New methods for digital halftoning and inverse halftoning

Murat Mese, Palghat P. Vaidyanathan

Murat Mese, Palghat P. Vaidyanathan, "New methods for digital halftoning and inverse halftoning," Proc. SPIE 4663, Color Imaging: Device-Independent Color, Color Hardcopy, and Applications VII, (28 December 2001); doi: 10.1117/12.452998

SPIE.

Event: Electronic Imaging, 2002, San Jose, California, United States

New Methods for Digital Halftoning and Inverse Halftoning

Murat Meşe and P. P. Vaidyanathan

California Institute of Technology, Pasadena, CA, USA

ABSTRACT

Halftoning is the rendition of continuous-tone pictures on bi-level displays. Here we first review some of the halftoning algorithms which have a direct bearing on our paper and then describe some of the more recent advances in the field. Dot diffusion halftoning has the advantage of pixel-level parallelism, unlike the popular error diffusion halftoning method. We first review the dot diffusion algorithm and describe a recent method to improve its image quality by taking advantage of the Human Visual System function. Then we discuss the inverse halftoning problem: The reconstruction of a continuous tone image from its halftone. We briefly review the methods for inverse halftoning, and discuss the advantages of a recent algorithm, namely, the Look Up Table (LUT) Method. This method is extremely fast and achieves image quality comparable to that of the best known methods. It can be applied to any halftoning scheme. We then introduce LUT based halftoning and tree-structured LUT (TLUT) halftoning. We demonstrate how halftone image quality in between that of error diffusion and Direct Binary Search (DBS) can be achieved depending on the size of tree structure in TLUT algorithm while keeping the complexity of the algorithm much lower than that of DBS.*

Keywords: Halftoning, dot diffusion, look up table, LUT, inverse halftoning, tree structure.

1. INTRODUCTION

Halftoning is the rendition of continuous-tone pictures on media on which only two levels can be displayed. Several algorithms have evolved for digital halftoning in the last decades. These include ordered dither, error diffusion, blue noise masks, green noise halftoning, direct binary search (DBS), and dot diffusion. In Section 2 we review some of the algorithms which have a direct relevance to our paper. We then define the Human Visual System (HVS) function used in evaluating the quality of halftones in Section 3. We use this HVS model to optimize the performance of a specific halftoning algorithm, namely dot diffusion.

The dot diffusion method for digital halftoning has the advantage of pixel-level parallelism unlike the popular error diffusion halftoning method. However, image quality offered by error diffusion is still regarded as superior to most of the other known methods. In Section 4 we show how the dot diffusion method can be improved by optimization of the so-called class matrix. By taking the human visual characteristics into account, we show that such optimization consistently results in images comparable to error diffusion, without sacrificing the pixel-level parallelism.

Another interesting problem addressed here is the inverse halftoning of images. Inverse halftoning is the reconstruction of a continuous tone image from its halftoned version. Inverse halftoning has a wide range of applications. Examples include image compression, printed image processing, scaling, enhancement, etc. In these applications, operations cannot be done on the halftone image directly, and inverse halftoning is mandatory. We briefly review the inverse halftoning algorithms in Section 5. In Section 6 we discuss a recent inverse halftoning algorithm, LUT (Look Up Table) based method for inverse halftoning of images. For each pixel, the algorithm looks at pixel's neighborhood (template) and depending upon distribution of pixels in template, it assigns a contone value from a precomputed LUT. The method is extremely fast (no filtering is

Further author information: (Send correspondence to Murat Meşe)

Murat Meşe.: E-mail: mese@systems.caltech.edu, Telephone: (310) 820 1903x28, Address: Sequoia Communications, 1990 S. Bundy Dr. # 395, Los Angeles, CA, 90025.

P.P. Vaidyanathan: E-mail: ppvnath@systems.caltech.edu, Telephone: (626) 395 4681, Address: Department of Electrical Engineering 136-93, California Institute of Technology, Pasadena, CA 91125 USA, Fax: (626) 795-8649

*This work was supported by National Science Foundation under Grant MIP 0703755.

Color Imaging: Device-Independent Color, Color Hardcopy, and Applications VII, Reiner Eschbach, Gabriel G. Marcu, Editors, Proceedings of SPIE Vol. 4663 (2002) © 2002 SPIE. · 0277-786X/02/$15.00

required) and the image quality achieved is comparable to the best methods known for inverse halftoning. The LUT inverse halftoning method does not depend on the specific properties of the halftoning method, and can be applied to any halftoning method. An algorithm for template selection for LUT inverse halftoning is also discussed. We demonstrate the performance of the LUT inverse halftoning algorithm on error diffused images.

Many of the entries in the LUT are unused because the corresponding binary patterns hardly occur in commonly encountered halftones. These are called nonexistent patterns. In Section 7, we discuss a tree structure which will reduce the storage requirements of an LUT by avoiding nonexistent patterns. First a small template LUT will be used to get a crude inverse halftone. Then this value will be refined by adaptively adding pixels to the template depending on context.

Later, we apply LUT and TLUT ideas on halftoning and we discuss LUT (Look Up Table) based halftoning and tree-structured LUT (TLUT) halftoning. Pixels from a causal neighborhood (template) and contone value of the current pixel will be included in LUT. The details of this algorithm are given in Section 9. We will demonstrate how error diffusion characteristics can be achieved with this method. The performance of LUT halftoning will be improved by TLUT halftoning. This is achieved by making the templates adaptive. Even though the TLUT method is more complex than LUT halftoning, it produces better halftones and requires much less storage than LUT halftoning. Afterwards, our algorithm is trained on halftones obtained by DBS. The complexity of TLUT halftoning is higher than that of error diffusion algorithm but much lower than that of the DBS algorithm. Also, the quality of TLUT halftones increases if the size of TLUT gets bigger. Thus, different halftone image qualities between that of error diffusion and DBS can be achieved depending on the size of tree structure in the TLUT algorithm.

2. DIGITAL HALFTONING

There are different kinds of halftoning algorithms. Let us denote the continuous tone (contone) image as $x(\mathbf{n})$ and the halftone of it as $h(\mathbf{n})$ where $\mathbf{n} = (n_1, n_2)^T$. We describe some of the halftoning algorithms below.

2.1. Ordered Dither

In this halftoning algorithm, the input image is compared with a periodic screen image. This screen image is also called the threshold matrix or the screen matrix. Basically, the screen matrix defines the order in which the dots are added to the lattice as the brightness is decreased, and it also determines the halftone image quality.

Depending on the screen matrix, the halftoning algorithm has different characteristics. The simplest screen matrix is the one which has constant value at each pixel. Most of the details which exist in the contone images are lost due to halftoning process if ordered dithering with this screen matrix is used and the contone images are barely visible in the corresponding halftone images.

In clustered ordered dither halftoning, the screen matrix is designed to mimic the analog halftoning process. When the pixel intensity decreases in the contone image, a dot will grow around the pixel. Examples of clustered dither screen matrices can be found in [1].

Another class of ordered dither halftoning algorithms is called dispersed ordered dithering. A design rule is established by Bayer [2]. In his work, the visibility of unwanted artificial texture is measured in terms of a Fourier analysis of the dot patterns at different brightness levels. When the dot pattern for a uniform patch has components at different wavelengths, components with the longest finite wavelength are assumed to be the most visible. With this criteria, Bayer has designed optimum screen matrices. With this screen matrix, there are more details visible in the halftone images than the halftones obtained with clustered dither halftoning.

