Scalable Data Parallel Algorithms for Texture Synthesis using Gibbs Random Fields

David A. Bader* Joseph JáJá† Rama Chellappa‡
{dbader,joseph,chella}@eng.umd.edu
Department of Electrical Engineering, and Institute for Advanced Computer Studies,
University of Maryland, College Park, MD 20742

Submitted: October 4, 1993 Revised: July 28, 1994

Abstract

This paper introduces scalable data parallel algorithms for image processing. Focusing on Gibbs and Markov Random Field model representation for textures, we present parallel algorithms for texture synthesis, compression, and maximum likelihood parameter estimation, currently implemented on Thinking Machines CM-2 and CM-5. Use of fine-grained, data parallel processing techniques yields real-time algorithms for texture synthesis and compression that are substantially faster than the previously known sequential implementations. Although current implementations are on Connection Machines, the methodology presented here enables machine independent scalable algorithms for a number of problems in image processing and analysis.

Keywords: Gibbs Sampler, Gaussian Markov Random Fields, Image Processing, Texture Synthesis, Data Parallel Algorithms.

1 Introduction

Random Fields have been successfully used to sample and synthesize textured images (e.g. [6], [4], [9], [7], [5]). Texture analysis has applications in image segmentation and classification, biomedical image analysis, and automatic detection of surface defects. Of particular interest are the models that specify the statistical dependence of the grey level at a pixel on those of its neighborhood. There are several well-known algorithms describing the sampling process for generating synthetic textured images, and algorithms that yield an estimate of the parameters of the assumed random process given a textured image. Impressive results related to real-world imagery have appeared in the literature ([6], [7], [5], [8], [3]). However, all these algorithms are quite computationally

*The support by NASA Graduate Student Researcher Fellowship No. NGT-50951 is gratefully acknowledged.
†Supported in part by NSF grant No. CCR-9103135 and NSF HPCC/GCAG grant No. BIR-9318183.
‡Supported in part by Air Force grant No. F49620-92-J0130.

demanding because they typically require on the order of $G\,n^2$ arithmetic operations per iteration for an image of size $n \times n$ with G grey levels. The implementations known to the authors are slow and operate on images of size 128×128 or smaller.

In this paper, we develop scalable data parallel algorithms for implementing the most important texture sampling and synthesis algorithms. The data parallel model is an architecture-independent programming model that allows an arbitrary number of virtual processors to operate on large amounts of data in parallel. All the algorithms described in this paper have been implemented and thoroughly tested on a Connection Machine CM-2 and a Connection Machine CM-5.

Section 2 develops parallel algorithms for texture synthesis using Gibbs and Gaussian Markov Random Fields. Parameter estimation for Gaussian Markov Random Field textures, using least squares, as well as maximum likelihood techniques, are given in Section 3. Conclusions are given in Section 4.

2 Texture Synthesis

2.1 A Parallel Gibbs Sampler

A discrete Gibbs random field (GRF) is specified by a probability mass function of the image as follows:

$$\Pr(X = x) = \frac{e^{-U(x)}}{Z}\,, \tag{1}$$

where $U(x)$ is the energy function, and $Z = \sum U(x)$, over all G^n images; G being the number of grey levels, and the image is of size $\sqrt{n} \times \sqrt{n}$. Except in very special circumstances, it is not feasible to compute Z. A relaxation-type algorithm described in [6] simulates a Markov chain through an iterative procedure that re-adjusts the grey levels at pixel locations during each iteration. This algorithm sequentially initializes the value of each pixel using a uniform distribution. Then a single pixel location is selected at random, and using the conditional distribution that describes the

Markov chain, the new grey level at that location is selected, dependent only upon the grey levels of the pixels in its local neighborhood. The sequential algorithm terminates after a given number of iterations.

