Adaptive Image Restoration and Segmentation Method Using Different Neighborhood Sizes

Chengcheng Li and William J. B. Oldham, Senior Member, IEEE

Texas Tech University
Department of Computer Science
Texas Tech University, Lubbock, TX, 79409, USA

ABSTRACT

The image restoration methods based on the Bayesian's framework and Markov random fields (MRF) have been widely used in the image-processing field. The basic idea of all these methods is to use calculus of variation and mathematical statistics to average or estimate a pixel value by the values of its neighbors. After applying this averaging process to the whole image a number of times, the noisy pixels, which are abnormal values, are filtered out. Based on the Tea-trade model, which states that the closer the neighbor, more contribution it makes, almost all of these methods use only the nearest four neighbors for calculation. In our previous research [1, 2], we extended the research on CLRS (image restoration and segmentation by using competitive learning) algorithm to enlarge the neighborhood size. The results showed that the longer neighborhood range could improve or worsen the restoration results. We also found that the autocorrelation coefficient was an important factor to determine the proper neighborhood size. We then further realized that the computational complexity increased dramatically along with the enlargement of the neighborhood size. This paper is to further the previous research and to discuss the tradeoff between the computational complexity and the restoration improvement by using longer neighborhood range. We used a couple of methods to construct the synthetic images with the exact correlation coefficients we want and to determine the corresponding neighborhood size. We constructed an image with a range of correlation coefficients by blending some synthetic images. Then an adaptive method to find the correlation coefficients of this image was constructed. We restored the image by applying different neighborhood CLRS algorithm to different parts of the image according to its correlation coefficient. Finally, we applied this adaptive method to some real-world images to get improved restoration results than by using single neighborhood size. This method can be extended virtually on all the methods based on MRF framework and result in improved algorithms.

Keywords: Bayesian's theorem, Markov random field, Gibb's distribution, Glifford Hammersley theorem, clique, competitive learning, Cholesky decomposition, Toplitz-matrix.

1. INTRODUCTION

Image restoration is usually the first step of the whole image processing process. It increases the definition of the image, gets rid of the noisy pixels by averaging, threshoulding or applying masks, and estimates the distorted or missing part of the image. Image segmentation and edge detection can be also counted as parts of image restoration process because we cannot restore the image by simply averaging the pixel values, which leads to a blurred image. Instead, we need to preserve the most important information – edges during the averaging process. The pixel value averaging and edge preservation processes should be done simultaneously during each update of the image.

Many researchers based their image restoration methods on calculus of variation and mathematical statistics. Bayesian's framework is the foundation of these methods. We can model the image as F = (F), where F is the matrix of pixel intensities. According to Bayesian's theorem

$$P(F \mid g) = \frac{P(g \mid F)P(F)}{P(g)} \qquad (1)$$

F can be calculated by a given image g, if P(g|F), P(F) and P(g) can be found. When we consider the pixel matrix as a MRF, we then can describe it as in Gibb's form by Clifford_Hammersley theorem. Then the problem becomes a minimization problem.

Geiger and Girosi's[3] PDAM[4] (parallel and deterministic algorithms from MRF's) first introduced line processes and described an image as a triplet F=(F,H,V), where H and V correspond to the matrix of horizontal and vertical edge elements. Thus, F is referred to as an intensity process, H as horizontal line process, and V as vertical line process. Then, the problem becomes a minimization problem of the form $\min_F E[(F, H, V)| g]$ according to Gibb's distribution.

The assumption that observations lie in an simplified MRF leads the computations to be restricted to immediate neighborhoods, for example, a pixel f_{ij} can only interact with $f_{i,j+1}$, $f_{i,j-1}$, $f_{i+1,j}$, and $f_{i-1,j}$. According to the tea-trade model, the closer of a neighbor, more contribution it will make. However, the ignorance of the further neighbors can lead to the missing of information. Our previous research showed the close relation between the autocorrelation coefficient of the image and the corresponding neighborhood size for restoration. We concluded that we should use the proper neighborhood size for each particular image according to the correlation coefficient of the image.

An image usually has a range of autocorrelation coefficients on different parts of it. Thus, to construct an adaptive method or set the criteria to determine the right neighborhood size for different parts of image for restoration is necessary. The following sections of this paper are going to discuss this problem in detail.

2. CLRS ALGORITHM

CLRS method uses the framework mentioned above and competitive learning concept, which is based on principle of low-level mammalian visual system. It deals with image restoration and segmentation problems from a more direct and easily understandable and acceptable aspect.

