically, the integral

$$p(\mathbf{y}) = \int_{\boldsymbol{\epsilon},\mathbf{a}} p(\mathbf{y}, \boldsymbol{\epsilon}, \mathbf{a}) d\boldsymbol{\epsilon} d\mathbf{a}\ , p(\mathbf{y}, \boldsymbol{\epsilon}, \mathbf{a}) = p(\mathbf{y}|\boldsymbol{\epsilon}) p(\boldsymbol{\epsilon}|\mathbf{a}) p(\mathbf{a})\ , \tag{6}$$

$$p(\mathbf{y}|\boldsymbol{\epsilon}) = \prod_{l=1}^{K} p(\mathbf{y}|\boldsymbol{\epsilon}_l), p(\mathbf{y}|\boldsymbol{\epsilon}_l) = N\left(\mathbf{B}\mathbf{Q}_l^{-1}\boldsymbol{\epsilon}_l, \beta\mathbf{I}\right)\ , \tag{7}$$

$$p(\boldsymbol{\epsilon}|\mathbf{a}) = \prod_{l=1}^{K}\prod_{i=1}^{N} p(\epsilon_l(i)|a_l(i)),\ p(\mathbf{a}) = \prod_{l=1}^{K}\prod_{i=1}^{N_H} p(a_l(i))\ ,$$

is intractable. Notice here that we have combined the K observation equations of (5) in one, by assuming that the data likelihood of a single observation is given by the product: $p(\mathbf{y}|\boldsymbol{\epsilon}) = \prod_{l=1}^{K} p(\mathbf{y}|\boldsymbol{\epsilon}_l)$. This idea stems from the principal of *opinion pooling* proposed in [6] that combines multiple probabilities.

The variational methodology, bypasses the difficulty of computing the integral in (6) and maximizes a *lower bound* $L(q(\boldsymbol{\epsilon}, \mathbf{a}), \theta)$ that can be found instead of the log-likelihood of the observations $\log p(y; \theta)$ [4]. This bound is obtained by subtracting from $\log p(\mathbf{y}; \theta)$ the Kullback-Leibler divergence, which is always positive, between an arbitrary $q(\boldsymbol{\epsilon}, \mathbf{a})$ and $p(\boldsymbol{\epsilon}, \mathbf{a}|\mathbf{y}; \theta)$.

When $q(\boldsymbol{\epsilon}, \mathbf{a}) = p(\boldsymbol{\epsilon}, \mathbf{a}|\mathbf{y}; \theta)$, this bound is maximized and $L(q(\boldsymbol{\epsilon}, \mathbf{a}), \theta) = \log p(\boldsymbol{y}; \theta)$. Because the exact posterior $p(\boldsymbol{\epsilon}, \mathbf{a}|\mathbf{y}; \theta) = \frac{p(\boldsymbol{\epsilon}, \mathbf{a}, \mathbf{y}; \theta)}{p(\mathbf{y}; \theta)}$ cannot be found

we are forced to find an approximation of it. The mean field approximation is a commonly used approach to maximize the variational bound w.r.t. $q(\boldsymbol{\epsilon}, \mathbf{a})$ and θ [4]. According to this approach the hidden variables are assumed to be independent, i.e. $q(\boldsymbol{\epsilon}, \mathbf{a}) = q(\boldsymbol{\epsilon})q(\mathbf{a})$. Thus, next we derive the variational algorithm that maximizes the bound $L(q(\boldsymbol{\epsilon})q(\mathbf{a}), \theta)$, aiming at maximizing approximately the logarithm of the likelihood.

Unconstrained maximization of the bound $L(q(\boldsymbol{\epsilon})q(\mathbf{a}), \theta)$ is suboptimal for this formulation. Thus, we resort the modified *constrained variational* approximation, as explained above. In short, the goal is to combine all the information given by the K observation equations of the new model in (5). According to this approach, each $q(\boldsymbol{\epsilon}_l)$ is constrained to have the form:

$$q(\boldsymbol{\epsilon}_l) = N(\mathbf{Q}_l\mathbf{m}, \mathbf{Q}_l^T\mathbf{R}\mathbf{Q}_l), \tag{8}$$

where $\mathbf{m}$ is a $N_H \times 1$ vector, is taken as mean of the high-resolution image, and $\mathbf{R}$ the $N_H \times N_H$ its covariance matrix. Using this model the parameters $\mathbf{m}$ and $\mathbf{R}$ are learned instead of $q(\boldsymbol{\epsilon}_l)$ in the framework of the proposed constrained variational methodology.