The sequential algorithm to generate a Gibbs random field described in [6] and [7] are used as a basis for our parallel algorithm. For all the algorithms given in this paper, we use a symmetric neighborhood N_s which is half the size of the standard neighborhood model N. This implies that if the vector $(\imath, \jmath) \in N$, then $(-\imath, -\jmath) \in N$, but only one of $\{(\imath, \jmath), (-\imath, -\jmath)\}$ is in N_s. Each element of array Θ is taken to represent the parameter associated with its corresponding element in N_s. We use the notation y_σ to represent the grey level of the image at pixel location σ.

Our Gibbs random field is generated using a simulated annealing type process. For an image with G grey levels, the probability $\Pr(X = k \mid \text{neighbors})$ is binomial with parameter $\Psi(T) = \frac{e^T}{1+e^T}$, and number of trials $G-1$. The array $\{\mathrm{T}\}$ is given in the following equation for a first-order model:

$$\mathrm{T} = \alpha + \theta_{(1,0)}(y_{\sigma+(1,0)} + y_{\sigma-(1,0)}) + \theta_{(0,1)}(y_{\sigma+(0,1)} + y_{\sigma-(0,1)}) \tag{2}$$

and is a weighted sum of neighboring pixels at each pixel location. Additional examples of $\{\mathrm{T}\}$ for higher order models may be found in [6].

This algorithm is ideal for parallelization. The calculation of $\{\mathrm{T}\}$ requires uniform communications between local processing elements, and all other operations needed in the algorithm are data independent, uniform at each pixel location, scalable, and simple.

An example of a binary synthetic texture generated by the Gibbs Sampler is given in Figure 4.

Table 1 shows the timings of a binary Gibbs sampler for model orders 1, 2, and 4, on the CM-5. More extensive tables for both the CM-2 and CM-5 can be found in [1].

2.2 Gaussian Markov Random Field Sampler

In this section, we consider the class of 2-D non-causal models called the Gaussian Markov random field (GMRF) models described in ([3], [5], and [9]). Pixel grey levels have joint Gaussian distributions and correlations controlled by a number of parameters representing the statistical dependence of a pixel value on the pixel values in a symmetric neighborhood. There are two basic schemes for generating a GMRF image model, both of which are discussed in [3]. The Iterative Gaussian Markov Random Field Sampler is similar to the Gibbs Sampler, but instead of the binomial distribution, we use the continuous Gaussian Distribution as the probability function. An efficient parallel implementation is straightforward and similar to that of the Gibbs Sampler.

The previous section outlined an algorithm for sampling GMRF textured images using an iterative method. Unfortunately, this algorithm may have to perform hundreds or even thousands of iterations before a stable texture is realized. Next we present a scheme which makes use of two-dimensional Fourier transforms and does not need to iterate. The Direct GMRF Sampler algorithm is realized from [3] as follows. We use the following scheme to reconstruct a texture from its parameters Θ and a neighborhood N_s:

$$\mathbf{y} = \frac{1}{M^2} \sum_{\sigma \in \Omega} \boldsymbol{f}_\sigma \, \frac{x_\sigma}{\sqrt{\mu_\sigma}} \tag{3}$$

where $\mathbf{y}$ is the resulting M^2 array of the texture image, and

$$x_\sigma = \boldsymbol{f}_\sigma^{*t} \, \boldsymbol{\eta} \ , \qquad \mu_\sigma = (1 - 2\Theta^T \Phi_\sigma), \forall \sigma \in \Omega, \qquad \Phi_\sigma = \ \mathrm{Col}[\cos \frac{2\pi}{M} \sigma^t r, r \in N_s]. \tag{4}$$

The sampling process is as follows. We begin with $\boldsymbol{\eta}$, a Gaussian zero mean noise vector with identity covariance matrix. We generate its the Fourier series, via the Fast Fourier Transform, using $\boldsymbol{f}_\sigma$, the Fourier vector defined below, and finally apply (3).

