A Comparative Study of MAP Estimation Methods in Image Denoising Problem based on Gaussian Markov Random Fields

Shun Kataoka

In this paper, we report the results of numerical comparisons of the MAP estimation methods for the image denoising problem based on Gaussian Markov random fields. When using Gaussian Markov random fields, the image denoising using can be reduced to the problem of solving simultaneous linear equations. There are various methods for solving simultaneous linear equations, but the computation time strongly depends on the structure of the coefficient matrix, and it is not clear which method is effective. Through the numerical experiments using standard images, we verify which solving method is effective in the image denoising problem based on Gaussian Markov random fields.

1 Introduction

Bayesian image processing based on Markov random fields (MRFs) is an effective framework that allows us to deal with the uncertainty in the image processing problems [1, 2, 3]. Among the various types of MRFs, the method using Gaussian Markov random fields (GMRFs) is a simple but easy-to-use image processing method [4, 5]. In this framework, the posterior distribution is expressed as a Gaussian distribution, and by adopting the MAP estimation, the image processing problem is reduced to the problem of finding the mean vector of the Gaussian distribution from the observed image. In a typical image processing problem such as image denoising, this problem of finding this mean vector is formulated as a problem of solving simultaneous linear equations.

In this paper, we quantitatively compare the computation time of various method for solving simultaneous linear equations using the image denoising problem as an example. In the typical denoising method using GMRFs, it is necessary to handle simultaneous linear equations with a large sparse coefficient matrix. For simultaneous linear equations with a large sparse coefficient matrix, there are various methods for finding solutions in $O(N)$, where N is the number of rows in the coefficient matrix. The actual computation time of these solving methods strongly depends on the structure of the coefficient matrix, and it is not clear which method is effective for the image denoising problem based on GMRFs. Therefore, in this paper, we conduct a comparative study of the computation time for the image denoising problem using the basic solving methods for simultaneous linear equations

with a large sparse coefficient matrix.

The remainder of this paper is organized as follows. In section 2, we define the posterior distribution for image denoising problems and derive a denoising method based on GMRFs and a Bayesian perspective. In section 3, we briefly explain basic methods for solving simultaneous linear equations with large sparse coefficient matrices. In section 4, we provides numerical comparative verification of the computation time for various solving methods described in section 3. Finally, in section, we present our concluding remarks.

2 Image denoising method based on GMRFs

We assume that the image consists of $H \times W$ pixels arranged on the square grid. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ be a $N(= HW)$ dimensional real random vector where the element x_i takes continuous values in $(-\infty, +\infty)$ and corresponds to the intensity of the i-th pixel (pixels are arranged in the raster scanning order on the image). That is, we assume that the image is a realization $\boldsymbol{x} = \mathbf{x}$ of the random vector $\boldsymbol{x}$. In this paper, we consider the image denoising problem. Under the assumption of the additive white Gaussian noise (AWGN), the goal of this problem is to estimate the original uncorrupted image $\mathbf{x}^* = (\mathrm{x}_1^*, \mathrm{x}_2^*, \ldots, \mathrm{x}_N^*)$ from the noise-corrupted image $\mathbf{y} = (\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_N)$ expressed as

$$\mathrm{y}_i = \mathrm{x}_i^* + \mathrm{n}_i, \tag{1}$$

where n_i is a Gaussian noise with zero mean and variance σ^2. In the Bayesian frame work, the original image $\boldsymbol{x} = \mathbf{x}^*$ can be estimated using the posterior distribution $P_{\mathrm{post}}(\boldsymbol{x} \,|\, \boldsymbol{y} = \mathbf{y})$ expressed as

$$P_{\mathrm{post}}(\boldsymbol{x} \,|\, \boldsymbol{y} = \mathbf{y}) = \frac{P_{\mathrm{like}}(\boldsymbol{y} = \mathbf{y} \,|\, \boldsymbol{x}) P_{\mathrm{pri}}(\boldsymbol{x})}{\int_{\mathbb{R}^N} P_{\mathrm{like}}(\boldsymbol{y} = \mathbf{y} \,|\, \boldsymbol{x}) P_{\mathrm{pri}}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}}, \tag{2}$$

where $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$ is a N dimensional random vector corresponding to the intensities of the noise-corrupted image, $P_{\mathrm{pri}}(\boldsymbol{x})$ is a prior distribution, and $P_{\mathrm{like}}(\boldsymbol{y} = \mathbf{y} \,|\, \boldsymbol{x})$ is a likelihood function.