In the VE-step, the maximization of $L(q(\boldsymbol{\epsilon}), q(\mathbf{a}), \theta)$ is performed with respect to $q(\mathbf{a})$, $\mathbf{m}$ and $\mathbf{R}$ keeping θ fixed, while in the VM-step, the maximization of the same quantity is performed with respect to θ keeping $q(\mathbf{a})$, $\mathbf{m}$, and $\mathbf{R}$ fixed. At the j-th iteration of the variational algorithm we have:

VE-step:

$$[\mathbf{m}^j, \mathbf{R}^j, q^j(\mathbf{a})] = \arg \max_{\mathbf{m},\mathbf{R},q(\mathbf{a})} L(q(\boldsymbol{\epsilon}; \mathbf{m}, \mathbf{R}), q(\mathbf{a}), \theta^{j-1}) \tag{9}$$

VM-step:

$$\theta^j = \operatorname*{argmax}_{\theta} L(q^j(\boldsymbol{\epsilon}; \mathbf{m}, \mathbf{R}), q^j(\mathbf{a}), \theta) \tag{10}$$

The updates for the VE-Step are:

$$q^j(\boldsymbol{\epsilon}_l; \mathbf{m}, \mathbf{R}) = N(\mathbf{Q}_l\mathbf{m}^j, \mathbf{Q}_l\mathbf{R}^j\mathbf{Q}_l^T), \tag{11}$$

$$\mathbf{m}^j = \beta\mathbf{R}^j\mathbf{B}^T\mathbf{y},\ \mathbf{R}^j = \left(\beta\mathbf{B}^T\mathbf{B} + \frac{1}{K}\sum_{l=1}^{K}\lambda_l^{j-1}\mathbf{Q}_l^T\mathbf{A}_l^{j-1}\mathbf{Q}_l\right)^{-1}. \tag{12}$$

From the above equations it is clear that $\mathbf{m}$ merges information from all filters $\mathbf{Q}_l$ to produce the estimate of $\mathbf{m}$ which is used as the estimate of $\mathbf{x}$.

Finally, the approximate posterior of $\mathbf{a}$ in the VE-step is given by

$$q^j(\mathbf{a}) = \prod_{l=1}^{K}\prod_{i=1}^{N_H} q^j(a_l(i)) \ ,$$

$$q^j(a_l(i)) = Gamma\left(a_l(i); \frac{\nu_l^{j-1}}{2} + \frac{1}{2}, \frac{\nu_l^{j-1}}{2} + \frac{\lambda_l^{j-1}}{2}u^j(i)\right) \ , \tag{13}$$

$$u^j(i) = (\mathbf{m}_l^j(i) + \mathbf{C}_l^j(i,i)), \mathbf{m}_l^j = \mathbf{Q}_l \mathbf{m}^j,\ \mathbf{C}_l^j = \mathbf{Q}_l \mathbf{R}^j \mathbf{Q}_l^T\ ,$$

for $l = 1, 2, \ldots, K$, $i = 1, 2, \ldots, N_H$. Also, $\mathbf{m}_l^j(i)$ is the i-th element of $\mathbf{m}_l^j$ and $\mathbf{C}_l^j(i,i)$ is the i-th diagonal element of $\mathbf{C}_l^j$.

In the VM-step, the bound is maximized w.r.t the parameters. For λ_l we have that the update formula is

$$\lambda_l^j = \frac{N}{\sum_{i=1}^{N} < a_l(i) >_{q^j(\mathbf{a})}\ u^j(i)}\ . \tag{14}$$

Similarly, for ν_l, $l = 1, 2, \ldots, K$, we have that ν_l^j is taken as the root of the function ϕ:

$$\begin{aligned}\phi(\nu_l) = \frac{1}{N_H}\sum_{i=1}^{N_H} \log < a_l(i) >_{q^j(\mathbf{a})} - \frac{1}{N}\sum_{i=1}^{N_H} < a_l(i) >_{q^j(\mathbf{a})} + \psi\left(\frac{\nu_l^{j-1}}{2} + \frac{1}{2}\right) \\ - \log\left(\frac{\nu_l^{j-1}}{2} + 2\right) - \psi\left(\frac{\nu_l}{2}\right) + \log\frac{\nu_l}{2} + 1\ ,\end{aligned} \tag{15}$$

where ψ is the digamma function. We find $\phi(\nu_l^j) = 0$ numerically using the bisection method.