$$\boldsymbol{f}_\sigma \ = \ \mathrm{Col}[1, \lambda_i \ , \lambda_i^2 \ \boldsymbol{t}_j \ , \ldots \ , \lambda_i^{M-1} \ \boldsymbol{t}_j \ ], \text{is an } M^2 \text{ vector}, \tag{5}$$

$$\boldsymbol{t}_j \ = \ \mathrm{Col}[1, \lambda_j \ , \lambda_j^2 \ , \ldots \ , \lambda_j^{M-1} \ ], \text{is an } M\text{-vector, and} \quad \lambda_i = \exp\left(\sqrt{-1} \tfrac{2\pi i}{M}\right). \tag{6}$$

3 Parameter Estimation for GMRF Textures

Given a real textured image, we wish to determine the parameters of a GMRF model which could be used to reconstruct the original texture through the samplers given in the previous section.

This section develops parallel algorithms for estimating the parameters of a GMRF texture. The methods of least squares (LSE) and of maximum likelihood (MLE), both described in [3], are used. We present efficient parallel algorithms to implement both methods. The MLE performs better than the LSE. This can be seen visually by comparing the textures synthesized from the LSE and MSE parameters, or by noting that the asymptotic variance of the MLE is lower than the LSE ([2], [10]).

3.1 Least Squares Estimate of Parameters

The least squares estimate detailed in [3] assumes that the observations of the GMRF image $\{y_\sigma\}$ obey the model

$$y_\sigma = \sum_{r \in N_s} \Theta_r [y_{\sigma+r} + y_{\sigma-r}] + e_\sigma, \quad \forall \sigma \in \Omega, \tag{7}$$

where $\{e_\sigma\}$ is a zero mean correlated noise sequence with variance ν and correlation with the following structure:

$$E(e_\sigma e_r) = -\Theta_{\sigma-r}\nu, \text{ if } (\sigma - r) \in N, \qquad \nu \text{ if } \sigma = r, \qquad 0 \text{ otherwise.} \tag{8}$$

Then, for $\boldsymbol{g}_\sigma = \text{Col}[y_{\sigma+r'} + y_{\sigma-r'}, r' \in N_s]$, the LSE are:

$$\Theta^\star = \left[\sum_\Omega \boldsymbol{g}_\sigma \ \boldsymbol{g}_\sigma^t\right]^{-1} \left(\sum_\Omega \boldsymbol{g}_\sigma \ y_\sigma\right) \qquad \nu^\star = \frac{1}{M^2} \sum_\Omega \left(y_\sigma - \Theta^{\star t} \boldsymbol{g}_\sigma \ \right)^2 \tag{9}$$

where Ω is the complete set of M^2 pixels, and toroidal wrap-around is assumed.

3.2 Maximum Likelihood Estimate of Parameters

We introduce the following approach as an improved method for estimating GMRF parameters of textured images. The method of maximum likelihood gives a better estimate of the texture

parameters, since the asymptotic variance of the MLE is lower than that of the LSE. We also show a much faster algorithm for optimizing the joint probability density function which is an extension of the Newton-Raphson method and is also highly parallelizable.

Assuming a toroidal lattice representation for the image $\{y_\sigma\}$ and Gaussian structure for noise sequence $\{e_\sigma\}$, the joint probability density function is the following:

$$p(y|\Theta,\nu) = \frac{1}{(2\pi\nu)^{\frac{M^2}{2}}} \sqrt{\prod_{\sigma\in\Omega}\Big(1 - 2\sum_{\tau_i\ \in N_s}\big(\Theta_{\tau_i}\ \Phi_{\tau_i}\ (\sigma)\big)\Big)}\quad e^{-\frac{1}{2\nu}\left[C(0)\quad -\quad \sum_{\tau_i\ \in N}\big(\Theta_{\tau_i}\ C(\tau_i\ )\big)\right]} \tag{10}$$