We define the prior distribution of the original image as

$$P_{\mathrm{pri}}(\boldsymbol{x}) = \frac{1}{Z_{\mathrm{pri}}} \exp(-E_{\mathrm{pri}}(\boldsymbol{x})), \tag{3}$$

with the energy function

$$E_{\mathrm{pri}}(\boldsymbol{x}) = -\frac{h}{2} \sum_{i \in V} (x_i - \mu)^2 - \frac{J}{2} \sum_{\{i,j\} \in E} (x_i - x_j)^2 \tag{4}$$

where $V = \{1, 2, \ldots, N\}$ and $E = \{\{i, j\} \,|\, i \text{ and } j \text{ are connected}\}$ are the sets of vertices and edges in the square grid, respectively. h, J, and μ are parameters in the prior distribution. If $h > 0$ is set to a large value, x_i tends to take close value to μ. And if $J > 0$ is set to a large value, neighboring x_i and x_j tend to take close values. Z_{pri} is a normalization constant of the prior distribution defined as

$$Z_{\mathrm{pri}} = \int_{\mathbb{R}^N} \exp(-E_{\mathrm{pri}}(\boldsymbol{x})) \,\mathrm{d}\boldsymbol{x}.$$

Since we assume the AWGN in the generation process of the noise-corrupted image, the likelihood function in equation (2) can be expressed as

$$P_{\text{like}}(\boldsymbol{y} = \mathbf{y} \,|\, \boldsymbol{x}) = \prod_{i \in V} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(\mathrm{y}_i - x_i)^2\right), \tag{5}$$

where $\sigma > 0$ is the standard deviation of the AWGN.

By substituting equations (3), (4), and (5) into equation (2), the posterior distribution is expressed as

$$P_{\text{post}}(\boldsymbol{x} \,|\, \boldsymbol{y} = \mathbf{y}) = \frac{1}{Z_{\text{post}}} \exp(-E_{\text{post}}(\boldsymbol{x} \,|\, \boldsymbol{y} = \mathbf{y})), \tag{6}$$

where

$$\begin{aligned} E_{\text{post}}(\boldsymbol{x} \,|\, \boldsymbol{y} = \mathbf{y}) = &-\frac{1}{2\sigma^2} \sum_{i \in V} (\mathrm{y}_i - x_i)^2 - \frac{h}{2} \sum_{i \in V} (x_i - \mu)^2 \\ &- \frac{J}{2} \sum_{\{i,j\} \in E} (x_i - x_j)^2 \end{aligned} \tag{7}$$

and

$$Z_{\text{post}} = \int_{\mathbb{R}^N} \exp(-E_{\text{post}}(\boldsymbol{x} \,|\, \boldsymbol{y} = \mathbf{y}))\, \mathrm{d}\boldsymbol{x}, \tag{8}$$

respectively.

Since the energy function of the posterior distribution in equation (7) is

a quadratic form, we can rewrite this function as

$$E_{\mathrm{post}}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\top}\Lambda\boldsymbol{x} - \boldsymbol{b}^{\top}\boldsymbol{x} + \mathrm{Const.}, \tag{9}$$

where $\boldsymbol{b}$ is the N dimensional vector whose i-th element is defined as

$$b_i = h\mu + \frac{\mathrm{y}_i}{\sigma^2} \tag{10}$$

and Λ is the $N \times N$ matrix whose (i,j)-th element is defined as

$$\lambda_{ij} = \begin{cases} h + J|\partial i| & (i = j) \\ -J & (\{i,j\} \in E) \\ 0 & (\{i,j\} \notin E) \end{cases}, \tag{11}$$

respectively. $\partial i = \{k \in V \mid \{i,k\} \in E\}$ is a set of all the neighboring vertices of vertex i. Using equation (9) and the assumption that x_i takes values in $(-\infty, +\infty)$, the posterior distribution in equation (6) can be expressed as the following multidimensional Gaussian distribution:

$$P_{\mathrm{post}}(\boldsymbol{x}) = \sqrt{\frac{\det\Lambda}{(2\pi)^N}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \Lambda^{-1}\boldsymbol{b})^{\top}\Lambda(\boldsymbol{x} - \Lambda^{-1}\boldsymbol{b})\right). \tag{12}$$

The mean vector and precision matrix (the inverse of the covariance matrix) of this Gaussian distribution are given by $\Lambda^{-1}\boldsymbol{b}$ and Λ, respectively. A Gaussian distribution defined based on a graph structure is called a Gaussian Markov random fields (GMRFs). Therefore, the posterior distribution in

equation (12) can be regarded as a GMRFs defined on a square grid.

The image denoising is achieved by finding the mode vector $\widehat{\mathbf{x}}$ that maximizes the posterior distribution in equation (12):

$$\widehat{\mathbf{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}} P_{\text{post}}(\boldsymbol{x} \mid \boldsymbol{y} = \mathbf{y}). \tag{13}$$

The estimation that use the realization that maximizes the posterior distribution as the estimated value is called MAP (maximum a posteriori) estimation. This maximization of the posterior distribution can also be regarded as the minimization of the energy function in equation (9). Since the posterior distribution is a Gaussian distribution, the mode vector $\widehat{\mathbf{x}}$ is equivalent to the mean vector of the posterior distribution:

$$\widehat{\mathbf{x}} = \Lambda^{-1}\boldsymbol{b}. \tag{14}$$

Therefore, the image denoising result $\widehat{\mathbf{x}}$ is given by the solution of the simultaneous linear equations

$$\Lambda\boldsymbol{m} = \boldsymbol{b}, \tag{15}$$

where $\boldsymbol{m}$ is a N dimensional unknown vector.