Even though the dispersed ordered dithering renders more details than clustered ordered dithering, in practice clustered ordered dithering is also used because of the so-called "dot gain". The dot gain is due to non-ideal behavior of printers. Even though ideal printers are supposed to be able to make dots which have predefined geometrical shapes such as a square, in reality there is a dot gain because the ink spreads from the predefined geometry into the neighboring pixels. This results in "dot gain". The clustered ordered dithering is more robust against dot gain because, when the pixel intensity decreases in the contone image, a dot will grow around the pixel, and this decreases the dot gain effects in the halftone image.

2.2. Error Diffusion

Another popular halftoning algorithm is error diffusion. This algorithm was introduced by Floyd and Steinberg [3]. Even though the algorithm requires neighborhood processes, it gives better halftone quality for printers which do not suffer from dot gain. In this method, the pixels are processed in raster scan order. The errors created due to quantization are 'diffused' with an error filter to the pixels which are not processed. A popular error filter was introduced by Floyd and Steinberg [3]. This filter shapes the halftone error such that it cannot be perceived when viewed at a distance. Basically, power spectrum of the halftone error will be concentrated in high frequencies, and this type of error is called *blue noise* [1]. Other error diffusion filters are also used [1]. These filter coefficients are optimized using human visual system characteristics [4][5]. The halftone quality offered by this method is superior to ordered dithering methods.

2.3. Dot diffusion

The dot diffusion method for halftoning introduced by Knuth [6] is an attractive method which attempts to retain the good features of error diffusion while offering substantial parallelism. The dot diffusion method for halftoning has only one design parameter, called **class matrix C**. It determines the order in which the pixels are halftoned. Thus, the pixel positions (n_1, n_2) of an image are divided into IJ classes according to (n_1 mod I, n_2 mod J) where I and J are constant integers. Table 1 is an example of the class matrix for $I = J = 8$. There are 64 class numbers. Let $x(n_1, n_2)$ be the continuous tone (contone) image with pixel values in the normalized range $[0, 1]$. Starting from class $k = 1$, we process the pixels for increasing values of k. For a fixed k, we take all pixel locations (n_1, n_2) belonging to class k and define the halftone pixels to be

$$h(n_1, n_2) = \begin{cases} 1 & \text{if } x(n_1, n_2) \geq 0.5 \\ 0 & \text{if } x(n_1, n_2) < 0.5 \end{cases} . \tag{1}$$

We also define the error $e(n_1, n_2) = x(n_1, n_2) - h(n_1, n_2)$. We then look at the eight neighbors of (n_1, n_2) and replace the contone pixel with an adjusted version for those neighbors which have a higher class number (i.e., those neighbors that have not been halftoned yet). To be specific, neighbors with higher class numbers are replaced with

$$x(i, j) + 2e(n_1, n_2)/w \quad \text{(for orthogonal neighbors)} \tag{2.2(a)}$$

$$x(i, j) + e(n_1, n_2)/w \quad \text{(for diagonal neighbors)} \tag{2.2(b)}$$

where w is such that the sum of errors added to all the neighbors is exactly $e(n_1, n_2)$. The extra factor of two for orthogonal neighbors (i.e., vertically and horizontally adjacent neighbors) is because vertically or horizontally oriented error patterns are more perceptible than diagonal patterns.

The contone pixels $x(n_1, n_2)$ which have the next class number $k + 1$ are then similarly processed. The pixel values $x(n_1, n_2)$ are of course not the original contone values, but the adjusted values according to earlier diffusion steps (2.2). When the algorithm terminates, the signal $h(n_1, n_2)$ is the desired halftone.

Usually an image is enhanced [6] before dot diffusion is applied. For this the continuous image pixels $C(i, j)$ are replaced by $C'(i, j) = \frac{C(i,j) - \alpha \bar{C}(i,j)}{1-\alpha}$ where $\bar{C}(i, j) = \frac{\sum_{u=i-1}^{i+1} \sum_{v=j-1}^{j+1} C(u,v)}{9}$. Here the parameter α determines the degree of enhancement. If $\alpha = 0$, there is no enhancement, and the enhancement increases as α increases. If $\alpha = 0.9$ then the enhancement filter simplifies to $C'(i, j) = 8C(i, j) + C(i, j) - \sum_{0<(u-i)^2+(v-j)^2<3} C(u, v)$. This algorithm is completely parallel requiring 9 additions per pixel, and no multiplications.

Fig. 1 shows the 512×512 continuous tone peppers image halftoned by using Knuth's class matrix. Notice that there are artificial periodic patterns in Fig. 1. Surprisingly, not much work has been done on optimization of the class matrix. We will show how this matrix can be optimized in Section 4 to obtain a halftone quality close to error diffusion. More details about dot diffusion can be found in [7] and [8]. These papers discuss the mathematical description of dot diffusion, embedded multiresolution dot diffusion, and the relation between error diffusion and dot diffusion algorithms.

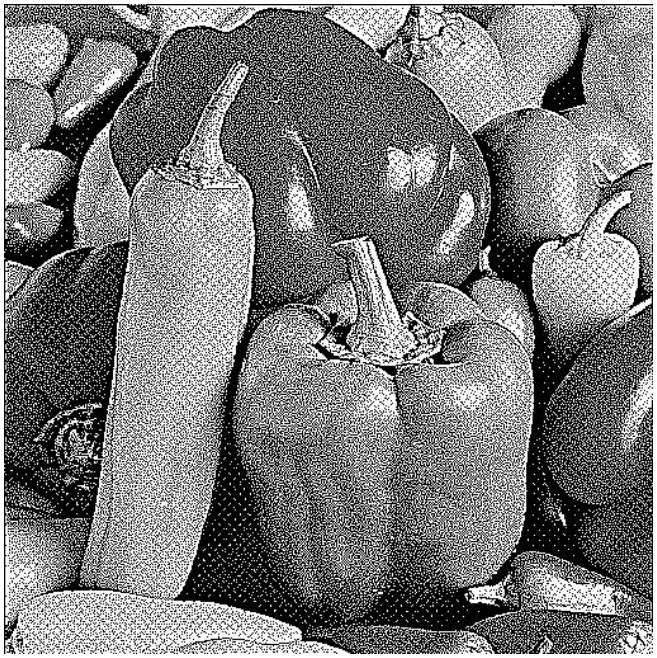

Figure 1: Dot diffusion with Knuth's class matrix.

2.4. Green Noise Halftoning

The concept of green noise and its advantages over blue noise for digital halftoning was introduced by Lau et.al. [9]. Green-noise dither patterns are composed of pixel-clusters making them less susceptible to image degradation from nonideal printing artifacts such as dot gain. Error diffusion with output dependent feedback and variations based on filter weight perturbation are shown to be good generators of green noise.

2.5. Direct Binary Search

DBS tries to find the best halftone image for a contone image. The best halftone here is the one which, when examined from a distance it matches the contone image the most. In Section 3 we define the human visual system function, and the best halftone is the one which has the least perceived halftoning error. DBS starts with an initial halftone image and uses a greedy approach: It checks whether toggling the pixel value or swapping the values of pixels nearby reduces the perceived halftoning error for each pixel in the halftone image and for each iteration. Even though DBS gives the best halftone quality, it is computationally expensive, and serves as a benchmark. More details on DBS algorithm can be found in [10].