In (10), $C(\tau_i\ )$ is the sample correlation estimate at lag τ_i . As described in [2] and [3], the log-likelihood function can be maximized: (Note that $F(\Theta,\nu) = \log p(y|\Theta,\nu)$).

$$\begin{aligned} F(\Theta,\nu) \quad = \quad - \quad \frac{M^2}{2}\log 2\pi\nu \quad + \quad \frac{1}{2}\sum_{\sigma\in\Omega}\Big(\log\big(1 - 2\sum_{\tau_i\ \in N_s}\big(\Theta_{\tau_i}\ \Phi_{\tau_i}\ (\sigma)\big)\big)\Big) \\ - \frac{1}{2\nu}\sum_{\sigma\in\Omega}\Big(y(\sigma)^2 - y(\sigma)\sum_{\tau_i\ \in N_s}\big(\Theta_{\tau_i}\ \big(y(\sigma+r_i) + y(\sigma-r_i)\big)\big)\Big) \end{aligned} \tag{11}$$

For a square image, Φ_{τ_i} is given as follows:

$$\Phi_{\tau_i}\ (\sigma) \quad = \quad \cos\left(\frac{2\pi}{M}\sigma^T r_i\right) \tag{12}$$

This non-linear function $\mathbf{F}$ is maximized by using an extension of the Newton-Raphson method. This new method first generates a *search direction* ϑ^k by solving the system

$$[\nabla^2 F(\Theta_k)]_{(r+1)\times(r+1)}\ [\vartheta^k]_{(r+1)\times 1} = -[\nabla F(\Theta_k)]_{(r+1)\times 1}. \tag{13}$$

Note that this method works well when $\nabla^2 F(\Theta_k)$ is a symmetric, positive-definite Hessian matrix. We then *maximize* the step in the search direction, yielding an approximation to λ_k which attains the local maximum of $F(\Theta_k + \lambda\vartheta)$ and also satisfies the constraints that each of the M^2 values in the logarithm term for $\mathbf{F}$ is positive. Finally, an *optimality test* is performed. We set $\Theta_{k+1} = \Theta_k + \lambda\vartheta$, and if Θ_{k+1} is sufficiently close to Θ_k, the procedure terminates. We give the first and second derivatives of $\mathbf{F}$ with respect to Θ_k and ν in [1].

For a rapid convergence of the Newton-Raphson method, it must be initialized with a good estimate of parameters close to the global maximum. We use the least squares estimate given in Subsection 3.1 as Θ_0, the starting value of the parameters.

In Figure 5, we show the synthesis using least squares and maximum likelihood estimates for tree bark obtained from standard textures library. Table 2 shows the respective parameters for both the LSE and MLE and give their log-likelihood function values. This example shows that the maximum likelihood estimate improves the parameterization. In addition, CM-5 timings for these estimates varying machine size, image size, and neighborhood models can be found in Figure 6 for both fourth and higher order models on this selection of real world textured images. The value plotted is the mean time over thirteen diverse images, and errors bars give the standard deviation. More explicit tables, as well as CM-2 timings, for these estimates can be found in [1].

4 Conclusions

We have presented efficient data parallel algorithms for texture analysis and synthesis based on Gibbs or Markov random field models. A complete software package running on the Connection Machine model CM-2 and the Connection Machine model CM-5 implementing these algorithms is available for distribution to interested parties. The experimental data strongly support the analysis concerning the scalability of our algorithms. The same type of algorithms can be used to handle other image processing algorithms such as image estimation ([8], [9]), texture segmentation ([5]), and integration of early vision modules. We are currently examining several of these extensions.

The parameters for the 256×256 image of tree bark texture in Figure 5 are given in Table 2.

References

[1] D. A. Bader, J. JáJá, and R. Chellappa. Scalable Data Parallel Algorithms for Texture Synthesis and Compression Using Gibbs Random Fields. Technical Report CS-TR-3123 and UMIACS-TR-93-80, UMIACS and Electrical Engineering, University of Maryland, College Park, MD, August 1993.