3 Solving simultaneous linear equations

3.1 Gauss-Seidel method

The Gauss-Seidel (GS) method is an iterative technique for solving simultaneous linear equations [6]. For the simultaneous equation in equation (15), the iterative procedure of this method is given by

$$m_i^{(t+1)} = \frac{1}{\lambda_{ii}} \left(b_i - \sum_{k=1}^{i-1} \lambda_{ik} m_k^{(t+1)} - \sum_{k=i+1}^{N} \lambda_{ik} m_k^{(t)} \right) \tag{16}$$

for $i = 1, 2, \ldots, N$. Since the matrix Λ is a positive define symmetric matrix, this iterative procedure is known to converge. The order of the computation time of the Gauss-Seidel method for equation (15) is $O(N)$, because the number of nonzero off-diagonal elements in the i-th row of matrix Λ is at most four.

3.2 Successive over-relaxation method

The Successive over-relaxation (SOR) method is an improved version of the GS method and aims to accelerate the convergence of the iterative calculations [6]. In the SOR method, the iterative procedure in equation (16) is modified as

$$m_i^{(t+1)} = (1-\omega) m_i^{(t)} + \frac{\omega}{\lambda_{ii}} \left(b_i - \sum_{k=1}^{i-1} \lambda_{ik} m_k^{(t+1)} - \sum_{k=i+1}^{N} \lambda_{ik} m_k^{(t)} \right) \tag{17}$$

for $i = 1, 2, \ldots, N$, where ω is called the relaxation factor. Similar to the GS method, the iterative procedure of the SOR method is guaranteed to converge for $0 < \omega < 2$ when the matrix Λ is positive define symmetric matrix. When $\omega = 1$, the computational procedure of the SOR method is the same as that of the GS method, and when $\omega > 1$, the SOR method is expected to converge faster than the GS method. Since the order of the computation time of the GS method for equation (15) is $O(N)$, the order of the computation time of the SOR method for equation (15) is also $O(N)$.

3.3 Conjugate gradient method

The conjugate gradient (CG) method is an iterative optimization method for solving unconstrained optimization problem [7]. Since the image denoising result $\widehat{\boldsymbol{x}}$ is the point that minimizes the energy function $E_{\mathrm{post}}(\boldsymbol{x} \,|\, \boldsymbol{y} = \mathbf{y})$, the simultaneous equation in equation (15) can also be solved by minimizing $E_{\mathrm{post}}(\boldsymbol{x} \,|\, \boldsymbol{y} = \mathbf{y})$.

The CG method finds the solution to the simultaneous linear equations in equation (15) by iterating the following procedure:

$$\alpha_t = \frac{(\boldsymbol{r}^{(t)}, \boldsymbol{r}^{(t)})}{(\boldsymbol{p}^{(t)}, \Lambda \boldsymbol{p}^{(t)})}, \tag{18}$$

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + \alpha_t \boldsymbol{p}^{(t)}, \tag{19}$$

$$\boldsymbol{r}^{(t+1)} = \boldsymbol{r}^{(t)} - \alpha_t \boldsymbol{p}^{(t)}, \tag{20}$$

$$\beta_t = \frac{(\boldsymbol{r}^{(t+1)}, \boldsymbol{r}^{(t+1)})}{(\boldsymbol{r}^{(t)}, \boldsymbol{r}^{(t)})}, \tag{21}$$

$$\boldsymbol{p}^{(t+1)} = \boldsymbol{r}^{(t+1)} + \beta_t \boldsymbol{p}^{(t)}, \tag{22}$$

where $\boldsymbol{p}^{(t)}$ and $\boldsymbol{r}^{(t)}$ are search direction vector and residual vector at iteration t, respectively, and $(\boldsymbol{a}, \boldsymbol{b})$ represents the inner product of vector $\boldsymbol{a}$ and vector $\boldsymbol{b}$. This procedure stars with a randomly determined initial vector $\boldsymbol{m}^{(0)}$, and the initial values of $\boldsymbol{r}^{(t)}$ and $\boldsymbol{p}^{(t)}$ are set as

$$\boldsymbol{r}^{(0)} = \boldsymbol{b} - \Lambda \boldsymbol{m}^{(0)} \tag{23}$$

and

$$\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)}, \tag{24}$$

respectively. The order of the computation time of the CG method for equation (15) is $O(N)$, because all the right-hand sides of equations (18)-(24) can be calculated in $O(N)$.

3.4 Incomplete Cholesky CG method

The incomplete Cholesky conjugate gradient (ICCG) method is an improved version of the CG method with effective preconditioning [8]. Instead of solving the simultaneous linear equations in equation (15), the ICCG method solves the equivalent simultaneous linear equation

$$(C^{-1}\Lambda(C^{\top})^{-1})C^{\top}\boldsymbol{m} = C^{-1}\boldsymbol{b} \tag{25}$$

by using the CG method. The $N \times N$ matrix C is called preconditioning matrix and is defined as

$$C = LD^{1/2}, \tag{26}$$

where $LDL^\top = CC^\top$ is an incomplete Cholesky decomposition of matrix Λ, the matrix L is a lower triangular matrix, and the matrix D is a diagonal matrix.