2.6. Blue Noise Masks

The blue noise mask is another way to get halftone quality similar to error diffusion [11]. This method is the same as ordered dithering except that the screen matrices are huge. The usual size for a blue noise mask are 128×128 and 256×256. These screen matrices are designed so that they render the gray levels as best as possible. The disadvantage of blue noise masks is that the resulting halftones do not have enhancement like the error diffusion inherently has.

3. HUMAN VISUAL SYSTEM (HVS) FUNCTION

When we see a picture from a distance, we do not perceive the exact picture. Basically our eye applies a low-pass filter to the images and this Human Visual System (HVS) function has been experimentally measured. Here we show an HVS function used in halftone image quality assessment. Assume that we have the original contone image and different halftones of the same contone image. We want to know what is the perceived halftoning error (PHE) for each of these images. When we perceive images, they are passed through a model of the HVS function. Thus, the PHE of a halftone image with respect to a contone image is the difference

between the perceived halftone image and perceived contone image. Since this model is linear, we apply the HVS function to the difference between the original and halftone images. The energy of the resulting image is defined to be the PHE. The calculation of PHE of a given image $x[m,n]$ for a given HVS function $h[m,n]$ is as follows: Let us denote the specific halftone of $x[m,n]$ as $d[m,n]$. The perceived error image will be $e[m,n] = (h*d)[m,n] - (h*x)[m,n]$. Then the energy of the perceived image is defined to be the PHE of image $x[m,n]$: $PHE = \Sigma_{m=-\infty}^{\infty}\Sigma_{n=-\infty}^{\infty}|e[m,n]|^2$. The image used in the optimization should be chosen wisely. For example in this paper we have chosen a gray scale ramp because the cost of a gray scale ramp is the average value of the costs of gray scales which exist in the gray scale ramp.

We will use a specific HVS model in this paper. In the frequency domain the HVS model is defined as follows: $H_c(u,v) = aL^b exp(-\frac{1}{s(\phi)}\frac{\sqrt{u^2+v^2}}{clog(L)+d})$. Here, u and v are the frequency variables in cycles/degree subtended at the retina and L is the average luminance in $candela/m^2$. The quantity $\sqrt{u^2+v^2}$ is therefore the radial frequency. The quantity ϕ is the angular frequency defined as $\phi = atan(\frac{u}{v})$. The various constants and the function $s(\phi)$ are defined as follows: $a = 131.6$, $b = 0.3188$, $c = 0.525$, $d = 3.91$, $s(\phi) = \frac{1-w}{2}cos(4\phi) + \frac{1+w}{2}$ where $w = 0.7$. The dependence on radial frequency $\sqrt{u^2+v^2}$ was developed in [12]. Then the angular dependence of the model, i.e., $s(\phi)$ was introduced in [13] following Daly's model [14]. We used $L = 10$ $candela/m^2$ in our experiments. With $h_c(x,y)$ denoting the inverse Fourier transform of $H_c(u,v)$, the discretized version $h[m,n] = h_c(Tm,Tn)$ is used in the calculations. The relation between $H(w_1,w_2)$ (the discrete fourier transform of $h[m,n]$) and $H_c(u,v)$ is as follows: $H(w_1,w_2) = (1/T)\Sigma_{k=-\infty}^{\infty}\Sigma_{l=-\infty}^{\infty}H_c(w_1 - 2\pi k/(2\pi T), w_2 - 2\pi l/(2\pi T))$. Sampling the inverse transform at interval $T = 0.0165$ corresponds to a certain printer resolution, R dpi, viewed at a specific distance, D inches. Since a length x viewed at a distance D subtends an angle of $\Theta = tan^{-1}(x/D) \approx x/D$ radians for $x \ll D$, the spacing of the dots will be: $T = 1/(RD)$radians $= (180/\pi)/(RD)$degrees. This clarifies the relation between T, D and R. In particular, $T = 0.0165$ corresponds to $RD = 300\text{dpi} \times 11.5827\text{in}$.

4. OPTIMIZATION OF DOT DIFFUSION ALGORITHM

Knuth introduced the notion of **barons** and **near-barons** in the selection of his class matrix. A baron has only low-class neighbors, and a near-baron has one high class neighbor. The quantization error at a baron cannot be distributed to neighbors, and the error at a near-baron can be distributed to only one neighbor. Knuth's idea was that the number of barons and near-barons should therefore be minimized. He exhibited a class matrix with two barons and two near-barons. The quality of the resulting halftones still exhibits periodic patterns similar to ordered dither methods (see Fig. 1). Knuth has also produced a class matrix with one baron and near-baron, but unfortunately these were vertically lined up to produce objectionable visual artifacts. In our experience, the baron/near-baron criterion does not appear to be the right choice for optimization. To explain this, define a k-baron to be a position which has k high-level neighbors. Thus $k = 0$ corresponds to a baron, $k = 1$ to a near baron, and $k = 8$ to an **antibaron.** We have produced a class matrix which minimizes the number of k-barons sequentially for $0 \le k \le 8$. The resulting halftone quality was found in most cases to be slightly worse than Knuth's original results, leading us to conclude that baron minimization is not the right approach. Therefore, we introduce a different optimization criterion based on the HVS, and show that the image quality is significantly improved, though the class matrix does not minimize barons.

In the optimization of dot diffusion algorithm, we are looking for a class matrix which minimizes the cost function because class matrix is the only design parameter of the algorithm. Notice that the optimization is equivalent to finding a combination of numbers from 1 to IJ such that the related cost is minimized. Here the cost function used is based on HVS model and it is the PHE defined in Section 3 with a specific image: For example, we have chosen a gray scale ramp because the cost of a gray scale ramp is the average value of the costs of gray scales which exist in the gray scale ramp. Since the cost function depends nonlinearly on its parameters, we will use an optimization procedure to get the desired class matrix. The choice of the class matrix that minimizes this cost function was performed using the **pairwise exchange algorithm** [15],[7].

4.1. Optimization Results

First we have done the optimization for 8×8 class matrix. We have optimized this class matrix for $RD = 300\text{dpi} \times 11.5827\text{in}$ [8]. This class matrix is shown in Table 1. We have summarized the perceived halftoning

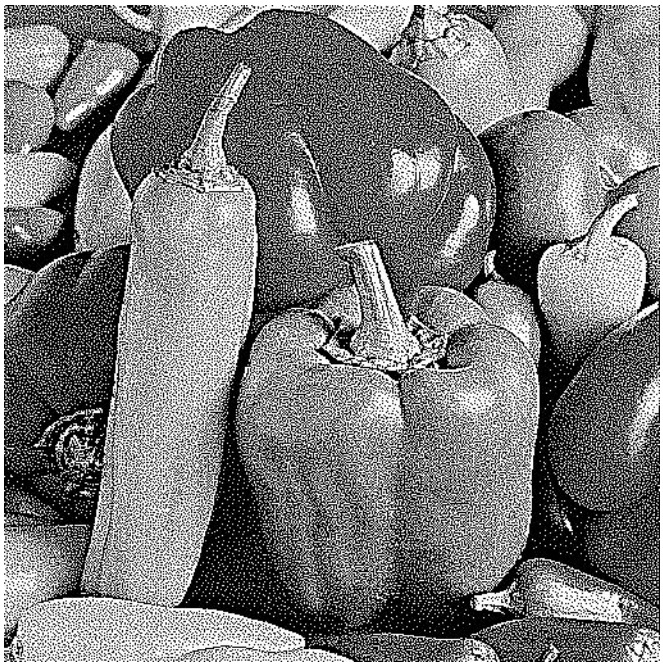

Figure 2: Dot Diffusion with HVS optimized 8×8 class matrix and enhancement.