Let g = $[g_{ij}]_{mn}$, denote the m x n matrix corresponding to the observed intensity values. Here g_{ij} denotes the observed intensity value at the (i,j)th point on the lattice. G denotes the prior estimates of the image. Assume, now we have a two-layer network, with layers numbered as L and M. The priors g_{ij} lies in layer L and corresponding to each (i, j) in layer M. Intensity value at point (i, j) in M denoted by f_{ij} represents the posterior estimate. Let g_{mk} in the neighborhood in layer L compete to update the value fij. In layer M. Contribution of each g_{ij} to f_{ij} is determined by the function C(K,d), where K is the gain parameter and d is given by

$$d = f_{ij} - g_{mk} \qquad (m,k) \in \tau_{ij} \tag{2}$$

and

$$C(K,d) = e^{-d^2/(2K^2)} \tag{3}$$

Normalizing C(K,d) over the neighborhood Γij we have

$$\Lambda[(i,j),(m,k),K] = \frac{e^{-d^2/(2K^2)}}{x[(i,j),K]} \tag{4}$$

where $x[(i,j),K] = \sum_{mk} e^{-d^2/(2K^2)}$. As a result, when d, the distance between two pixel values is small, the contribution function is large, hence, the closest contributing the most. This concept is analogous to the leaky learning form of competitive learning.

The update rule is given by the following equation

$$\Delta f_{ij} = \alpha \sum_{mk} \Lambda((i,j),(m,k).K)(g_{mk} - f_{ij})$$
$$- \beta K\left(\frac{\partial E_l}{\partial f_{ij}} + \frac{\partial E_i}{\partial f_{ij}}\right) \tag{5}$$

The first term of this equation has been discussed above where α denotes the learning rate. The second term is to smooth the discontinuities between the posterior neighbors, which adds the line processes onto the image during each update iteration. β is the parameter to control the weight of the edges we add. If β is too large, the energy function cannot converge; if β is small, it will lead to a blurred image.

The following figure shows a neighborhood size of nearest eight pixels.

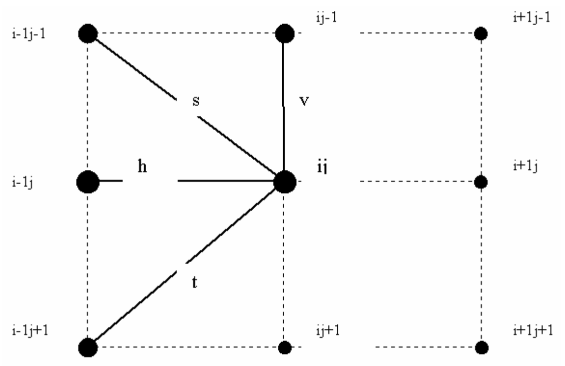

Figure 1. Eight-neighbor System

When we only consider the contribution from the immediate neighbors which are the nearest four pixel points (i, j-1), (i+1, j), (i,j+1), and (i-1,j), E_l and E_i can be calculated by the following formulas.

$$E_l = \sum (f_{i,j} - f_{i,j-1})^2 (1 - h_{ij}) + L_H \sum h_{ij}$$
$$+ \sum (f_{i,j} - f_{i-1,j-1})^2 (1 - v_{ij}) + L_V \sum v_{ij} \tag{6}$$

is the line process term, and

$$E_i = I_{ch} \sum h_{i,j} h_{i,j+1} + I_{cv} \sum v_{i,j} v_{i+1,j}$$
$$+ I_h \sum h_{i,j} [(1 - h_{i+1,j} - v_{i,j} - v_{i,j+1})^2 + (1 - h_{i-1,j} - v_{i-1,j} - v_{i-1,j+1})^2]$$
$$+ I_v \sum v_{i,j} [(1 - v_{i,j+1} - h_{i,j} - h_{i+1,j})^2 + (1 - v_{i,j-1} - h_{i,j-1} - h_{i+1,j-1})^2] \tag{7}$$
$$+ I_{kh} \sum h_{ij} (1 - h_{ij}) + I_{kv} \sum v_{ij} (1 - v_{ij})$$

is the interaction term. The line processes are given by

$$h_{ij} = \tanh(C_h |\Delta^h f_{ij}|) \tag{8}$$

$$v_{ij} = \tanh(C_v \,|\, \Delta^v f_{ij} \,|) \tag{9}$$.

