Reducing the Complexity of Multiscale Error Diffusion

Ka-Chun Lui and Yuk-Hee Chan

Department of Electronic and Information Engineering
The Hong Kong Polytechnic University, Hong Kong

Abstract - **Multiscale error diffusion (MED) is superior to conventional error diffusion algorithms as it can eliminate directional hysteresis completely. However, the complexity of this frame-oriented process is much higher and makes it not suitable for real-time applications. In this paper, a fast MED algorithm is proposed. The complexity of this algorithm is remarkably reduced as compared with conventional MED algorithms. It also supports parallel processing.**

I. Introduction

Error diffusion is a widely used digital halftoning technique for emulating a grayscale image with black and white dots [1]. Various error diffusion algorithms have been developed in recent years [2]. Among them, multiscale error diffusion (MED) algorithms [3-5] were shown to be superior to conventional error diffusion algorithms such as Floyd and Steinberg's algorithm[6] in a way that it can eliminate directional hysteresis completely. This is achieved by the fact that, unlike those conventional algorithms, MED algorithms do not quantize pixels in a fixed sequential order and diffuse the quantization error with a non-causal filter. However, due to their frame-oriented nature, their complexity is comparatively high and makes them not suitable for real-time applications. This paper presents an alternative to realize multiscale error diffusion. Without sacrificing the output quality, its realization complexity is remarkably lower than that of the other MED algorithms and its realization can be done in parallel as well.

II. Proposed Algorithm

Similar to other MED algorithms, the proposed algorithm removes the scanning-path and filter constraints to eliminate directional hysteresis. The difference is that it puts its focus on the realization complexity and reduces it by tackling the technical problems in a different way.

Unlike [3-5], the proposed algorithm is block-based to allow parallel processing. The quantization error of a pixel is usually consumed during its propagation to a distant pixel. The diffusion result of two distant pixels is likely to be independent and hence processing blocks in parallel makes sense to a certain extent.

In conventional MED[3,4], all pixel values of the output image **B** are initialized to be zero and then, based on the grayscale input image **X**, an appropriate number of pixels of **B** are picked iteratively to assign value 1 until the average pixel intensity of **B** is equivalent to that of **X**. From another point of view, white dots are iteratively put in a black background. A considerable amount of realization effort is paid for locating the positions to put the white dots and this effort is proportional to the number of white dots to be introduced. The proposed algorithm reduces the complexity by reducing the number of dots to be put and the amount of effort to locate a position for a dot.

To reduce the number of dots to be handled, the proposed algorithm first estimates the average intensity of **X**. Without losing of generality, we assume that the maximum and the minimum pixel values of **X** are, respectively, 1 and 0. If the average pixel value of **X** is less than 0.5, white dots should be introduced to a black background. Otherwise, black dots should be introduced to a white background to reduce the realization effort. Hereafter, we assume that white dots are the minority dots and they are introduced to a black background. If it is the opposite, one can negate all pixel values of **X** before carrying out the proposed algorithm and negate all pixel values of the output at the end. The dot budget is defined to be the number of minority dots to be settled and it is the rounded value of $\mathbf{min}(S_x\text{-}I_x,I_x)$, where S_x is the total number of pixels in **X** and I_x is the sum of all pixel values of **X**. Operator $\mathbf{min}(\bullet)$ picks the minimum value among the inputs.

The grayscale input image **X** is then partitioned into a number of 4×4 non-overlapped blocks. For each block, an intensity pyramid is constructed as shown in Figure 1. In formulation, we have

$$E^k_{(p,q)}(i,j)=\begin{cases}\sum_{m=0,1}\sum_{n=0,1} E^{k-1}_{(p,q)}(2i+m,2j+n) & \text{if } k=1,2\\ X_{(p,q)}(i,j) & \text{if } k=0\end{cases}$$
$$\text{for } i,j=0,1\cdots\max(0,2^{2-k}-1) \qquad (1)$$

where $X_{(p,q)}(i,j)$ is the intensity value of the $(i,j)^{th}$ pixel of the $(p,q)^{th}$ block of **X** and $E^k_{(p,q)}(i,j)$ is the value of the $(i,j)^{th}$ element of the k^{th} level of the intensity pyramid associated with the $(p,q)^{th}$ block of **X**.