[2] J. E. Besag and P. A. P. Moran. On the Estimation and Testing of Spacial Interaction in Gaussian Lattice Processes. *Biometrika*, 62:555–562, 1975.

[3] R. Chellappa. Two-Dimensional Discrete Gaussian Markov Random Field Models for Image Processing. In L. N. Kanal and A. Rosenfeld, editors, *Progress in Pattern Recognition*, volume 2, pages 79–112. Elsevier Science Publishers B. V., 1985.

[4] R. Chellappa and R. L. Kashyap. Synthetic Generation and Estimation in Random Field Models of Images. In *IEEE Comp. Soc. Comf. on Pattern Recog. and Image Processing*, pages 577–582, Dallas, TX, August 1981.

[5] F. S. Cohen. Markov Random Fields for Image Modelling & Analysis. In U. Desai, editor, *Modelling and Applications of Stochastic Processes*, chapter 10, pages 243–272. Kluwer Academic Press, Boston, MA, 1986.

[6] G. R. Cross and A. K. Jain. Markov Random Field Texture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5:25–39, January 1983.

[7] R. C. Dubes and A. K. Jain. Random Field Models in Image Analysis. *Journal of Applied Statistics*, 16:131–164, 1989.

[8] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, November 1984.

[9] F. C. Jeng, J. W. Woods, and S. Rastogi. Compound Gauss-Markov Random Fields for Parallel Image Processing. In R. Chellappa and A. K. Jain, editors, *Markov Random Fields: Theory and Application*, chapter 2, pages 11–38. Academic Press, Boston, MA, 1993. Bell Communications Research and ECSE Department, Renssalaer Polytechnic Institute.

[10] R. L. Kashyap and R. Chellappa. Estimation and Choice of Neighbors in Spacial Interaction Models of Images. *IEEE Transactions on Information Theory*, IT-29:60–72, January 1983.

A Figures and Tables

Algorithm 1 *Direct Gaussian MRF Sampler*
Reconstruct a GMRF texture from parameters, assuming toroidal wrap-around and an M^2 image size
Input:
$\Theta \leftarrow$ the set of parameters for the given model.
$\{ G \}$ is the number of grey levels.
image $\leftarrow$ a parallel variable for the image. (Complex)
$\{\Phi_r\} \leftarrow$ a parallel variable with serial elements for each parameter in the model.
begin
1. Initialize the real part of the image in parallel to Gaussian noise
   with mean = 0 and standard deviation = 1.
2. Initialize the imaginary part of the image in parallel to 0.
3. Divide the image by $\sqrt{\nu}$.
4. Perform a parallel, in-place FFT on the noise.
5. **For all** pixels σ **in parallel do**:
   5.1 **For each** $r \in N_s$, $\Phi_r = \cos \frac{2\pi}{M} \sigma^t r$
   5.2 Calculate μ_σ from Equation (4).
   5.3 Divide the image by $\sqrt{\mu_\sigma}$.
6. Perform a parallel, in-place, inverse FFT on the image.
7. Scale the result to grey levels in the interval [0..G-1].

end

Figure 1: Gaussian MRF Sampler Algorithm

Image	*Order = 1*		*Order = 2*		*Order = 4*	
Size	16/vu CM-5	32/vu CM-5	16/vu CM-5	32/vu CM-5	16/vu CM-5	32/vu CM-5
8k	0.046053	0.024740	0.051566	0.027646	0.068486	0.038239
16k	0.089822	0.046824	0.099175	0.052411	0.130501	0.068630
32k	0.176997	0.089811	0.199399	0.099493	0.252421	0.132646
64k	0.351123	0.178046	0.398430	0.194271	0.560224	0.257647
128k	0.698873	0.351517	0.759017	0.383425	0.943183	0.582303
256k	1.394882	0.700164	1.526422	0.759747	1.874973	0.962165
512k	2.789113	1.394216	3.047335	1.520437	3.744542	1.892460
1M	5.577659	2.782333	6.009608	3.063054	7.428823	3.785890