Table 1: 8×8 optimized class matrix.

37	41	34	14	60	61	7	9
16	12	36	59	46	17	50	24
45	27	33	58	5	3	42	48
29	2	57	30	43	15	20	11
26	18	55	49	4	32	10	54
25	21	53	40	38	6	64	52
8	28	35	13	39	22	63	56
51	44	19	23	31	62	1	47

errors in Table 2. In this paper, perceived errors are normalized so that perceived error of a gray scale ramp halftoned by the Floyd Steinberg error diffusion algorithm is unity. Our optimized class matrix achieves 40.04% less PHE than Knuth's class matrix and 51.61% more PHE than error diffusion.

Example: The 512×512 continuous tone peppers image was halftoned by using Knuth's class matrix (Fig. 1, $PHE = 30.77$), and by the optimized 8×8 class matrix (Fig. 2, $PHE = 30.35$). [†] The images in this paper are printed out with $R = 150$ *dpi*, thus they should be viewed from a distance $D \approx 23$ *inches*. It is clear that the dot diffusion method with the optimized 8×8 class matrix is visually superior to dot diffusion method with Knuth's class matrix. In fact, dot diffusion with the optimized 8×8 class matrix offers a quality comparable to Floyd-Steinberg error diffusion method ($PHE = 3.86$). Note that, we cannot use the PHE values of error diffused images and images obtained by dot diffusion with enhancement to compare the visual quality of these two methods because of the enhancement step. Thus visual inspection is necessary. Error diffused images suffer from worm-like patterns which are not in the original image, whereas dot diffused halftones do not contain these artifacts. Notice that the artificial periodic patterns in Fig. 1 are absent in error diffused halftones and in the dot diffusion with the optimized 8×8 class matrix (Fig. 2).

4.2. Dot Diffusion without Enhancement

If the halftone images obtained with enhancement (Fig. 2,$PHE = 30.35$) and without enhancement ($PHE = 6.90$) are compared, it can be concluded that the enhancement step reduces the periodic structures in the

[†]We observed that the enhancement step in dot diffusion is the cause of higher PHE values. In the next section, enhancement step will be removed from dot diffusion algorithm.

Table 2: Perceived halftoning errors (PHE) for 8×8 class matrices.

Halftoning Method	Dot Diffusion Knuth's Class Matrix	Dot Diffusion Optimized Class Matrix	Error Diffusion (Floyd Steinberg)
Perceived Halftoning Error	2.53	1.52	1.00

halftone image, but it might be objectionable in some applications because of its very visible sharpening effect (e.g., see Fig. 2). It turns out that we can get good halftones without use of the enhancement step provided we make the class matrix larger than the standard 8×8 size. The price paid for the larger class matrix is that the parallelism of the algorithm is compromised. However, in practice, even with a 16×16 class matrix, we have plenty of parallelism for any desktop printing implementation: Assume that we want to render the image at 600 dpi, and process 16 rasters (lines or rows) of an 8.5×11 inch square page simultaneously. Then we will have (600*8*16)/256=300 pixels that can be processed simultaneously.[‡] The real disadvantage of increasing the size of the class matrix is the fact that the number of rasters that must be processed at the same time increases.

We found that if a 16×16 matrix is used, the halftone images resulting from the optimization of this matrix are very good even without the enhancement step. (For comparison we note here that whenever enhancement is used, the class matrix can be as small as 5×5 without creating noticeable periodicity patterns.) Such optimization was carried out using a gray scale ramp as the training image. The HVS function was used in the optimization, and the associated cost was optimized using the pairwise exchange algorithm. The PHE of a gray scale halftoned by dot diffusion with 16×16 optimized class matrix is 1.19. This class matrix can be found in [8]. We can compare this PHE value with the PHE values in Table 2. The PHE for optimized 16×16 class matrix is only 19.38% worse than the error diffusion. Also, the PHE for optimized 8×8 class matrix is 27.00% worse than the PHE for optimized 16×16 class matrix.

The peppers image halftoned with the resulting class matrix is shown in Fig. 3 ($PHE = 5.90$). There are no periodic artifacts in this result. While the overall visible noise level appears to be higher than for error diffusion, the problematic halftone patterns of error diffusion in the mid gray level are eliminated here. (Examine the body of the middle pepper in error diffused peppers image in [7]). By comparing Fig. 1 and 3 we see that 16×16 dot diffusion without enhancement is also superior to 8×8 enhanced dot diffusion using Knuth's matrix because there are no noticeable periodic patterns any more, and there are no enhancement artifacts.

5. INVERSE HALFTONING PROBLEM

Inverse halftoning is the reconstruction of a continuous tone image from its halftone. Since there can be more than one continuous tone image giving rise to a particular halftone image, there is no unique inverse halftone of a given halftoned image. Thus, extra properties of images are needed in order to do inverse halftoning. The basic assumption in all inverse halftoning algorithms is that "natural" images have "mostly lowpass" characteristics. Simple low pass filtering can remove most of the noise due to halftoning but it also removes edge information. Besides lowpass filtering, there are more sophisticated approaches for inverse halftoning. The method of projection onto convex sets (POCS) has been used by Analoui and Allebach [16] for halftone images produced by ordered dithering. For error diffused halftones, Hein and Zakhor [17] have successfully used the POCS approach. A different method called logical filtering has been used by Fan [18] for ordered dither images. Wong has used an iterative filtering method for inverse halftoning of error diffused images [19]. The method of overcomplete wavelet expansions has been used in [20] to produce inverse halftones with good quality for error diffused images by separating the halftoning noise from the original image through edge detection. Another method for inverse halftoning of error diffused images was introduced by Kite et al. in [21]. This method is not only fast but also yields images of very good quality. The method uses space varying filtering based on gradients obtained from the image. For inverse halftoning of dot diffused images two methods were discussed by Meşe and Vaidyanathan [7]. One of the methods uses POCS which is an iterative algorithm. The other one is based on wavelet decomposition of images to differentiate the halftoning noise from the original image.

[‡]The number 8 in the numerator of this expression is based on the assumption of an 8 inch wide active printing area.

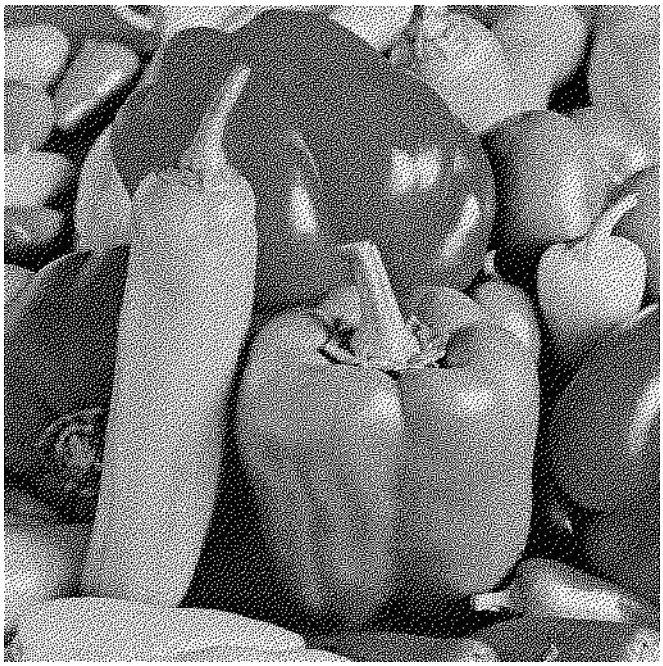

Figure 3: Dot Diffusion with HVS optimized 16x16 class matrix and no enhancement.