5 Computational Implementation

In this section we describe some of the implementation details of the variational algorithm derived in Sect. 4, given by (12), (14) and (15).

5.1 GPGPU Linear Solver

The most computationally intensive operation of our algorithm is the multiplication of the matrix $\mathbf{R}^{-1}$ (its inverse is given by (12)), which is the matrix of the linear system we aim to solve, with a vector $\mathbf{p}$. To parallelize these operations, we take advantage of the structure of $\mathbf{R}^{-1}$, which is composed by products and sums of circulant and diagonal matrices and implement them efficiently on the GPU using CUDA.

The multiplication of the diagonal matrix $\mathbf{A}_l$ with a vector, can be viewed as an element wise multiplication of two vectors and is parallelized very easily. For the implementation of this operation, each thread running on the GPU performs a multiplication of two elements.

The products of circulant matrices $\mathbf{H}$ and $\mathbf{Q}_l$ with vectors is similarly straight forward to parallelize, though it is slightly more complicated. We first note that a circulant matrix can be diagonalized in the DFT domain. We use this diagonal form and perform the operation with circulant matrices by alternating between the spatial and frequency domain.

The computation of the diagonal elements $\mathbf{C}_l^j(i,i)$ in (13) is also a very challenging computational task since the matrix $\mathbf{R}$ is of size $N_H \times N_H$ with

$N_H = 65,536$ for 256×256 high-resolution images. In this work a random sampling Monte-Carlo method is used for this computation. Due to space constraints we will not provide the details. However, generation of each sample requires the solution of the linear system $\mathbf{Rd} = \mathbf{e}$ where $\mathbf{e}$ is a random vector. The parallelization described above is used for the solution of this system. However, this computation is very expensive computationally and slows down significantly the proposed algorithm. Thus, we also propose an algorithm which avoids this computation.

5.2 Initialization

Initially, the image $\mathbf{m}$ and the noise variance β are set equal to the estimates obtained from the application of the super-resolution algorithm in [9], where a stationary simultaneously auto-regressive image prior is used. This algorithm is very efficient because it can be implemented entirely in the DFT domain.

The registration parameters for the proposed algorithm are computed by the BFGS [7] optimization by solving the following minimization problem:

$$\zeta_k^* = \arg \min_{\zeta_k} |||\mathbf{W}_{N_L}(\zeta_k)\mathbf{y}_k - \mathbf{y}_1||_2^2, \ k = 1, \ldots, P \ , \tag{16}$$

where $\mathbf{W}_{N_L}(\zeta_k)$ is the low-resolution counterpart of $\mathbf{W}(\theta_k)$ of size $N_L \times N_L$. Finite differences are used to compute the gradient required by this approach.

The overall algorithm proposed can be summarized as:

1. Find the registration parameters using (16).
2. Find the initial high-resolution image $\mathbf{m}^0$ and the noise variance β^0 by using the algorithm in [9]
3. Repeat the computations in (12), (14) and (15) until convergenece.

6 Numerical Experiments

In order to test the proposed methodology, we used both artificially generated and real data. We compared the proposed algorithm with two previous state-of-the-art algorithms. The total variation (TV) regularization proposed in [13] and the non-stationary prior proposed in [5]. Both of these works deal with the super-resolution problem, however, they assume an imaging model different from (2). For a fair comparison we modify these algorithms to include the same registration model and in addition we initialize then with the same procedure described in Sect. 5. The non-stationary prior with the Maximum a Posterior (MAP) proposed in [5] and the Majorization-Minimization (MM) in [13] are abbreviate as MMTV and NSMAP, respectively. Furthermore, we use the same GPU linear solver described in Sect. 5 in order to solve the linear systems used by these algorithms to reconstruct the high-resolution image. This is straightforward because the matrices of the linear systems are of the same form with $\mathbf{R}^{-1}$. The model parameters for both algorithms were found by trial and error

experiments, contrary to the proposed algorithm where the model parameters are found automatically.