When we add the contribution from the nearest eight neighbors. There are two more line processes s and t.

$$s_{ij} = \tanh(C_s \,|\, \Delta^s f_{ij} \,|)\,, \quad \Delta^s = f_{ij} - f_{i-1,j-1} \tag{10}$$

$$t_{ij} = \tanh(C_t \,|\, \Delta^t f_{ij} \,|)\,, \quad \Delta^t = f_{ij} - f_{i-1,j+1} \tag{11}$$

And E_l and E_i can be calculated by the following formulas.

$$E_l = \sum (f_{i,j} - f_{i,j-1})^2 (1 - h_{ij}) + L_H \sum h_{ij}$$
$$+ \sum (f_{i,j} - f_{i-1,j1})^2 (1 - v_{ij}) + L_V \sum v_{ij}$$
$$+ \sum (f_{i,j} - f_{i-1,j+1})^2 (1 - s_{ij}) + L_s \sum s_{ij} \tag{12}$$
$$+ \sum (f_{i,j} - f_{i-1,j-1})^2 (1 - t_{ij}) + L_t \sum t_{ij}$$

$$E_i = I_{ch} \sum h_{i,j} h_{i,j+1} + I_{cv} \sum v_{i,j} v_{i+1,j} + I_{cs} \sum s_{i,j} s_{i+1,j-1} + I_{ct} \sum t_{i,j} v_{i+1,j+1}$$
$$+ I_h \sum h_{i,j} [(1 - h_{i+1,j} - v_{i,j} - v_{i,j+1} - s_{i,j} - s_{i+1,j+1} - t_{i,j} - t_{i+1,j-1})^2$$
$$+ (1 - h_{i-1,j} - v_{i-1,j} - v_{i-1,j+1} - s_{i-1,j} - s_{i,j+1} - t_{i-1,j} - t_{i,j-1})^2]$$
$$+ I_v \sum v_{i,j} [(1 - v_{i,j+1} - h_{i,j} - h_{i+1,j} - s_{i,j} - s_{i+1,j+1} - t_{i,j} - t_{i+1,j-1})^2$$
$$+ (1 - v_{i,j-1} - h_{i,j-1} - h_{i+1,j-1} - s_{i,j-1} - s_{i+1,j} - t_{i,j-1} - t_{i+1,j-2})^2]$$
$$+ I_s \sum s_{i,j} [(1 - s_{i+1,j+1} - h_{i,j} - h_{i+1,j} - v_{i,j} - v_{i,j+1} - t_{i,j} - t_{i+1,j-1})^2 \tag{13}$$
$$+ (1 - s_{i-1,j-1} - h_{i-1,j-1} - h_{i,j-1} - v_{i-1,j-1} - v_{i-1,j} - t_{i-1,j-1} - t_{i,j-2})^2]$$
$$+ I_t \sum t_{i,j} [(1 - t_{i+1,j-1} - h_{i,j} - h_{i+1,j} - v_{i,j} - v_{i,j+1} - s_{i,j} - s_{i+1,j+1})^2$$
$$+ (1 - t_{i-1,j+1} - h_{i-1,j+1} - h_{i,j+1} - v_{i-1,j+1} - v_{i-1,j+2} - s_{i-1,j+1} - s_{i,j-2})^2]$$
$$+ I_{kh} \sum h_{ij} (1 - h_{ij}) + I_{kv} \sum v_{ij} (1 - v_{ij}) + I_{ks} \sum s_{ij} (1 - s_{ij}) + I_{kt} \sum t_{ij} (1 - t_{ij})$$

Thus, the computational complexity increases dramaticly along with the enlargement of the neighborhood size. During the rest of the paper, we keep the same line processes of the eight-neighbor formula and only modify the first term of the update equation when the neighborhood size is larger than eight.

3. CONSTRUCTION OF SYNTHETIC IMAGES

Convolution mask method

In our previous study, we used convolution mask method to generate the images with the needed correlation coefficient. Here, we also use the idea of convolution mask to generate part of the synthetic image. The basic idea of convolution mask is the following.

To find the exact relation between correlation coefficient and corresponding proper neighborhood size, we need to construct an synthetic image with a Toplitz-like[5, 6] correlation coefficient matrix. To make the problem more explicit, we need to construct a random image with the following correlation matrix, where ρ is the correlation coefficient.

$$R = \begin{bmatrix} & & & \vdots & & & \\ & \ddots & & {\rho_r}^2 & & \iddots & \\ & & \rho_r\rho_c & \rho_{rc} & \rho_r\rho_c & & \\ \cdots & {\rho_c}^2 & \rho_c & 1 & \rho_c & {\rho_c}^2 & \cdots \\ & & \rho_r\rho_c & \rho_r & \rho_r\rho_c & & \\ & \iddots & & {\rho_r}^2 & & \ddots & \\ & & & \vdots & & & \end{bmatrix} \qquad (14)$$

ρ_c and ρ_r are the coefficients of column and row. The size of the matrix can be as big as the whole image if we do not ignore the coefficient with high powers.