In Section 6 we propose LUT (Look Up Table) based methods for inverse halftoning of images. The LUT for inverse halftoning is obtained from the histogram gathered from a few sample halftone images and corresponding original images. For each pixel, the algorithm looks at the pixel's neighborhood (template) and depending upon distribution of pixels in template, it assigns a contone value from a precomputed LUT. The method is extremely fast (no filtering is required) and the image quality achieved is comparable to the best methods known for inverse halftoning. The LUT inverse halftoning method does not depend on the specific properties of the halftoning method, and can be applied to any halftoning method. We demonstrate the performance of the LUT inverse halftoning algorithm on error diffused images. We also extend LUT inverse halftoning to color halftones.

Many of the entries in the LUT are unused because the corresponding binary patterns hardly occur in commonly encountered halftones. These are called nonexistent patterns. In Section 7, we propose a tree structure which will reduce the storage requirements of an LUT by avoiding nonexistent patterns. First a small template LUT will be used to get a crude inverse halftone. Then this value will be refined by adaptively adding pixels to the template depending on context. We will demonstrate its performance on error diffused images.

6. LOOK UP TABLE (LUT) METHOD FOR INVERSE HALFTONING

In this section we discuss Look Up Table (LUT) methods for inverse halftoning of images [22]. The LUT method was first used by Netravali et al. to display dithered images [23]. However, in that work the authors assumed that the dither matrix and the registration of halftoned image with the dither matrix are known. Notice that this information may not be known for a particular halftone image. In another paper, Ting and Riskin used LUT method in halftone compression to get a temporary contone image [24]. However, obtaining high quality contone image was not the aim in that work.

Our LUT inverse halftoning method does not involve any linear filtering at all, and it does not explicitly assume any image model. To determine the inverse halftone value at a point, the algorithm looks at the pixel's neighborhood and, depending upon the distribution of pixels in the neighborhood, it assigns a contone value from a precomputed LUT. The number of pixels used in the neighborhood is relatively small, typically 16. For a 16 pixel neighborhood, we need $2^{16} = 64K$ bytes of LUT, and for a 19 pixel neighborhood, we need $2^{19} = 512K$ bytes of LUT. The method is completely parallel and it does not require any computation because it is an LUT based algorithm. This makes it faster than the previously known methods. Moreover, the method provides very good image quality, often even better than the method in [21] as we shall demonstrate.

Table 3. Neighborhood used in inverse halftoning (rectangular support, "Rect" template).

a	a	a	a
a	a	a	a
a	a	O	a
a	a	a	a

Table 4. Another possible neighborhood used in inverse halftoning ("19pels" template).

b	a	a	a	b
a	a	a	a	b
a	a	O	a	a
	a	a	a	
		a		

Since inverse halftoning cannot be done without using extra properties of images, we use sample images for training the LUT. In the training phase, sample images and corresponding halftone versions are used to obtain the LUT. The training phase is simple, and should be done once for a specific halftone method. Later in this section the details of the algorithm are given. Then we show application of LUT inverse halftoning to actual images. Note that our LUT inverse halftoning algorithm accepts binary images only and extension to scanned halftones remains to be explored.

The first step in LUT inverse halftoning is to choose the neighborhood or template (a collection of pixels) for the pixels to be used in the prediction. We have used different templates as shown in Tables 3 and 4. In Table 3, a rectangular template is shown, and this template is abbreviated as "**Rect**" in comparison tables later. "**O**" denotes the estimated pixel. All of the pixels are used in the prediction. In Table 4 another template is shown. This pattern consists of a symmetric part and some additional pixels. In the 16 pixel pattern the "**a**" pixels and the "O" pixel are used in the prediction of the contone value of pixel "O." This template is denoted as "**16pels**" in comparison tables. Another example of a template is obtained by adding "**b**" pixels to the "**16pels**" template, and this template is referred to as "**19pels**" template.

Let us assume that there are N pixels (including the pixel being estimated) in the neighborhood and that they are ordered in a specific way. Let us also call the pixel values as p_0,p_1,...,p_{N-1}. Note that there are 2^N different patterns since $p_i \in \{0, 1\}$ for $i = 0, 1, ..., N-1$. Assuming that the original image is an 8 bit image, our LUT, T, should return a value for each pattern, i.e., $T(p_0, p_1, ..., p_{N-1}) \in \{0, 1, ..., 255\}$. During the inverse halftoning phase, the pixels in the region of support will be arranged in a specific order, and the contone value will be obtained from the LUT.

In the design of LUT we obtain the expected contone value for each template pattern. Then this contone value will be assigned to the corresponding LUT position for that pattern. Details on the design of LUT can be found in [22]. The contone values for nonexistent patterns should be estimated from the existent patterns. The "best linear estimator" was proposed for nonexistent pattern estimation [22]. In this method, a linear model is used to estimate the contone values in terms of halftone values inside the template.

6.1. Experimental results on LUT inverse halftoning

In order to illustrate the performance of our method, we will use Lena and mandrill images in our experiments. For visual comparison we now show the inverse halftone images for mandrill in Fig. 4 and Fig. 5. Figure 4 is obtained by using a recent method [21] called the "fastiht2" method (perhaps the best known method which achieves high PSNR in inverse halftoning or error diffused images and which is also quite fast). Figure 5 is obtained with our new LUT method. It is clear that the LUT method performs very well compared to "fastiht2'. For example, it preserves high frequency information with nearly no loss (observe the hair on the cheeks). In this example the LUT method uses a training set containing 30 images which can be found at [25] and the template used was the "Rect" support. The training set does not include Lena and mandrill images. In Table 5, we have shown the average PSNR values for LUT inverse halftone images using different templates and inverse halftones obtained by "fastiht2". LUT inverse halftoning achieves better average PSNR values even for 16 pixels template. Note that the training is done only once for a specific halftoning algorithm. (Training is only required once for a specific halftoning method and training time should not be confused with inverse halftoning time of an image.) Similar comparison for the case of Lena confirms that the LUT method performs as well as the "fastiht2' method. The performance of our algorithm on clustered dot and dispersed dot ordered dither images and the details on how the inverse halftone quality is affected by various factors such as the handling of nonexistent patterns, the content of the training set, and template selection can be found in [22].

Table 5: Comparison of different templates. Halftones are obtained by error diffusion.

Method	LUT Method			fastiht2
Template	16pels	Rect	19pels	(Kite et.al.)
Average PSNR	26.43	26.50	26.61	25.95
Storage	64KB	64KB	512KB	-

Besides how to design the LUT for a given template, another question is how we can choose the best template for inverse halftoning. In [22] a recursive algorithm to choose the template is given. Starting from an empty template, inverse halftone images using the pixels in the previous template and one more candidate pixel are obtained. Thus if a smaller template of size K is needed, the first K pixels can be taken into the template.