In order to conduct experiments where the ground truth is known, we used synthetic data. One set of eight 128×128 low-resolution images were generated using the well-known "Cameraman" image of size 256×256 according to the imaging model given by (1), with decimation factor $d = 2$. One type of blur and three types of noise were used (resulting in three image sets): uniform point spread function (PSF) of size 5×5 and AWG noise corresponding to signal to noise ratio (SNR) $SNR = 40, 30$, and $20dB$. This metric and the MSE metric between the restored image and the original that was used to evaluate the performance of the algorithm, are defined as

$$SNR = 10 \log_{10} \frac{\|\mathbf{z}_i\|_2^2}{N_H \sigma^2} dB, \; MSE = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}{N_H} \; ,$$

where σ^2 is the variance of the additive noise and N_H is the size of the zero mean image $\mathbf{z}_i$ and $\mathbf{x}$ and $\hat{\mathbf{x}}$ are the original and estimated images, respectively.

In Fig. 1a the low-resolution image of the experiment with uniform blur 5×5 and $SNR = 20$ is shown. In Fig. 1 we show the super-resolved images (b) with MMTV, (c) with NSMAP, (d) the herein proposed variational algorithm *without* the diagonal elements $\mathbf{C}_l^j(i,i)$ labeled as *ALG1* and (e) the herein proposed variational algorithm *with* the diagonal elements $\mathbf{C}_l^j(i,i)$ labeled as *ALG2*.

In Table 1 we provide the MSE results for these three experiments. From these results it is clear that the proposed algorithms provides superior results as compared to MMTV and NSMAP.

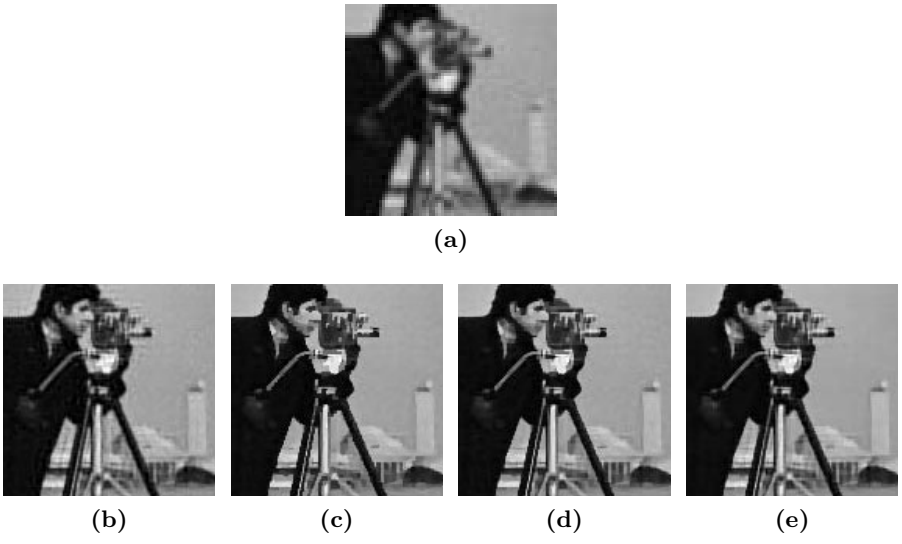
Table 1. MSE's for the experiments using synthetic data

Method	$SNR = 40$	$SNR = 30$	$SNR = 20$
[13]	85	92	141
[5]	78	82	113
ALG1	63	72	**95**
ALG2	**62**	**71**	97

We also used the proposed super-resolution algorithm on a real data set that includes four low-resolution degraded images that contain both translations and rotations. One of them is shown in Fig. 2a. Each low-resolution image is of size 256×256. The $2\times$ super-resolved images obtained by (b) MMTV, (c) NSMAP and (d)-(e) the herein proposed variational algorithms are shown in Fig. 2.