The iterative procedure of the ICCG method is given as follows:

$$\alpha_t = \frac{(\boldsymbol{r}^{(t)}, (LDL^\top)^{-1}\boldsymbol{r}^{(t)})}{(\boldsymbol{p}^{(t)}, \Lambda\boldsymbol{p}^{(t)})}, \tag{27}$$

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + \alpha_t \boldsymbol{p}^{(t)}, \tag{28}$$

$$\boldsymbol{r}^{(t+1)} = \boldsymbol{r}^{(t)} - \alpha_t \boldsymbol{p}^{(t)}, \tag{29}$$

$$\beta_t = \frac{(\boldsymbol{r}^{(t+1)}, (LDL^\top)^{-1}\boldsymbol{r}^{(t+1)})}{(\boldsymbol{r}^{(t)}, (LDL^\top)^{-1}\boldsymbol{r}^{(t)})}, \tag{30}$$

$$\boldsymbol{p}^{(t+1)} = (LDL^\top)^{-1}\boldsymbol{r}^{(t+1)} + \beta_t \boldsymbol{p}^{(t)}, \tag{31}$$

where the initial values of $\boldsymbol{r}^{(t)}$ and $\boldsymbol{p}^{(t)}$ are set as

$$\boldsymbol{r}^{(0)} = \boldsymbol{b} - \Lambda\boldsymbol{m}^{(0)} \tag{32}$$

and

$$\boldsymbol{p}^{(0)} = (LDL^\top)^{-1}\boldsymbol{r}^{(0)}, \tag{33}$$

respectively, using the randomly determined initial vector $\boldsymbol{m}^{(0)}$. It is worth noting that the calculation of $\boldsymbol{v} = (LDL^\top)^{-1}\boldsymbol{r}$ can be easily accomplished by solving two simultaneous linear equations

$$L\boldsymbol{y} = \boldsymbol{r} \tag{34}$$

and

$$DL^\top \boldsymbol{v} = \boldsymbol{y} \tag{35}$$

in sequence. Similar to CG method, the right-hand sides of equations (27)-(35) and the incomplete Cholesky factorization can all be calculated in $O(N)$, so the computation time of the ICCG method for equation (15) is also $O(N)$.

4 Numerical Experiments

In this section, we perform numerical comparisons of image denoising methods using the various simultaneous equation solving methods described in the previous section. In this experiments, the standard images in figure 1 were used in three different sizes ($N = 128 \times 128, 256 \times 256,$ and 512×512). All the experiments were implemented using Julia and were run single-threaded on an Ubuntu 24.04.1 LTS (64 bit) machine with an AMD Ryzen7 8840U running at 3.3 GHz (maximum boost clock 5.1 GHz) and 16GB RAM. The parameter h was set to 1.0×10^{-10}, and the parameter μ was set to the average intensity of the noise-corrupted image $\mathbf{y}$. The value of parameter J

Figure 1: Gray scale standard images used in the experiments.

Table 1: The value of parameter J used in the experiments

	image size		
image name	$N = 128 \times 128$	$N = 256 \times 256$	$N = 512 \times 512$
cameraman	0.70×10^{-3}	1.38×10^{-3}	2.52×10^{-3}
house	1.22×10^{-3}	2.52×10^{-3}	4.94×10^{-3}
airplane	0.70×10^{-3}	1.44×10^{-3}	2.42×10^{-3}
lake	0.58×10^{-3}	1.18×10^{-3}	1.86×10^{-3}
mandrill	0.58×10^{-3}	0.98×10^{-3}	1.40×10^{-3}
peppers	1.00×10^{-3}	2.02×10^{-3}	3.14×10^{-3}
male	0.94×10^{-3}	1.64×10^{-3}	2.26×10^{-3}
bridge	0.56×10^{-3}	1.00×10^{-3}	1.22×10^{-3}

was set according to the type and size of the image based on the results of preliminary experiments, as shown in table 1, and parameter σ^2 was set to the actual variance value of the AWGN used to generate to noise-corrupted images.

First, we investigated the performance of the SOR method with respect to the relaxation factor ω. Figures 2-4 show the average computation time