Let x(i,j) be a pixel of the random image at i,j position in a two-dimensional random field. Also, let us assume Toeplitz forms for correlation coefficients as

$$\rho[x(i,j), x(i+k, j+l)] = \rho_k^{(c)}\rho_l^{(r)} \qquad (15)$$

The above is the correlation coefficient between x(i,j) and x(i+k,j+l) and can be assumed to be the product of $\rho_k^{(c)}$ and $\rho_l^{(r)}$.

Here, we assume ρ_c = ρ_r .= ρ If ρ is small enough, for example, 0.2, then ρ^2 =0.04 and ρ^3 =0.008. We can ignore ρ^3 and all the coefficients with the power higher than 3. To do this, we can compare the restoration results from four-neighbor and eight-neighbor CLRS programs without considering the contributions from further pixels.

Now, we know what kind of the random image we want; the problem becomes how we can go in retro-direction with the coefficient matrix to generate the image. We implement the generation of the image by using convolution mask.

The most elementary combination of the pixels in a neighborhood is given by an operation which multiplies each pixel in the range of the filter mask with the corresponding weighting factor of the mask, adds up the products, and writes the result to the position of the center pixel. This operation is known as a discrete convolution and is defined as:

$$G'_{mn} = \sum_{m'=-r}^{r}\sum_{n'=-r}^{r} H_{m'n'}G_{m-m',n-n'} = \sum_{m'=-r}^{r}\sum_{n'=-r}^{r} H_{-m',-n'}G_{m+m',n+n'} \qquad (16)$$

This equation assumes an odd-sized mask with $(2r+1)\times(2r+1)$ coefficients.

Let us assume that we perform the convolution line by line and from the left to the right. When we apply the convolution mask onto one single pixel, the values at all pixels positioning above and to the left of this pixel have already processed the correlation properties we need by the previous computed results. The pixels lower and to the right of the current pixel do not have the correlation properties. Consequently, the calculated result of the pixel does not have the perfect match of the correlation properties as the mask since half of the pixels in the mask do not have the desired values and are going to be updated right afterwards. This problem is common to all kinds of neighborhood operations, not only convolutions. The solution to this problem is to apply the mask several times on the original images.

Suppose the correlation coefficient ρ_c = ρ_r .= ρ = 0.8. The weight of the each neighbor is given by

Weight = $\dfrac{\rho_{k,l}^{(K+l)}}{\sum \rho_{k,l}^{(K+l)}}$ Then the convolution must is $\begin{bmatrix} 0.111 & 0.139 & 0.111 \\ 0.139 & 0 & 0.139 \\ 0.111 & 0.139 & 0.111 \end{bmatrix}$.

We applied this mask on a random image ten times and found the distribution of the pixel values tended to be Gaussian and the correlation coefficient mask was in the needed Toplitz matrix format.

LU Decomposition method

The convolution mask works well in generating large correlation coefficient image. When working with small correlation coefficient, even if we only apply the mask three times, it is hard the get ρ lower than 0.5. So, we need to find another method to generate ρ lower than 0.5.

X is the column vector on the needed image. The covariance is Σ=E[(X-M)(X-M)^T], where M is the mean. Then Σ can be further described as Σ=ΓRΓ, where Γ is a diagonal matrix with the diagonal elements $c_{ii} = \sigma_i$, and R is the Toplitz-like correlation coefficient matrix.

$$R = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1n} \\ \rho_{12} & 1 & & \\ \vdots & & \ddots & \vdots \\ \rho_{n1} & & \cdots & 1 \end{bmatrix}$$