Every 4 adjacent blocks are grouped together to form a macroblock of 8×8 pixels. Except those macroblocks whose total pixel intensity value is less than 0.5, which implies no more white dot should be put to them, all macroblocks are processed in parallel as follows. For each macroblock, the block which carries the maximum total intensity (i.e. the block which has the maximum $E^2_{(p,q)}(0,0)$) in the macroblock is picked and the most wanted pixel in the block is located with the intensity pyramid associated with the selected block by following the maximum intensity guidance. Specifically, when the maximum intensity guidance is adopted, one should always proceed from the current node at level k to its child node of maximum

1-4244-0549-1/06/$20.00 ©2006 IEEE.

$E^{k-1}_{(p,q)}(i,j)$. At any time, when there is more than one maximum encountered in the search, we select one of them randomly. When level 0 is reached, the selected node specifies the most wanted pixel. For instance, if the node holding $E^{0}_{(p,q)}(x,y)$ is the node, the $(x,y)^{th}$ pixel of the block will be the most wanted pixel. Unless the selected pixel is a boundary pixel of the macroblock, $X_{(p,q)}(x,y)$ should then be quantized to 1 and $B_{(p,q)}(x,y)$, the intensity value of the $(x,y)^{th}$ pixel of the $(p,q)^{th}$ block of **B,** is assigned value 1. The reason for discriminating the boundary pixels of a macroblock will be discussed later. For the sake of reference, the region in which a pixel can be quantized after being selected is referred to as a qualified region.

Suppose the selected pixel is in the qualified region of a macroblock. After quantizing it to 1, its quantization error $e = X_{(p,q)}(x,y) - 1$ is diffused to the neighbors of the pixel with a non-causal filter to update **X** as follows.

$$X_{(p,q)}(x+m,y+n) = \begin{cases} 0 & \text{if } m=n=0 \\ X_{(p,q)}(x+m,y+n) + e\ w(m,n) & \text{else} \end{cases} \quad \text{for } -1 \le m,n \le 1 \qquad (2)$$

where $w(m,n)$ is a coefficient of the diffusion filter defined as $\frac{1}{12}\begin{bmatrix} 1 & 2 & 1 \\ 2 & * & 2 \\ 1 & 2 & 1 \end{bmatrix}$. The asterisk marks the position of $w(0,0)$.

Since only the central part of a macroblock can be quantized and the diffusion filter is of size 3×3, no error can be diffused outside the macroblock and hence all marcoblocks can be processed independently. In other words, all macroblocks can be processed in parallel to reduce the processing time. This explains why, in the proposed algorithm, boundary pixels of a macroblock are discriminated and not further processed when they are selected. After diffusion, all intensity pyramids of the affected blocks are updated.

In order to provide a chance to handle the boundary pixels of a macroblock and eliminate the potential blocking artifacts caused by the block-based approach, the proposed algorithm changes the way how it groups blocks to form a macroblock in the course of halftoning as follows. After processing all macroblocks as mentioned before in parallel once, all blocks of **X** are regrouped to form new macroblocks. Four grouping schemes are used in turns in the proposed algorithm. As an example, Figure 2 shows how the 4 schemes group the blocks in an image of size 6×6 blocks differently. A pixel which is a boundary pixel of a macroblock in a particular round may not be a boundary pixel of a macroblock again in next round. By doing so, all pixels of **X** can be taken care in the course. Note that the regrouping does not affect the intensity pyramids of the blocks and hence does not increase the complexity in this aspect.

The overall effect of using the grouping schemes in turns and excluding the boundary pixels of a macroblock from being processed is equivalent to processing overlapped 6×6-pixel regions each of which overlaps each of its 4-connected neighboring 6×6-pixel regions with an area of 2×6 or 6×2 pixels. Blocking artifacts can hence be eliminated with this approach.