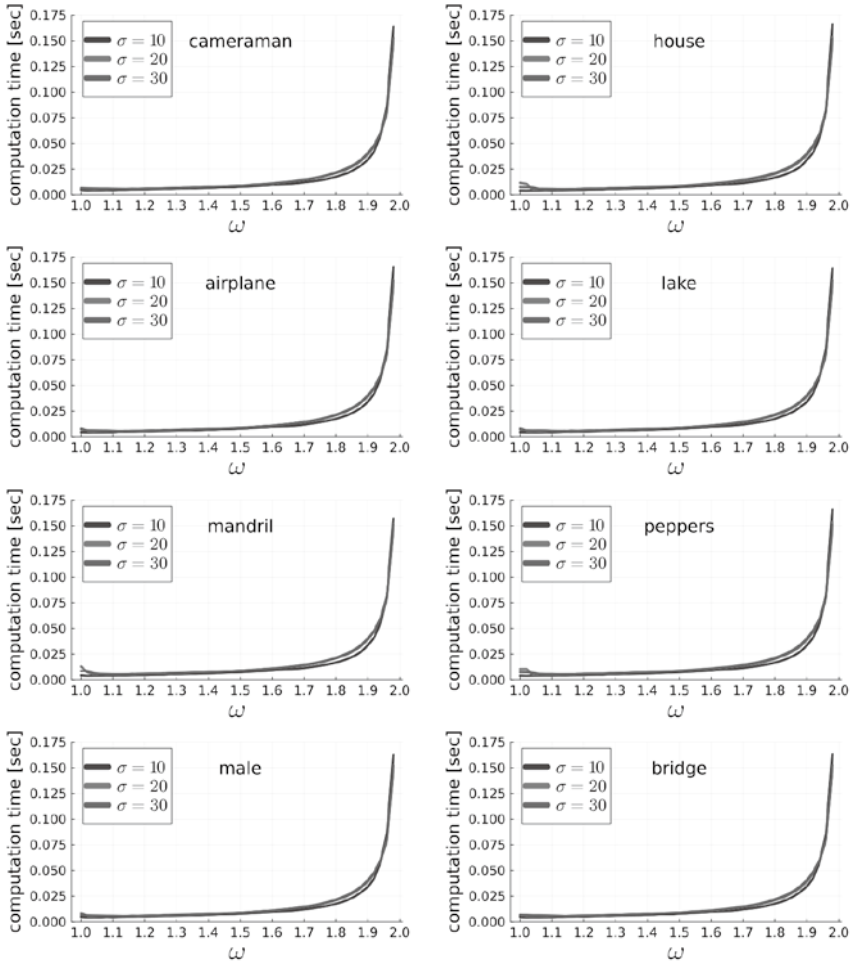

Figure 2: Average computation time over 500 trials versus the relaxation factor ω for $N = 128 \times 128$ images.

over 500 trials versus the relaxation factor ω for each image ($N = 128 \times 128, 256 \times 256,$ and 512×512) in figure 1 and different noise level. The noise-corrupted images were generated by adding AWGN of noise level σ to the

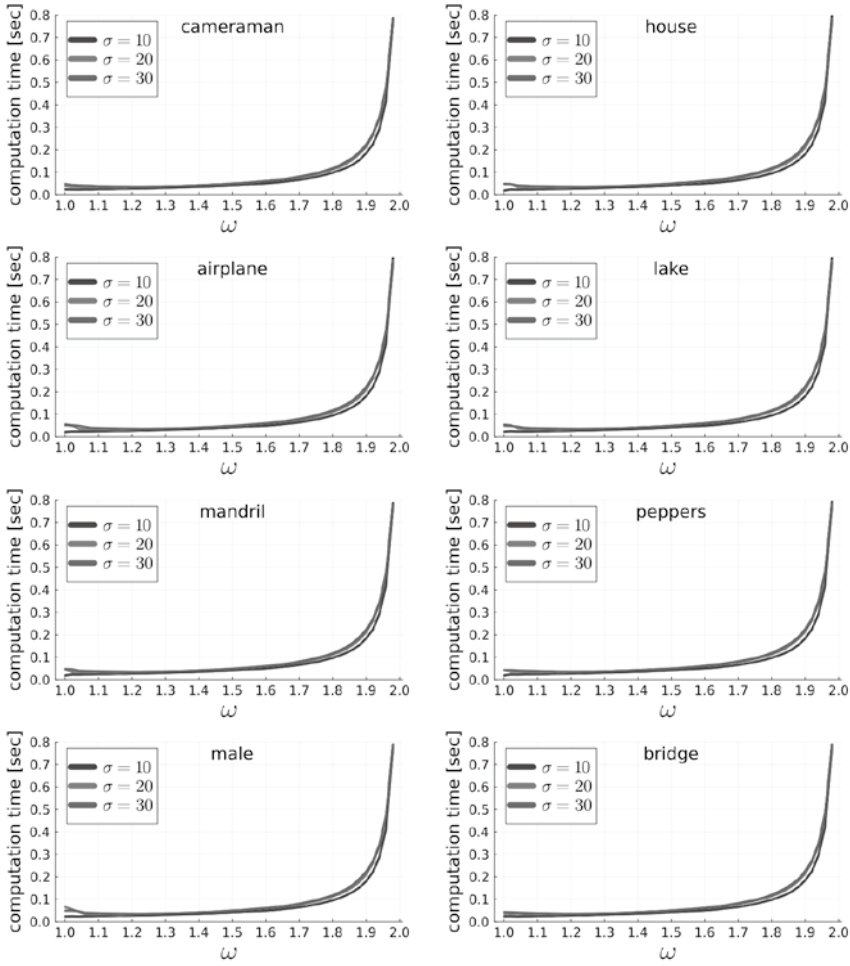

Figure 3: Average computation time over 500 trials versus the relaxation factor ω for $N = 256 \times 256$ images.

original noise-less images for each trial. In these figures, the setting $\omega = 1$ corresponds to the GS method. For any image size, when the relaxation factor ω is small, the computation time was either the same as or slightly