An explicit expression of a normal distribution is

$$N_X(M,\Sigma) = \frac{1}{(2\pi)^{1/2}\,|\Sigma|^{1/2}}\exp\{-\frac{1}{2}d^2(X)\}, \qquad (17)$$

where $d^2(X) = (X-M)^T\Sigma^{-1}(X-M)$

From LU decomposition, we got

$\Sigma = \Gamma LU\Gamma$, and $\Sigma^{-1} = \Gamma^{-1}L^{-1}U^{-1}\Gamma^{-1}$ then

$$d^2(X) = -(X-M)^T\Gamma^{-1}U^{-1}L^{-1}\Gamma^{-1}(X-M) \qquad (18)$$

Since R is a symmetric matrix, L=U^T; Γ is diagonal matrix, Γ^{-1}=(Γ^{-1})^T. so,

$$d^2(X) = -[L^{-1}\Gamma^{-1}(X-M)]^T[L^{-1}\Gamma^{-1}(X-M)] \qquad (19)$$

Let us assume $L^{-1}\Gamma^{-1}(X-M) = X'$, then the transfer vector X' is a standard normal vector with zero mean and unit variance. Now we can calculate X by

$$X = L\Gamma X' + M \qquad (20)$$

When we combine X's as the columns to an image, this image will have the Toplitz-matrix like correlation coefficient property on column which is in the vertical direction. This can be verified by calculating the correlation matrix S which is defined by

$$S = E[XX^T] = E[(L\Gamma X')(L\Gamma X')^T] = \Gamma\Gamma^T = R \qquad (21)$$

To let the image have the needed property on the horizontal direction, we can easily follow the above steps except that we need to use row vectors instead of column vectors. To combine these row and column vectors, we get an image by using

$$X = \Gamma_1 \; L_1 X'_1 + \Gamma_2 X'_2 U_2 + M \qquad (22)$$

Here, X is a matrix (the needed image) rather than a vector. Attention, L_1 and U_2 are not from the same LU decomposition of R. For example, if we want to generate an image with 400x512 pixels, $X_1^{'}$ and $X_2^{'}$ are 400x512 matrix with standard normal deviates (zero mean and unit variance). L_1 is from LU decomposition of a 400x400 matrix R_1 while U_2 is from LU decomposition of a 512x512 matrix R_2, where R_1 and R_2 are from (21). Γ_1, Γ_2 and M, which are the standard deviation on vertical and horizontal directions and the mean, can be set arbitrary.

We can actually rotate the above matrix X to any directions and by adding them together, we can generate an image with the needed correlation matrix property on every direction. However, since we only need this image to have correlation coefficient lower than 0.5 and this image is suggested restoring by the nearest four neighbors, only the vertical and horizontal properties are necessary.

By doing LU decomposition to a symmetric matrix, we found a direct way to calculate L and U by an equation instead of using Cholesky decomposition method.

If R is a block Toeplitz matrix which is defined above, and LU=R, then

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \rho & \sqrt{1-\rho^2} & 0 & \cdots & 0 \\ \rho^2 & \rho\sqrt{1-\rho^2} & \sqrt{1-\rho^2} & \cdots & 0 \\ \vdots & & & \ddots & \\ \rho^{n-1} & \rho^{n-2}\sqrt{1-\rho^2} & \rho^{n-3}\sqrt{1-\rho^2} & \cdots & \sqrt{1-\rho^2} \end{bmatrix} \qquad (23)$$

This result can be easily proved by induction proof.

4. ADAPTIVE METHOD

The idea of constructing an adaptive method is to restore the images by automatically detecting the correlation coefficient and using CLRS algorithm with the corresponding neighborhood size. To do this, we need to set the thresholds for using the different neighborhood size. Using the synthetic images generated from methods mentioned above, we found the best restoration results for each correlation coefficient. The details of the relation can be found in the following table.

Table1. Correlation Coefficient and Corresponding Neighborhood Size for Image Restoration by CLRS Algorithm

Average correlation coefficient	**Neighborhood size for best restoration result**
<=0.5 for 1-pixel away horizontally and vertically	Nearest 4 neighbors horizontally and vertically
<=0.7 for 1-pixel away and >=0.5 for moving 1-pixel away in 45 and 135-degree directions	Nearest 8 neighbors in all directions
<=0.7 for 1-pixel away horizontally and vertically and >=0.5 for 2-pixel away horizontally and vertically while the 45 and 135-degree correlation coefficient is below 0.2	Nearest 8 neighbors in the horizontal and vertical directions
<=0.8 for 1-pixel away and >=0.5 for 2-pixel away horizontally and vertically, and >=0.5 for 2-pixel away in 45 and 135-degree directions	Nearest 24 neighbors in all directions
>=0.8 for 1-pixel away and >=0.5 for 3-pixel away horizontally and vertically, and >=0.5 for 3-pixel away in 45 and 135-degree directions.	Nearest 48 neighbors in all directions.