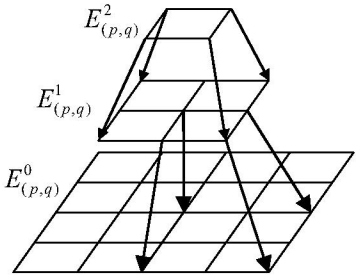


Figure 1. Intensity pyramid associated with a 4×4 block

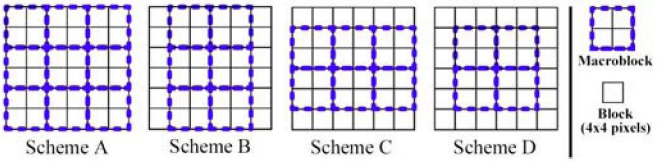


Figure 2. Four grouping schemes

Pseudo Code

Determine the number of minority dots that should be put
Construct intensity pyramids of all blocks
While the minority dots have not yet totally been located
  For Scheme *S* = A, B, C, D
    Partition image X with Grouping Scheme *S*
    For each macroblock
      If total residual intensity of the macroblock>0.5
      (*Here we assume minority dots are white. The criterion should be adjusted if they are black.*)
        Locate the most wanted pixel
        If it is in the qualified region of the macroblock
          Quantize it
          Diffuse quantization error
        End
      End
    End
    Update intensity pyramids of all affected blocks
  End
End

Parallel

Figure 3. Pseudo code of the proposed algorithm

Figure 3 summaries the proposed algorithm in pseudo code. The algorithm iteratively allocates white dots to **B** until all budgeted dots are used up. At each iteration, a considerable number of white dots are allocated at a time. When allocating the dots, it quantizes corresponding pixels of **X** and diffuses the quantization errors. It is possible that, at the last stage of the halftoning process, while there are still budgeted dots on hand, there is no macroblock whose total residual intensity is larger than 0.5. In such a case, to settle the budgeted dots left behind, we select a proper number of macroblocks which carry the most total residual

intensity among all macroblocks to locate the pixels for putting the dots.

To handle boundary pixels or corner pixels of **X**, diffusion filters such as $\frac{1}{8}\begin{bmatrix}0 & 0 & 0\\ 2 & * & 2\\ 1 & 2 & 1\end{bmatrix}$ or $\frac{1}{5}\begin{bmatrix}0 & 0 & 0\\ 0 & * & 2\\ 0 & 2 & 1\end{bmatrix}$ are used instead to avoid energy leakage. The macroblocks containing these corner or boundary pixels should also extend their qualified regions to allow quantizing these pixels.

III. Complexity Analysis

Assume that the input image is of size $N\times N$, where N is a multiple of 4. At the initial stage, N^2 additions are required to construct $N^2/16$ energy pyramids and determine the total intensity level of minority dots.

The realization complexity for the steps left behind is then roughly proportional to the number of minority dots ($\le N^2/2$) to be settled in the output. To settle a dot, all involved operations are confined in a macroblock. First of all, 9 comparisons are required to locate the most wanted pixel. If it is in the qualified region of a macroblock, the searching effort will not be wasted. Since there are 36 pixels in the qualified region of an 8×8 macroblock, a reasonable estimate of the hit ratio is 36/64 though the real hit ratio is higher than this in our simulation. Accordingly, on average, the effort for locating a qualified pixel is 9(64/36)=16 comparisons. 2 multiplications and 9 additions are required to diffuse the quantization error. Finally, at most 12 additions are required to update the affected blocks in the macroblock. This extreme case happens when all 4 blocks are affected as shown in Figure 4.

By considering comparison as addition, the upper bound of the complexity is roughly $2N^2/2$ multiplications and $N^2+(9+16+12)N^2/2$ additions, which implies at most 1 multiplication and 19.5 additions per pixel.

The initialization stage of Katsavounidis's algorithm [3] takes N^2-1 additions. For each introduced dot, it takes $3\log_2 N$ comparisons to locate the most wanted pixel, 9 additions and 2 multiplications to diffuse the quantization error and at most $4\log_2 N$ additions to update the energy pyramid. Unlike our proposed algorithm, Katsavounidis's algorithm introduces white dots instead of minority dots and hence the number of introduced dots is bounded by N^2 instead of $N^2/2$. The upper bound of the complexity is roughly $2N^2$ multiplications and $N^2-1+(7\log_2 N+9)N^2$ additions, which implies at most 2 multiplications and $7\log_2 N+10$ additions per pixel. Consequently, in contrast to the proposed algorithm in which the complexity bound per pixel is a constant, its complexity bound per pixel is $O(\log_2 N)$.

Since block overlapping and block shifting are, respectively, used in [4] and [5] to remove block effect, the structure of the intensity pyramids involved is more complicated as compared with that used in Katsavounidis's algorithm. Accordingly, their realization complexity is even higher. In particular, the complexity bound of [5] is roughly 3-fold of that of [3].