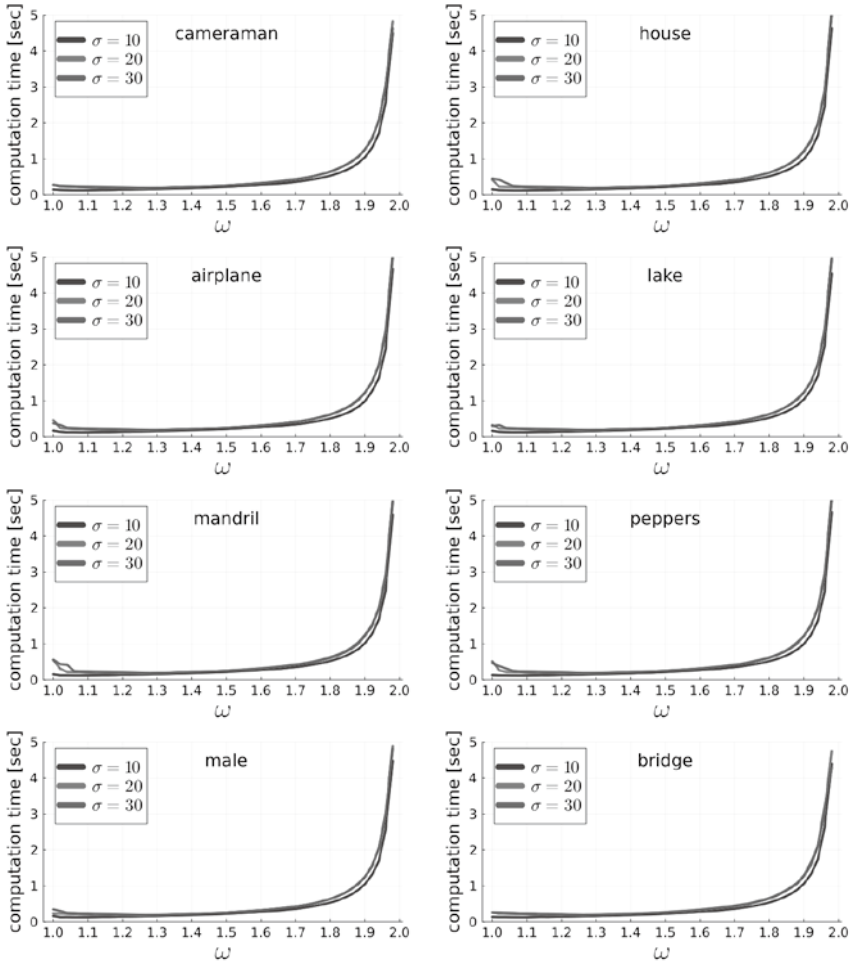

Figure 4: Average computation time over 500 trials versus the relaxation factor ω for $N = 512 \times 512$ images.

improved from the GS method. However, as the value of ω increases, the computation time tends to become significantly larger, and this tendency is the same for all noise levels and image sizes.

Table 2: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 128 \times 128$ and $\sigma = 10$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	$\mathbf{3.43 \times 10^{-3}}$	3.98×10^{-3}	5.93×10^{-3}
house	3.73×10^{-3}	$\mathbf{3.30 \times 10^{-3}}$	4.83×10^{-3}
airplane	$\mathbf{3.63 \times 10^{-3}}$	3.86×10^{-3}	5.68×10^{-3}
lake	$\mathbf{4.94 \times 10^{-3}}$	7.43×10^{-3}	5.07×10^{-3}
mandril	$\mathbf{3.43 \times 10^{-3}}$	3.93×10^{-3}	5.66×10^{-3}
peppers	10.36×10^{-3}	$\mathbf{5.41 \times 10^{-3}}$	7.51×10^{-3}
male	3.59×10^{-3}	$\mathbf{3.92 \times 10^{-3}}$	6.05×10^{-3}
bridge	7.36×10^{-3}	$\mathbf{6.97 \times 10^{-3}}$	7.00×10^{-3}

Table 3: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 128 \times 128$ and $\sigma = 20$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	$\mathbf{4.04 \times 10^{-3}}$	4.05×10^{-3}	6.01×10^{-3}
house	5.09×10^{-3}	$\mathbf{3.78 \times 10^{-3}}$	5.42×10^{-3}
airplane	4.06×10^{-3}	$\mathbf{3.84 \times 10^{-3}}$	5.87×10^{-3}
lake	4.81×10^{-3}	$\mathbf{3.79 \times 10^{-3}}$	5.39×10^{-3}
mandril	$\mathbf{3.89 \times 10^{-3}}$	4.04×10^{-3}	5.72×10^{-3}
peppers	9.23×10^{-3}	$\mathbf{3.80 \times 10^{-3}}$	6.40×10^{-3}
male	4.12×10^{-3}	$\mathbf{4.07 \times 10^{-3}}$	5.88×10^{-3}
bridge	4.63×10^{-3}	$\mathbf{3.85} \times 10^{-3}$	5.44×10^{-3}

Tables 2-4 show the average computation time over 500 trials of the GS, CG, ICCG methods for each image with $N = 128 \times 128$ and noise levels $\sigma = 10, 20,$ and 30. Tables 5-7 show the average computation time for $N = 256$ and tables 8-10 show the average computation time for $N = 512$. In these tables, the values are in seconds and the best results for each image and noise level are shown in bold. When the noise level is small, the GS method and the CG method have shorter computation times than the ICCG method,