Remark: If the images are mostly smooth images, the inverse halftone quality of error diffused halftones can be improved by applying a simple smoothing algorithm on the LUT inverse halftoned image. Here we apply a small median filter (a 3×3 separable median filter which requires only 6 comparisons per pixel) on the LUT inverse halftone result to get even smoother, high PSNR inverse halftone images. The Lena image LUT inverse halftoned with 'Rect' template and then 3×3 median filtered is can be found at [22](PSNR=31.50dB).

The LUT method can be extended for color halftones [22]. The simple extension of LUT inverse halftoning algorithm to color halftones is to treat the color planes separately. For each plane first choose the template and then design the LUT table for that template. Even though this gives satisfactory results we can do better for color inverse halftoning. In most of the "natural images" there is a high correlation between the color planes. This can be exploited in inverse halftoning. During the LUT creation phase, if we carefully include the pixels from different color planes to predict the contone value of a particular color plane, this means that we are exploiting the correlation between the color planes. The color inverse halftone images obtained with LUT method are visually pleasing [22].

7. TREE-STRUCTURED LUT (TLUT) INVERSE HALFTONING

The LUT sizes in LUT inverse halftoning can become bigger if high quality inverse halftone is required. Each binary combination of pixels in the template corresponds to one entry in the LUT. But many of these combinations hardly arise in practice, and are referred to as **nonexistent patterns**. For example, in error diffusion with a eighteen pixel neighborhood, we found that 22.75% of the patterns are nonexistent. For clustered dot ordered dither 78.22% are nonexistent, and for Bayer's dispersed dot ordered dithering 86.22%. Here we show how to take advantage of nonexistent patterns and reduce storage [26],[27]. By using a tree structure, the storage requirements can be a fraction of its LUT equivalent. In this sense, the tree structure can be regarded as a 'compressed' version of the LUT. First a small template LUT will be used to get a crude inverse halftone for each pixel. Then this value will be refined by adaptively adding pixels to the template depending on the context. These adaptive pixels will be placed in a tree structure. Like the LUT method, tree-structured LUT inverse halftoning does not involve any filtering, but there will be more steps in inverse halftoning.

We first show the type of tree structure we are using in inverse halftoning. Then we will describe our inverse halftoning with tree structure in Sec. 7.1. The experimental results on error diffused halftone images are reported and discussed in Section 7.2.

Tree structure: Let us denote the size of the initial template used as a (this is typically small, e.g., $a = 9$). We will define 2^a binary trees corresponding to the different patterns in the template. Each tree node is either split further or it is a leaf. Tree nodes are split so that the contone values of LUT obtained with initial template can be refined. If a node is split, then the location of additional pixel,(i, j), is stored in the node and two more nodes attached to this node as its children. If a tree node is a tree leaf, then a contone value is stored in the node [22].

7.1. Inverse halftoning with tree structure

In tree-structured LUT inverse halftoning, we try to find a tree leaf for each pixel in the halftone image. After finding the tree leaf, the contone value stored in the tree leaf will be assigned as the inverse halftone value of the pixel. To find the corresponding tree leaf for each halftone pixel location, we will do the following:

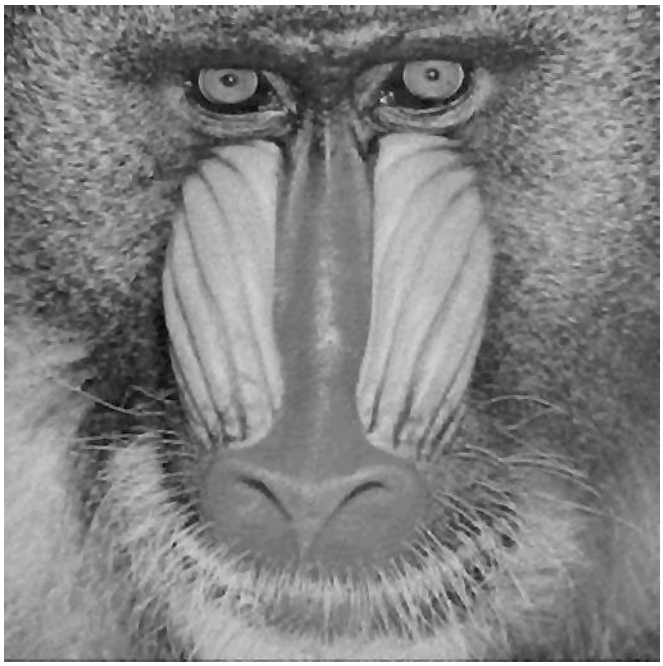

Figure 4: Inverse halftoning by fastiht2 (Kite et al.) (PSNR=22.59dB). The halftone was obtained by error diffusion.

1. First look at the pattern inside the initial template $\mathcal{B}$ of size a. Each pattern will correspond to one of the binary trees. The root of the corresponding tree is declared as the current node.
2.Each node is either split into two nodes or it is a leaf. If a node is a leaf, then the average contone value is stored in the node. This value is assigned as the inverse halftone value at the pixel.
3. If a node is split into two, then the location (i, j) of the additional pixel is stored in the node. Get the halftone value of the pixel which is (i, j) away from the current pixel. If this value is 0(1), then the left (right) node is assigned as the current node. Then go to step 2.

The design of tree structure for inverse halftoning is discussed in [26]. We also have to find the contone values for each tree leaf given a tree structure and additional pixel locations. Afterwards, these contone values will be assigned as inverse halftone values in the tree-structured LUT inverse halftoning algorithm. We will use training images in this process, i.e., halftone images and corresponding contone images. First we find the tree leaves for each pixel in the training set using the inverse halftoning algorithm. Let us denote the set of contone values of pixels which have the same tree leaf t as S_t where a_t is the size of S_t. Thus $S_t = \{x_1, x_2, \ldots, x_{a_t}\}$. Then the closest integer to the mean value of S_t will be assigned as the contone value of the leaf (c_t): $c_t = int(\sum_{i=1}^{a_t} x_i/a_t)$.

7.2. Experimental results for TLUT inverse halftoning

We will first apply the TLUT inverse halftoning algorithm on error diffused halftones to demonstrate the performance of our algorithm. Then, TLUT will be applied on clustered dot and dispersed dot ordered dither halftones. Note that, our inverse halftoning algorithm can be applied to any type of halftoning algorithm.

First we choose the initial template using the algorithm given in [22]. We have included 30 images in our training set and another 30 images in our test set. These sets contain both smooth and non-smooth images and these images can be found at [25]. Table 6 shows the template, called $TFS13$, for Floyd Steinberg error diffused images. In the table '0' denotes the current pixel, and the 'x' denotes the other pixels in the template. The biggest template is chosen such that there are no nonexistent patterns in the halftone training images.

In Table 7, we have summarized the performance of tree structure inverse halftoning. The trees used differ only in the number of tree leaves. The initial template for all of these tree structures is $TFS13$ and it has thirteen pixels. The average PSNR value of the inverse halftone images in the test set (with respect to the original contone images) are shown in this table. We also added the PSNR values of TLUT inverse halftoned Lena images. Note that Lena image is not in the training set.