A real-world image usually has a range of correlation coefficients, so we cannot just apply the restoration algorithm on an image with single neighborhood size. We actually need a robotic and adaptive method, which can apply different neighborhood range algorithms to different regions on the image. The solution to this problem is to divide the whole images into regions according to the correlation coefficient. The size of the region is user defined and the correlation coefficient of the region stored to an array. During the restoration process, the regions are restored separately and the different CLRS algorithms are called by the values obtained from the correlation coefficient array. To test the result of this adaptive method, we did experiment on a synthetic image. The image (Figure 2.) is composed of five different images randomly which are the images with the different correlation coefficients described in Table 1. And then, we added ome percent of noise to Figure 1. After that, we restored the image by the nearest four-neighbor, nearest eight-neighbor and the adaptive CLRS algorithms. The restoration results cannot be differentiated by naked eyes. So we compared the results numerically.

5. RESULTS

Results on synthetic image

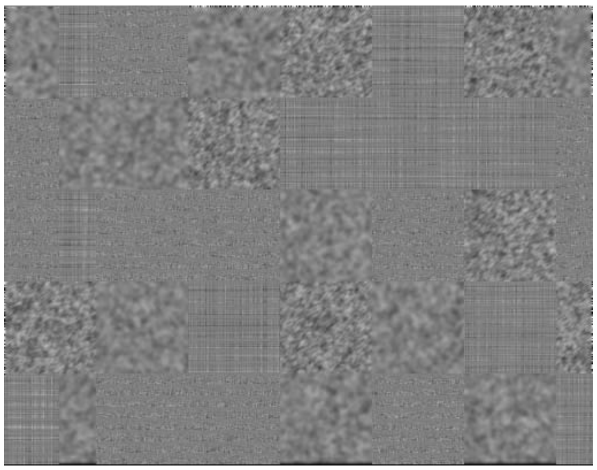

Figure 2. Synthetic Image Composed by Five Images with Correlation Coefficients Shown in Table 1.

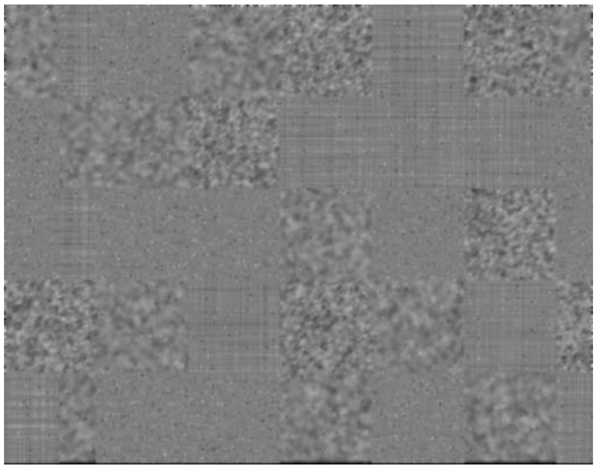
Figure 3. Restored by Nearest Four-neighbor Program

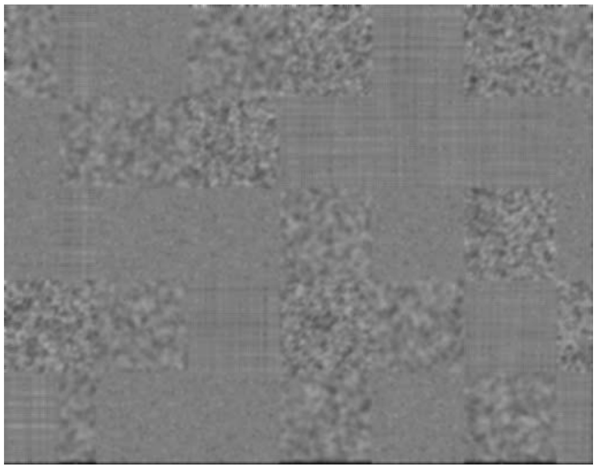
Figure 4. Restored by Nearest Eight-neighbor Program

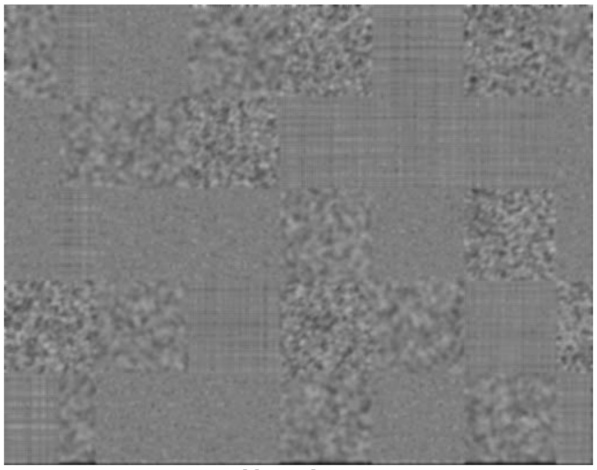
Figure 5. Restored by Adaptive CLRS Program

Table 2. Comparison between Figure 1. and Figure 2.