Table 4: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 128 \times 128$ and $\sigma = 30$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	5.02×10^{-3}	$\mathbf{4.41 \times 10^{-3}}$	6.79×10^{-3}
house	5.96×10^{-3}	$\mathbf{4.56 \times 10^{-3}}$	6.28×10^{-3}
airplane	4.82×10^{-3}	$\mathbf{4.17 \times 10^{-3}}$	6.61×10^{-3}
lake	5.74×10^{-3}	$\mathbf{4.20 \times 10^{-3}}$	6.22×10^{-3}
mandril	4.68×10^{-3}	$\mathbf{4.39 \times 10^{-3}}$	6.58×10^{-3}
peppers	5.47×10^{-3}	$\mathbf{4.27 \times 10^{-3}}$	5.61×10^{-3}
male	4.83×10^{-3}	$\mathbf{4.25 \times 10^{-3}}$	6.69×10^{-3}
bridge	5.62×10^{-3}	$\mathbf{4.23 \times 10^{-3}}$	5.69×10^{-3}

Table 5: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 256 \times 256$ and $\sigma = 10$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	$\mathbf{15.78 \times 10^{-3}}$	17.22×10^{-3}	24.61×10^{-3}
house	18.10×10^{-3}	$\mathbf{16.81 \times 10^{-3}}$	22.42×10^{-3}
airplane	$\mathbf{15.10 \times 10^{-3}}$	16.77×10^{-3}	23.87×10^{-3}
lake	17.26×10^{-3}	$\mathbf{15.69 \times 10^{-3}}$	22.40×10^{-3}
mandril	$\mathbf{14.74 \times 10^{-3}}$	17.28×10^{-3}	24.25×10^{-3}
peppers	$\mathbf{13.47 \times 10^{-3}}$	16.24×10^{-3}	22.18×10^{-3}
male	$\mathbf{15.35 \times 10^{-3}}$	16.84×10^{-3}	24.51×10^{-3}
bridge	17.24×10^{-3}	$\mathbf{15.77 \times 10^{-3}}$	22.65×10^{-3}

and when $N = 512 \times 512$, the GS method has the shortest computation times for all images. On the other hand, when the noise level is large, the CG method and ICCG method have shorter computation times than the GS method. In particular, when $N = 128 \times 128$ and 256×256, the CG method has the shortest computation times for all images.

Table 6: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 256 \times 256$ and $\sigma = 20$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	23.43×10^{-3}	$\mathbf{21.25 \times 10^{-3}}$	25.38×10^{-3}
house	25.92×10^{-3}	$\mathbf{21.16 \times 10^{-3}}$	25.37×10^{-3}
airplane	23.48×10^{-3}	$\mathbf{21.06 \times 10^{-3}}$	25.53×10^{-3}
lake	25.18×10^{-3}	$\mathbf{21.02 \times 10^{-3}}$	25.40×10^{-3}
mandril	22.75×10^{-3}	$\mathbf{21.23 \times 10^{-3}}$	25.38×10^{-3}
peppers	24.63×10^{-3}	$\mathbf{21.10 \times 10^{-3}}$	25.18×10^{-3}
male	22.47×10^{-3}	$\mathbf{20.88 \times 10^{-3}}$	26.17×10^{-3}
bridge	25.22×10^{-3}	$\mathbf{20.99 \times 10^{-3}}$	25.24×10^{-3}

Table 7: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 256 \times 256$ and $\sigma = 30$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	29.15×10^{-3}	$\mathbf{24.22 \times 10^{-3}}$	29.12×10^{-3}
house	32.09×10^{-3}	$\mathbf{24.84 \times 10^{-3}}$	30.81×10^{-3}
airplane	29.32×10^{-3}	$\mathbf{23.41 \times 10^{-3}}$	28.97×10^{-3}
lake	29.44×10^{-3}	$\mathbf{23.13 \times 10^{-3}}$	29.79×10^{-3}
mandril	27.78×10^{-3}	$\mathbf{23.54 \times 10^{-3}}$	28.96×10^{-3}
peppers	29.68×10^{-3}	$\mathbf{23.76 \times 10^{-3}}$	27.98×10^{-3}
male	28.44×10^{-3}	$\mathbf{23.19 \times 10^{-3}}$	31.25×10^{-3}
bridge	30.36×10^{-3}	$\mathbf{23.22 \times 10^{-3}}$	28.76×10^{-3}

5 Concluding Remarks

In this paper, we performed a comparative examination of various image denoising methods using GMRFs. There are various methods for solving simultaneous linear equations with large sparse coefficient matrices, and different image denoising algorithms can be derived even when using the same GRMFs depending on which solving method is used. In a comparison of

Table 8: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 512 \times 512$ and $\sigma = 10$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	$\mathbf{78.53 \times 10^{-3}}$	92.65×10^{-3}	114.68×10^{-3}
house	$\mathbf{85.55 \times 10^{-3}}$	94.26×10^{-3}	113.33×10^{-3}
airplane	$\mathbf{78.20 \times 10^{-3}}$	93.56×10^{-3}	114.91×10^{-3}
lake	$\mathbf{82.58 \times 10^{-3}}$	92.23×10^{-3}	110.88×10^{-3}
mandril	$\mathbf{79.38 \times 10^{-3}}$	92.86×10^{-3}	114.52×10^{-3}
peppers	$\mathbf{81.99 \times 10^{-3}}$	92.18×10^{-3}	110.75×10^{-3}
male	$\mathbf{78.21 \times 10^{-3}}$	92.54×10^{-3}	113.64×10^{-3}
bridge	$\mathbf{84.84 \times 10^{-3}}$	91.82×10^{-3}	110.43×10^{-3}