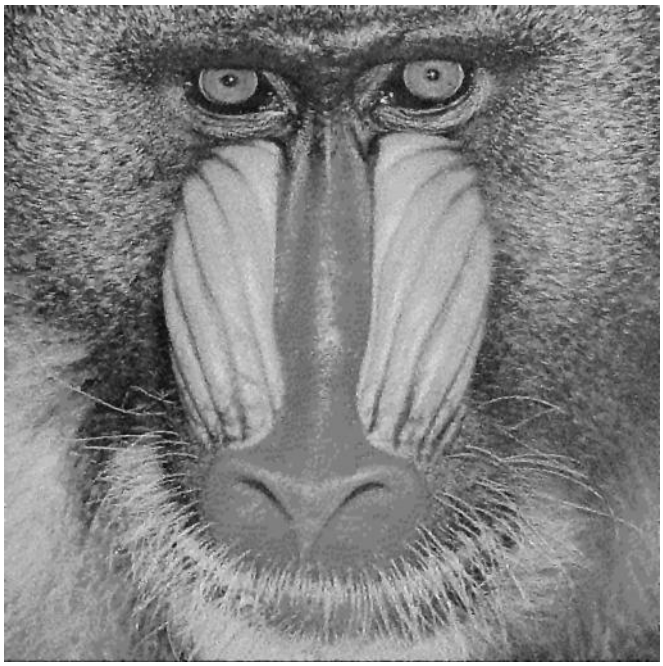

Figure 5. LUT inverse halftoning with "Rect" template (PSNR=24.42dB). The halftone was obtained by error diffusion.

Table 6. Initial template $TFS13$ used in tree structure LUT inverse halftoning of Floyd-Steinberg Error Diffused Images.

x	x	x	x	
x	x	0	x	x
x	x	x	x	

Table 7. Comparison of Different Tree Structures used in TLUT inverse halftoning.

Tree	$FS1$	$FS2$	$FS3$	$FS4$	$FS5$
Storage(total)	13824B	18432B	23040B	27648B	32256B
Average PSNR	26.64dB	26.93dB	27.01dB	27.05dB	27.08dB
Lena	30.33dB	30.73dB	30.84dB	30.91dB	30.95dB

Similarly in Table 5 we have summarized the inverse halftone image quality, and storage requirements of 16 and 19 size LUT inverse halftoning [28], and the inverse halftoning algorithm 'fastiht2' reported in [21]. In the last row, the average PSNR value of the inverse halftone images in the test set are reported.

From these tables it can be seen that the image quality achieved with tree structure $FS1$ is better than the image quality achieved with LUT inverse halftoning using the '19pels' template. Note that the latter needs 512KB for storage whereas $FS1$ needs less than $13.5KB$. Also, the inverse halftone image quality achieved with $FS1$ is better than fastiht2 method. Besides, the inverse halftone quality is superior for mandrill image which has a lot of high frequency content. Image quality for $FS1$ can be increased by adding more tree leaves to $FS1$. Bigger trees with storage requirements are reported in second, third and fourth columns of table 7. Even though, the average PSNR of the TLUT inverse halftoned images in the test set increases, visually, the image quality remains almost the same. For visual comparison the inverse halftone images for Lena images can be seen better at the website [25].

Similarly TLUT algorithm can be applied to 3×6 clustered dot ordered dither halftones [27]. The resulting TLUT inverse halftone quality is better than the image quality of the result of algoritm II in [29] and this can be achieved with a storage requirement of only 9.56KB [27]. When we train our TLUT trees on dispersed dot ordered dithered with 8×8 Bayer's matrix we get similar results [27].

8. LUT HALFTONING

We will process pixels one by one in raster-scan order. In order to decide the halftone value at a chosen cell (or pixel location), we will use the halftone values already decided in a carefully selected template or neighborhood of the chosen cell and the contone value of the chosen pixel. The template design phase is merely a process of deciding which neighboring cells should be involved in the prediction. The template selection algorithm can be found in [30].

Let us assume that there are N pixels (excluding the pixel being predicted) in the neighborhood and they are ordered in a specific way. Let us also call the halftone values of pixels as p_0,p_1,...,p_{N-1} and the contone value of the pixel being predicted as c. Note that there are $2^N 2^8$ different patterns since $p_i \in \{0,1\}$ for $i = 0,1,...,N-1$ and $c \in \{0, 2^8 - 1\}$. Since the halftone image is a bilevel image, our LUT should return a binary value for each pattern $(p_0 p_1 ... p_{N-1} c)$: $T_(p_0 p_1 ... p_{N-1} c)$. The details on the design of the LUT and the non-existent pattern estimation can be found in [27] and [30].

LUT example: We have chosen $N = 15$ and constructed our template using the algorithm given in the previous section. The training set included several error diffused halftone images (see below). This template can be found in Ref. 27. Note that we need $2^N 2^8 = 2^{23} Bits$ (1MBytes) to store the LUT. We have trained our LUT with images using the following images: Lena, peppers, grey ramp, boat, airplane, Zelda, grey scale ramp and two more smooth images. The halftones of these images for training are obtained with error diffusion. Afterwards we halftoned goldhill with the designed LUT. Notice that goldhill was **not in the training set**. Except in regions of very low and very high grey levels the LUT halftoning method gives the same image quality as error diffusion. Notice that, the storage requirement of LUT halftoning is also high. In the next section we will introduce TLUT halftoning in order to overcome these problems.

9. TREE-STRUCTURE LUT HALFTONING

As mentioned above, LUT halftoning has some defects in regions of very low and very high grey levels. It can be seen that the halftone pattern for $g = \frac{1}{256}$ has approximately 16×16 periodicity. LUT halftoning with a template which does not have a pixel which is 16×16 pixels away from the halftone pixel being predicted cannot capture this periodicity. Different cells should be added to capture different halftone patterns. However, the template size cannot be increased arbitrarily because of storage problems. This problem can be solved by adding cells adaptively to the template. These "*adaptive cells*" can be stored efficiently in a tree structure.

Tree structure: Let us denote the size of the initial template used as a (this is typically small, e.g., $a = 11$). We will define 2^{a+8} binary trees corresponding to the different patterns in the template. Each tree node is either split further or it is a leaf. Tree nodes are split so that the halftone values of LUT obtained with initial template can be refined. If a node is split, then the location of additional pixel,(i,j), is stored in the node and two more nodes are attached to this node as its children. If a tree node is a tree leaf, then a halftone value is stored in the node. The details on tree structure can be found in [27] and [27].

TLUT halftoning algorithm: In TLUT halftoning, we try to find a tree leaf for each pixel in the halftone image. After finding the tree leaf, the halftone value stored in the tree leaf will be assigned as the halftone value of the pixel. To find the corresponding tree leaf for each halftone pixel location we will do the following:
1. First look at the pattern inside the initial template $\mathcal{T}$ of size a. Each different pattern will correspond to one of the binary trees. The root of the corresponding tree is declared as the current node.
2. Each node is either split into two nodes or it is a leaf. If a node is a leaf, then the halftone value is stored in the node. This value is assigned as the halftone value at the pixel.
3. If a node is split into two, then the location (i,j) of the additional pixel is stored in the node. Get the halftone value of the pixel which is (i,j) away from the current pixel. If this value is 0(1), then left(right) node is assigned as the current node. Then go to step 2.