Table 1: Gibbs Sampler timings for a binary ($G = 2$) image (execution time in seconds per iteration on a CM-5 with vector units)

Algorithm 2 *Least Squares Estimator for GMRF*
Using the method of Least Squares, estimate the parameters of image Y.
Assume toroidal wrap-around, an M^2 image size, and a given neighborhood.
Input:
$\{\mathbf{Y}\} \leftarrow$ the image.
$\Theta \leftarrow$ the scalar array of parameter estimates for each neighborhood element.
begin
1. **For all** pixels **in parallel do**:
   1.1 **For each** $r \in N_s$ **do**
      1.1.1 $\mathbf{g}_\sigma[r] = y_{\sigma+r} + y_{\sigma-r}$
   1.2 **For** $\imath$ from 1 to $|N_s|$ **do**
      1.2.1 **For** $\jmath$ from 1 to $|N_s|$ **do**
         1.2.1.1 **Calculate** $gcross_\sigma[\imath,\jmath] = \mathbf{g}_\sigma[\imath] \times \mathbf{g}_\sigma[\jmath]$.
2. **For** $\imath$ from 1 to $|N_s|$ **do**
   2.1 **For** $\jmath$ from 1 to $|N_s|$ **do**
      2.1.1 Compute **in parallel** the sum $gmatrix[\imath,\jmath] = \sum_{\sigma\in\Omega} gcross_\sigma[\imath,\jmath]$.
3. **For all** pixels σ **in parallel do**:
   3.1 **For each** $r \in N_s$ **do**
      3.1.1 **Calculate** $gv_\sigma[r] = \mathbf{g}_\sigma[r] \times y_\sigma$
4. **For each** $r \in N_s$ **do**
   4.1 Compute **in parallel** the sum $gvec[r] = \sum_{\sigma\in\Omega} gv_\sigma[r]$
5. Solve the $|N_s| \times |N_s|$ linear system of equations:
   $[gmatrix]_{|N_s|\times|N_s|} \times [\Theta^\star]_{|N_s|\times 1} = [gvec]_{|N_s|\times 1}$
6. **Calculate** $\nu^\star = \frac{1}{M^2} \sum_{\sigma\in\Omega} (y_\sigma - \Theta^{\star t}\mathbf{g}_\sigma)^2$
end

Figure 2: Least Squares Estimator Algorithm

Parameter	LSE	MLE
(1,0)	0.590927	0.568643
(0,1)	0.498257	0.497814
(1,1)	-0.281546	-0.272283
(-1,1)	-0.225011	-0.219671
(0,2)	-0.125950	-0.128427
(2,0)	-0.203024	-0.162452
(2,2)	-0.014322	-0.017466
(2,-2)	-0.002711	-0.007541
(3,0)	0.060477	0.034623
(0,3)	0.024942	0.015561
(4,0)	-0.019122	-0.006186
(0,4)	-0.009040	-0.003748
(-2,1)	0.045105	0.036778
(1,-2)	0.031217	0.040860
(1,2)	0.061537	0.067912
(2,1)	0.067865	0.055445
ν	22205.84	65.45
$F(\Theta)$	-266147.34	-264245.13