Table 9: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 512 \times 512$ and $\sigma = 20$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	140.95×10^{-3}	$\mathbf{131.18 \times 10^{-3}}$	131.18×10^{-3}
house	143.90×10^{-3}	$\mathbf{130.36 \times 10^{-3}}$	132.35×10^{-3}
airplane	139.05×10^{-3}	$\mathbf{129.72 \times 10^{-3}}$	130.29×10^{-3}
lake	141.55×10^{-3}	$\mathbf{129.89 \times 10^{-3}}$	131.27×10^{-3}
mandril	131.15×10^{-3}	131.00×10^{-3}	$\mathbf{129.60 \times 10^{-3}}$
peppers	137.82×10^{-3}	130.97×10^{-3}	$\mathbf{130.62 \times 10^{-3}}$
male	136.44×10^{-3}	131.19×10^{-3}	$\mathbf{130.84 \times 10^{-3}}$
bridge	142.40×10^{-3}	130.52×10^{-3}	$\mathbf{128.82 \times 10^{-3}}$

the GS method and the SOR method, it was confirmed that the computation time was not significantly improved by introducing the relaxation parameters, and that the computation time became larger depending on the parameter settings. In our experiments comparing the GS method, the CG method, and the ICCG method, it was confirmed that when the noise level was small, MAP estimation using the GS method was faster, and conversely, when the noise level was large, the CG method and the ICCG method were

Table 10: Average computation time over 500 trials of the GS, CG, and ICCG method for $N = 512 \times 512$ and $\sigma = 30$. The values in this table are in seconds, and the best results for each image are shown in bold.

	GS	CG	ICCG
cameraman	165.29×10^{-3}	$\mathbf{141.66 \times 10^{-3}}$	143.64×10^{-3}
house	159.23×10^{-3}	133.39×10^{-3}	$\mathbf{132.59 \times 10^{-3}}$
airplane	164.43×10^{-3}	$\mathbf{139.95 \times 10^{-3}}$	143.34×10^{-3}
lake	162.30×10^{-3}	140.47×10^{-3}	$\mathbf{134.14 \times 10^{-3}}$
mandril	161.44×10^{-3}	138.87×10^{-3}	$\mathbf{136.84 \times 10^{-3}}$
peppers	158.54×10^{-3}	136.42×10^{-3}	$\mathbf{134.75 \times 10^{-3}}$
male	167.31×10^{-3}	$\mathbf{137.01 \times 10^{-3}}$	137.89×10^{-3}
bridge	158.76×10^{-3}	141.12×10^{-3}	$\mathbf{134.46 \times 10^{-3}}$

faster.

In our experiments, the parameters of the posterior probability were fixed, but in actual image processing, these parameter values are estimated from the observed images as well as the denoised image. This parameter estimation is usually performed using the maximum likelihood estimation method using the EM algorithm [9]. In typical denoising methods using Gaussian Markov random field, the EM algorithm requires the calculation of the covariance matrix (the inverse of the precision matrix), which also involves solving simultaneous linear equations [10]. In parameter estimation using the EM algorithm, we need to solve simultaneous linear equations many times, so a fast method for solving simultaneous linear equations is also important. When solving similar simultaneous linear equations many times, the ICCG method may be more effective than the CG method because the incomplete Cholesky decomposition only needs to be calculated once. In the future, we will compare and verify the performance of the CG method and the ICCG method in image denoising problem including

parameter estimation.

References

[1] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.PAMI-6, no.6, pp.721–741, 1984.

[2] K. Tanaka, "Statistical-mechanical approach to image processing," Journal of Physics A: Mathematical and General, vol.35, no.37, p.R81, 2002.

[3] S.Z. Li, Markov Random Field Modeling in Image Analysis, Springer, 2009.

[4] H. Nishimori, "Bussei kenkyu (in japanese)," vol.73, no.5, pp.850–857, 2000.

[5] M. Yasuda, J. Watanabe, S. Kataoka, and K. Tanaka, "Linear-time algorithm in bayesian image denoising based on gaussian markov random field," vol.E101-D, no.6, pp.1629–1639, 2018.

[6] R.S. Varga, Matrix Iterative Analysis, Springer, 2000.

[7] M.R. Hestense and E. Stiefel, "Methods of conjugate gradients for solving linear systems," Journal of Research of the National Bureau of Standards, vol.49, no.6, pp.409–436, 1952.

[8] G.H. Golub and C.F. Van Loan, Matrix Computations, Jphns Hopkins University Press, 2012.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," Journal of the royal statistical society: series B (methodological), vol.39, no.1, pp.1–22, 1977.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, Numerical Recipes in C, Cambridge University Press, 1992.