TLUT example: We have chosen our initial template to consist of the first eleven elements of the template used for LUT halftoning. Then we have refined the trees with the training set as in Sec. VI for LUT halftoning with $\mathcal{N}_{17}^{'}$. The TLUT halftone image can be seen in [27],[30]. The problems with very high and very low

Figure 6: Goldhill halftoned with TLUT halftoning trained on DBS Images.

grey levels in LUT halftoned images do not occur for TLUT halftoning algorithm. The total storage cost is approximately 158KB which is much less than storage requirements of LUT halftoning (1 MB).

We have also trained TLUT on DBS halftones. The halftones are obtained by applying fifty steps of DBS iterations on FS error diffused halftones [10]. Our initial template consists of the first eleven elements of the template used in LUT halftoning [30]. The TLUT halftoned image for Goldhill is shown in Fig. 6. Notice that the quality of the halftone image is better than error diffused halftone image (especially in the sky). The dots are more uniformly distributed in the sky in TLUT halftoned image whereas artificial worm artifacts in the sky can be noticed easily in error diffused images. The images can be found at the website [25] for better viewing.

10. CONCLUSION

In this paper we first briefly mentioned digital halftoning methods. In order to asses the halftone quality, a model for Human Visual System was given. This model was used in optimization of the dot diffusion halftoning algorithm. Even though dot diffusion offers more parallelism than the popular error diffusion algorithm, it has not received much attention in the past. This is partly because the noise characteristics of error diffusion method are generally regarded as superior. We have shown that dot diffusion method with a carefully optimized class matrix is very promising. Since the enhancement step prior to halftoning can be objectionable in some cases, we eliminated this by making the class matrix larger. Then we reviewed inverse halftoning algorithms and discussed a novel method for inverse halftoning which produces images of very good quality. The LUT method for inverse halftoning is extremely fast and the image quality achieved is comparable to the best methods known for inverse halftoning. The method does not depend on the specific properties of the halftoning method, and can be applied to any of them. Then we have discussed tree-structured LUT (TLUT) inverse halftoning in order to reduce the storage requirements of LUT inverse halftoning. In Section 9, a new LUT based halftoning method was discussed. The algorithm is capable of producing good quality halftones. In order to refine the halftones, tree-structured LUT halftoning was proposed. We have demonstrated that any halftoning algorithm can be simulated with this method. When this algorithm is trained on computationally complex halftoning algorithms which produce very good halftones, it will reduce the computational complexity substantially.

REFERENCES

1. R. A. Ulichney, "Dithering with blue noise," *Proc. of IEEE* **76**, pp. 56–79, 1988.

2. D. Anastassiou, "An optimum method for two level rendition of continuous tone pictures," *Proc. of Conf. Rec. IEEE ICC* **1**, pp. 26–11–26–15, 1973.
3. R. Floyd and L. Steinberg, "An adaptive algorithm for spatial greyscale," *Proc. SID*, pp. 75–77, 1976.
4. C. A. B. B. W. Kolpatzik, "Optimized error diffusion for image display," *Journal of Electronic Imaging* **10**, pp. 2777–2792, 1992.
5. P. W. Wong, "Adaptive error diffusion and its application in multiresolution rendering," *IEEE Transactions on Image Processing* **5**, pp. 1184–1196, 1996.
6. D. E. Knuth, "Digital halftones by dot diffusion," *ACM Tr. on Graphics* **6**, pp. 245–273, 1987.
7. M. Meşe and P. Vaidyanathan, "Optimized halftoning using dot diffusion and methods for inverse halftoning," *IEEE Transactions on Image Processing* **9**, pp. 691–709, 2000.
8. M. Meşe and P. Vaidyanathan, "Properties of improved dot diffusion for image halftoning," *Journal of Imaging Science and Technology* **45**, pp. 291–296, 2001.
9. G. A. D.L. Lau and N. Gallagher, "Green noise digital halftoning," *Proc. of IEEE* **86**, pp. 2424–2444, 1996.
10. J. P. A. M. A. Seldowitz and D. E. Sweeney, "Synthesis of digital holograms by direct binary search," *Applied Optics* **26**, pp. 2788–2798, 1987.
11. T. Mitsa and K. J. Parker, "Digital halftoning technique using a blue noise mask," *J. Opt. Soc. Am. A* **9**, pp. 1920–1929, 1992.
12. R. Nasanen, "Visibility of halftone dot textures," *IEEE Transactions on Systems, Man and Cybernetics* **14**, pp. 920–924, 1984.
13. J. Allebach, "Fm screen design using dbs algorithm," *Proc. of ICIP* **1**, pp. 549–552, 1996.
14. C. Y. P.C. Chang and T. Lee, "Subroutine for the generation of a two dimensional human visual contrast sensitivity function," *Eastman Kodak Tech. Rep.* .
15. J. Allebach and R. Stradling, "Computer-aided design of dither signals for binary display of images," *Applied Optics* **18**, pp. 2708–2713, 1979.
16. M. Analoui and J. P. Allebach, "New results on reconstruction of continuous-tone from halftone," *IEEE Intl. Conf. Acoust. Spech Signal Process.* **3**, pp. 313–316, 1992.
17. S. Hein and A. Zakhor, "Halftone to continuous-tone conversion of error-diffusion coded images," *IEEE Trans. Image Processing* **4**, pp. 208–216, 1995.
18. Z. Fan, "Retrieval of images from digital halftones," *Proc. of ISCAS*, pp. 313–316, 1992.
19. P. W. Wong, "Inverse halftoning and kernel estimation for error diffusion," *IEEE Transactions on Image Processing* **4**, pp. 486–498, 1995.
20. K. R. Z. Xiong and M. Orchard, "Inverse halftoning using wavelets," *Proc. of Intl. Conf. on Image Processing* **1**, pp. 569–572, 1996.
21. B. E. T. Kite, N.D. Venkata and A. Bovik, "A high quality, fast inverse halftoning algorithm for error diffused halftones," *Proceedings of ICIP*, 1998.
22. M. Meşe and P. Vaidyanathan, "Look up table (lut) method for inverse halftoning," *IEEE Transactions on Image Processing* **10**, pp. 1566–1578, 2001.
23. A. N. Netravali and E. G. Bowen, "Display of dithered images," *Proc. SID* **22**, pp. 185–190, 1981.
24. M. Y. Ting and E. A. Riskin, "Error-diffused image compression using a binary-to-gray-scale decoder and predictive pruned tree-structured vector quantization," *IEEE Trans. Image Proc.* **3**, pp. 854–858, 1994.
25. http://www.systems.caltech.edu/mese/research/.
26. M. Meşe and P. Vaidyanathan, "Tree-structured method for improved lut inverse halftoning," *Proc. of IEEE EUSIPCO*, 2000.
27. M. Meşe and P. Vaidyanathan, "Tree-structured method for lut inverse halftoning and for image halftoning," *to appear in IEEE Transactions on Image Processing* .
28. M. Meşe and P. Vaidyanathan, "Template selection for lut inverse halftoning and application to color halftones," *Proc. of IEEE ICASSP*, 2000.
29. R. d. Q. J. Luo and Z. Fan, "A robust technique for image descreening based on the wavelet transform," *IEEE Trans. on Signal Processing* **46**, pp. 1179–1184, 1998.
30. M. Meşe and P. Vaidyanathan, "Look up table method for image halftoning," *Proc. of ICIP*, 2000.