We tested the proposed algorithms in terms of their speed also. The main body of the algorithms was implemented in MATLAB (R2009a). The graphics chip used is NVIDA's GTX 285, which contains 240 CUDA cores, 1GB RAM and 1.47 GHz core clock frequency. For 256×256 image, the proposed algorithm *ALG2* with use of the GPU takes about 60' while the other three NSMAP, MMTV

and *ALG1* take about 1'. This large difference is due to the computation of the diagonal elements $\mathbf{C}_l^j(i,i)$. The acheived speed up by running these algorithm on the GPU was 8-10x as compared to using a CPU (Intel Core i7, 2.47Mhz).



(a) **(b)** **(c)** **(d)** **(e)**

Fig. 1. (a) Low-resolution observation and 2× super-resolved images using: (b) MMTV, (c) NSMAP, (d) and (e) and the herein proposed variational algorithms *ALG1* and *ALG2*

(a) **(b)** **(c)** **(d)** **(e)**

Fig. 2. (a) Low-resolution observation and 2× super-resolved images using: (b) MMTV, (c) NSMAP and the proposed (d) *ALG1* and (e) *ALG2*

7 Conclusions and Future work

We proposed a variational inference super-resolution algorithm where all the model variables and parameters are estimated automatically. We demonstrated numerical experiments that showed the superiority of the proposed methodology. Precisely, the resolution of the super-resolved images shown in Fig. 1 and 2

has greatly improved. Furthermore, the super-resolved images with the proposed algorithm have better edge structure and are visually more pleasant. Also, the MSE in Table 1 with the experiments on the artificial data the proposed algorithm to be superior to all other tested algorithms.

The GPU implementation of the linear solver achieved up to a 10x speed-up as compared to the CPU. This allowed us to estimate the diagonal elements of the inverse of matrix, which gave significant better results in terms of MSE, compared to other two state-of-the-art algorithms.

References

1. Park, S., Park, M., Kang, M.: Super-resolution Image Reconstruction: a Technical Overview. IEEE Signal Processing Magazine 20(3), 21–36 (2003)
2. Tom, B., Galatsanos, N., Katsaggelos, A.: Reconstruction of a high resolution image from multiple low resolution images. In: Chaudhuri, S. (ed.) Super-Resolution Imaging, ch. 4, pp. 71–105. Kluwer, Dordrecht (2001)
3. Katsaggelos, A.K., Molina, R., Mateos, J.: Super resolution of images and video. Synthesis Lectures on Image, Video, and Multimedia Processing, Morgan and Claypool (2007)
4. Tzikas, D., Likas, A., Galatsanos, N.: Life After the EM Algorithm: The Variational Approximation for Bayesian Inference. IEEE Signal Processing Magazine 25(6), 131–146 (2008)
5. Chantas, G., Galatsanos, N., Woods, N.: Super Resolution Based on Fast Registration and Maximum A Posteriori Reconstruction. IEEE Transactions on Image Processing 16(7), 1821–1830 (2007)
6. Genest, C., Zidek, J.V.: Combining probability distributions: A critique and an annotated bibliography. Statistical Science (1986)
7. Nash, S.G., Sofer, A.: Linear and Nonlinear Programming. McGraw Hill, New York (1996)
8. Chantas, G., Galatsanos, N.P., Likas, A., Saunders, M.: Variational Bayesian image restoration based on a product of t-distributions image prior. Transactions on Image Processing 17(10), 1795–1805 (2008)
9. Woods, N.A., Galatsanos, N.P., Katsaggeloss, A.K.: Stochastic methods for joint restoration, interpolation and registration of multiple under sampled images. Transactions on Image Processing 15(1), 201–213 (2006)
10. Keane, A.: CUDA (compute unified device architecture) (2006), `http://developer.nvidia.com/object/cuda.html`
11. Buatois, L., Caumon, G., Levy, B.: Concurrent Number Cruncher: An Efficient Sparse Linear Solver on the GPU. In: Perrott, R., Chapman, B.M., Subhlok, J., de Mello, R.F., Yang, L.T. (eds.) HPCC 2007. LNCS, vol. 4782, pp. 358–371. Springer, Heidelberg (2007)
12. Bolz, J., Farmer, I., Grinspun, E., Schröder, P.: Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid. ACM Transactions on Graphics 22(3), 917–924 (2003)
13. Galatsanos, N.: A Majorization-Minimization Approach to Total Variation Image Reconstruction of Super-Resolved Images. In: EUSIPCO 2008, Lausanne, Switzerland (2008)