Original Image	Noisy Image	Sum of the squared difference	Average Pixel Value Difference
Figure 2.	1% noise	6409321	7.28011

Table 3. Comparison of Figure 1. and Figure 3.

Original Image	Restored Image	Sum of the squared pixel difference	Average Pixel Value Difference
Figure 2.	Figure 3.	3695959	5.477226
α	β	κ	Iterations
0.2	0.03	5	20

Table 4. Comparison between Figure 1. and Figure 4.

Original Image	Restored Image	Sum of the difference square	Average Pixel Value Difference
Figure 2.	Figure 4.	4105273	5.830952
α	β	κ	Iterations
0.2	0.03	5	15

Table 5. Comparison between Figure 1. and Figure 5.

Original Image	Restored Image	Sum of the squared difference	Average Pixel Value Difference
Figure 2.	Figure 5.	3220269	5.099020
α	β	κ	Iterations
0.2	0.03	5	15

From Figure 3, 4, 5 and Table 3, 4 and 5, we see that with different neighborhood size, the CLRS algorithm does a good job in getting rid of the noises on the image. The four-neighbor program reduces the pixel value difference from 7.3 to 5.5. Some regions with low correlation coefficient are too much averaged by the nearest eight-neighbor program, and the image looks more blurred than the result from using the four-neighbor program, so the overall pixel value difference does not show improvement. By using the adaptive method, the image does not show too much blur as eight-neighbor program does, and the average pixel value difference is reduced to 5.1, which is about eight percent improvement from using four neighbors and fourteen percent improvement from using eight neighbors.

Results on real-world images

There is a problem to test the restoration result by using the synthetic image method mentioned above. Since the whole image is composed by different image squares randomly. The boundaries of the different images should be regarded as edges and there is absolutely no relation between the pixels belonged to different image squares. However, when we restore the pixels on the boundaries, we counted the contributions from pixels in different image squares, and this is wrong. This causes the blur of the boundaries and can extend to the whole image after iterations. To solve this boundary problem, we can either ignore the restoration of the pixels which are close to the boundaries or set small value to the size of the regions for calling different CLRS algorithms. This problem does not exist in the real-world images because a real-world image usually has different correlation coefficients distributed on the image and the pixels in the neighborhood has somehow relations to each other. The restoration results from a real-world image can be seen in the following figures.