Table 2: Θ Parameters for Tree Bark Texture

Algorithm 3 *Maximum Likelihood Estimator for GMRF*
Note that $\Theta_k \equiv \langle \theta_1, \theta_2, \ldots, \theta_r, \nu \rangle$.
Also, this algorithm assumes toroidal wrap-around of the image.
Note that in Step 5, $\beta < 1.0$, and we use $\beta = 0.8$.
Input:
$\{\mathbf{Y}\} \leftarrow$ the image.
begin
1. **Find** Initial Guess Θ_0 using LSE Algorithm 2.
2. **Compute** $\nabla F(\Theta_k) \equiv \langle \frac{\partial F}{\partial \theta_1} , \frac{\partial F}{\partial \theta_2} , \ldots , \frac{\partial F}{\partial \theta_r} , \frac{\partial F}{\partial \nu} \rangle$.
3. **Compute** $\nabla^2 F(\Theta_k)$
4. **Solve** the following linear system of equations for vector ϑ
   $[\nabla^2 F(\Theta_k)]_{(r+1)\times(r+1)} \; [\vartheta]_{(r+1)\times 1} = -[\nabla F(\Theta_k)]_{(r+1)\times 1}$
5. **Determine** the largest λ *from* $\{1, \beta, \beta^2, \beta^3, \ldots\}$ such that
   (5a.) $1 - 2 \sum_{\tau_i \in N_s} \left(\Theta_{\tau_i} \; \Phi_{\tau_i} \; (\sigma) \right) \; > 0$; (note that these represent M^2 constraints)
   (5b.) $F(\Theta_k + \lambda \vartheta) > F(\Theta_k)$
6. **Set** $\Theta_{k+1} = \Theta_k + \lambda \vartheta$
7. If $|F(\Theta_{k+1}) - F(\Theta_k)| > \epsilon$ then go to Step 2.

end

Figure 3: Maximum Likelihood Estimator Algorithm

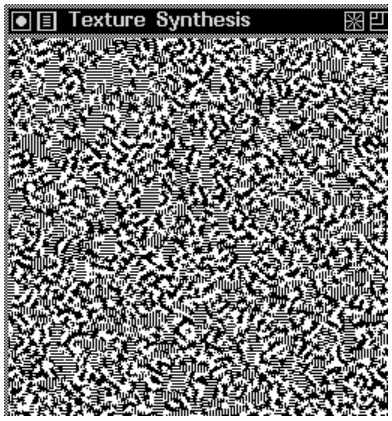

Figure 4: Isotropic Inhibition Texture using Gibbs Sampler (Texture 9b from [6]).

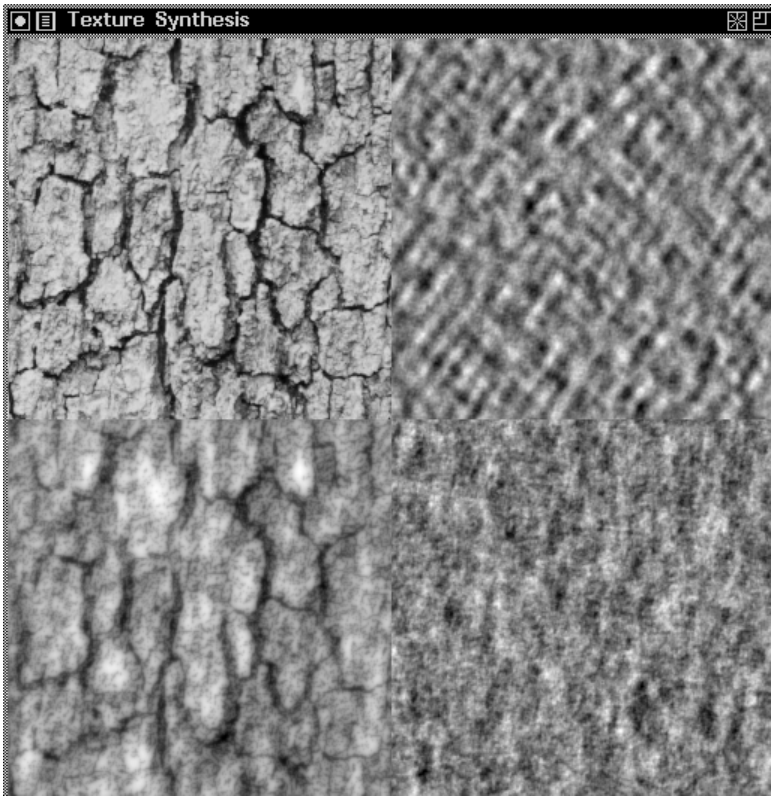

Figure 5: Tree Bark Texture: (clockwise from top left) original image, reconstructed from the LSE, MLE, and Compressed image. A model whose parameters are listed below was used.