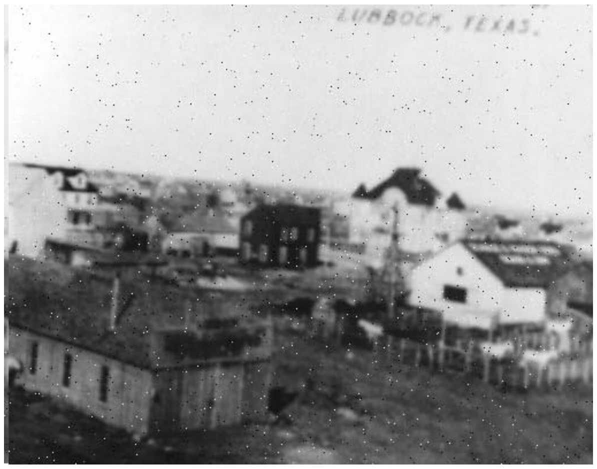

Figure 6. Noisy Lubbock Image

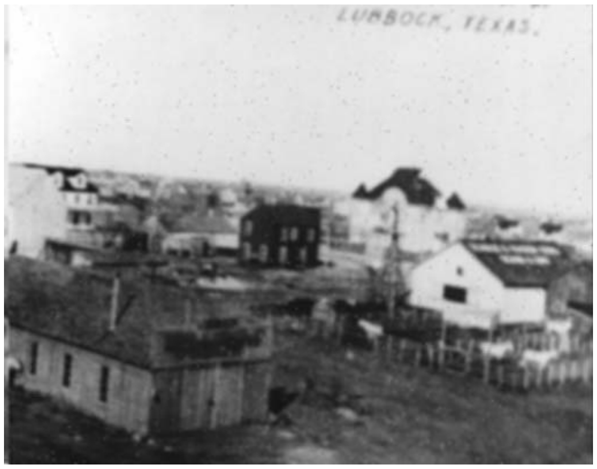

Figure 7. Restored by Nearest Four-neighbor Program

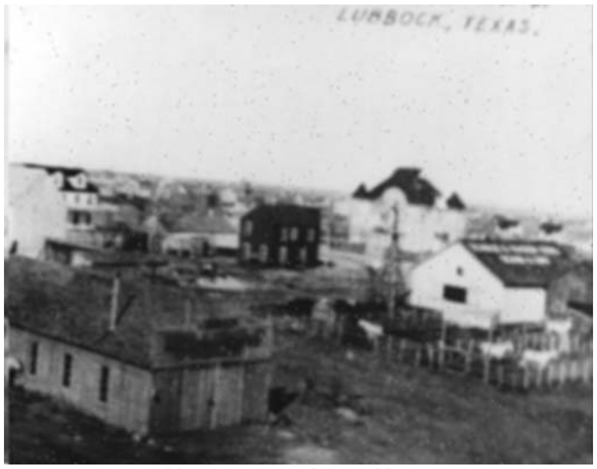

Figure 8. Restored by Nearest Eight-neighbor Program

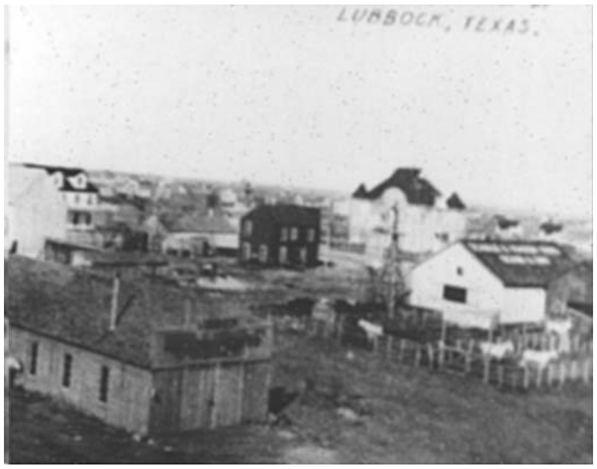

Figure 9. Restored by the Adaptive Method

Figure 6. is a noisy image of Lubbock. It has all the needed correlation coefficients mentioned in Table 1. except for the third one, which needs to be restored by the nearest eight vertical and horizontal neighbors. Figure 7. and Figure 8. show the restoration results by using the nearest four and eight-neighbor CLRS algorithms respectively. The noises are averaged out better by using the eight-neighbor program because of the contributions from more neighbors. On the other hand, the details are also blurred by the eight-neighbor program. Figure 9., which uses the adaptive method, shows better noise reduction result on the background and the sky because of the high correlation coefficient there. It also shows clearer details

6. CONCLUSIONS

Different neighborhood size can improve or worsen the restoration results. The autocorrelation coefficient is an important factor to determine the proper neighborhood size for image restoration. Based on this assumption, we developed an adaptive method, which uses the CLRS algorithm, to determine the correlation coefficients for different regions on the same image so that we could maximize the restoration results by using the proper neighborhood size. The adaptive method shows better restoration results than the common used nearest four and eight-neighbor methods. This conclusion is based on experimenting on both synthetic and real-world images.

Theoretically, this adaptive method can be extended to all the image restoration methods, which are based on MRF and Baysian’s framework.

REFERENCES

[1] Chengcheng Li “Image Restoration and Segmentation in a Larger Neighborhood System.”, Master’s thesis, Texas Tech University, 2002.
[2] Vir V.Phoha and William J.B.Oldham, “ Image Recovery and Segmentation Using Competitive Learning in a Layered Network.” IEEE Trans. on Neural Networks, Vol.7 No.4, 1996.
[3] Geman S. and Geman D., “ Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images”, IEEE Tans. On PAMI, 1984.
[4] D.Geiger and F.Girosi, “ Parallel and Deterministic Algorithms from MRFs: Surface Reconstruction.” IEEE Trans. on PAMI, Vol.13, No.5, May 1991.
[5] Pan, Victor Y., “Structured Matrices and Polynomials Unified Superfast Algorithms,” Springer, New York, pp 149-152, 2001
[6] Schott, James R., Matrix Analysis for Statistics, Wiley Series In Probability And Statistics, pp 304-305, 1997