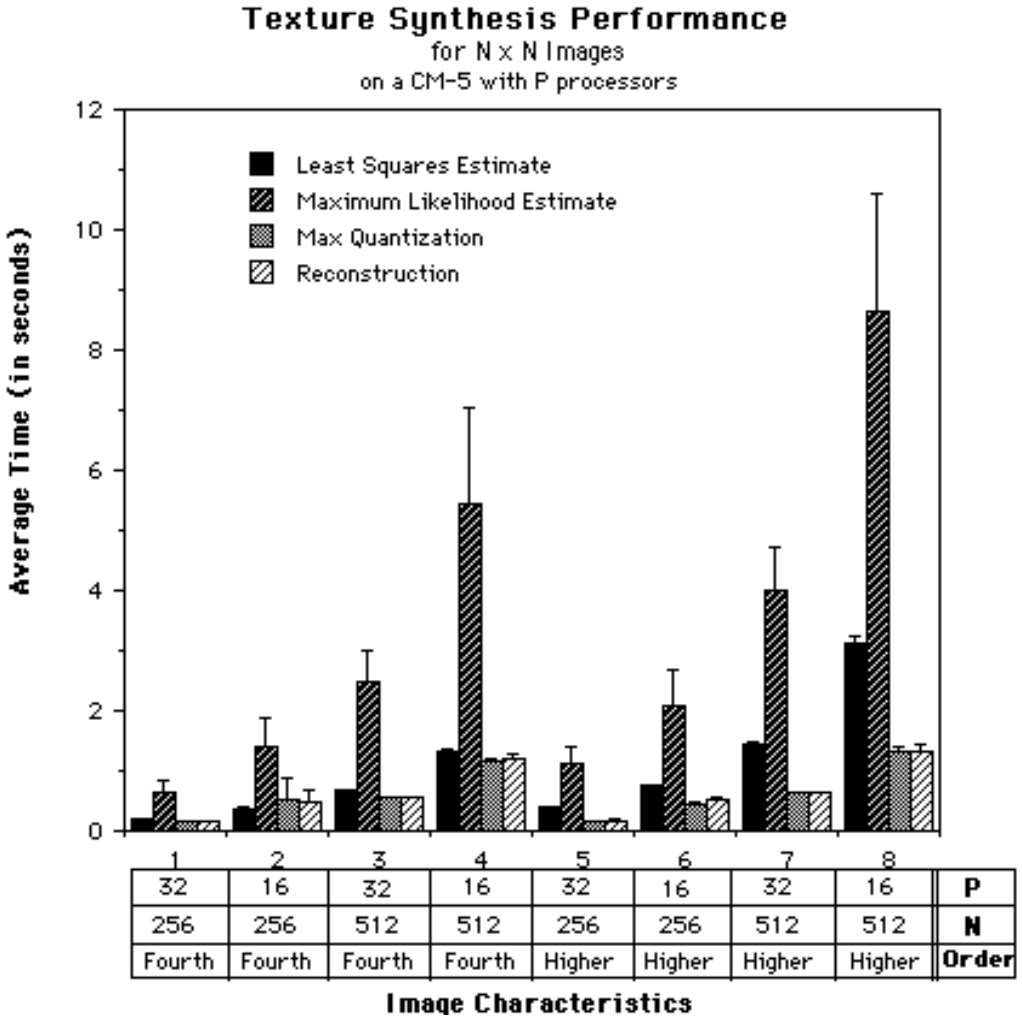

Figure 6: Timings for parallel image processing techniques