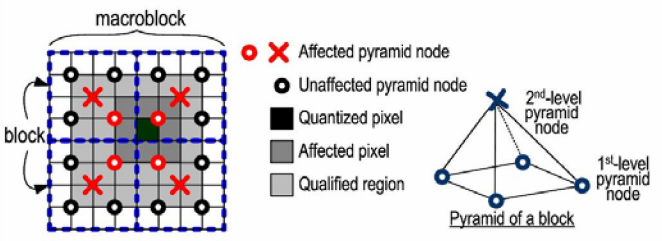


Figure 4. Elements to be updated in a pyramid

IV. Simulation Results

Simulations were carried out to evaluate the performance of various MED algorithms. Figure 4 shows their performance in terms of *radically averaged power spectrum density* (RAPSD). RAPSD is defined as the average power of the halftoning result of a constant gray-level input in the annular ring with center radius f_p as follows [9].

$$P(f_p)=\frac{1}{N(R(f_p))}\sum_{f\in R(f_p)}\hat{P}(f) \tag{3}$$

where $N(R(f_p))$ is the number of frequency samples in $R(f_p)$ which is an annular ring of width Δ_p partitioned from the spectral domain. $\hat{P}(f)$ is the magnitude square of the Fourier transform of the output pattern divided by the sample size. For easier comparison, in all plots shown in Figure 5, any RAPSD value which is larger than 10 is clipped to 10.

A good blue noise generator should produce a result which carries weak or no low frequency spectral components. The result should also provide a flat high frequency spectral region and a spectral peak at blue noise principal frequency f_b. In order to provide a clear picture of the performance of the algorithm, a white surface which marks the principal frequency f_b for a particular gray level is added in each of the plots as a reference for comparison. One can see that the proposed algorithm is as good as [4] and [5]. The harmonics appeared in Figure 5a explain why strong pattern noise appears in the outputs of [3].

Simulations were also carried out to evaluate the performance of the proposed algorithm on a set of *de facto* standard 512×512 8-bit gray scale images including *Lena*, *Baboon*, *Boat*, *Peppers* and *Barbara*.

Halftone visibility metric [2] can be used to measure the visible distortion between **X** and **B**. This metric is defined as

$$MSE_{hv}=\frac{1}{N\times N}\left\|hvs(\mathbf{X},vd,dpi)-hvs(\mathbf{B},vd,dpi)\right\|^2 \tag{4}$$

where *hvs* is the HVS filter function defined in [2], *vd* is the viewing distance in inches and *dpi* is the printer resolution. In our simulations, the viewing distance was fixed at 20 inches and printer resolution of 600dpi was considered. The MSE_{hv} of MED_k[3], MED_c98[4] and the proposed algorithms are, respectively, 0.227, 0.020 and 0.028.

Figure 6 shows the diffusion results of some MED algorithms for subjective evaluation. No blocking effect

appears in the output of the proposed block-based MED algorithm.

Table 1 shows the average number of operations required to halftone the evaluated set of 512×512 test images in the simulation. The minority dots of most test images are white and hence the gain achieved by handling minority dots instead of white dots is not remarkable in the simulation. Even so, the complexity of the proposed algorithm is roughly 65% of that of [3]. Note that this reduction does not result in any sacrifice in the quality. The processing time can be further reduced if the algorithm is realized in parallel.

V. Conclusions

In this paper we proposed a fast block-based parallel MED algorithm. Like any other MED algorithms, this algorithm can completely eliminate directional hysteresis. With an appropriate combination of partition and grouping schemes, the algorithm reduces its complexity remarkably as compared with other MED algorithms. Unlike other MED algorithms, the proposed algorithm supports parallel processing, which makes it possible for real-time applications. Simulation results show that it is comparable to [4] and [5] and superior to [3] in terms of output quality. No obvious blocking effect appears in its output.

Acknowledgment

This work was supported by a grant (A-PA3U) from The Hong Kong Polytechnic University.

References

[1] D. L. Lau and G. R. Arce, Modern Digital Halftoning, Marcel Dekker, New York NY, USA 2001.

[2] M. Mese and P.P.Vaidyanathan, "Recent advances in digital halftoning and inverse halftoning methods," IEEE Transactions on Circuits and Systems I, June 2002, Vol.49, pp.790-805

[3] I. Katsavounidis, C. C. J. Kuo, "A multiscale error diffusion technique for digital halftoning," IEEE Trans. on Image Processing, Mar 1997, Vol. 6, No.3, pp.483-490

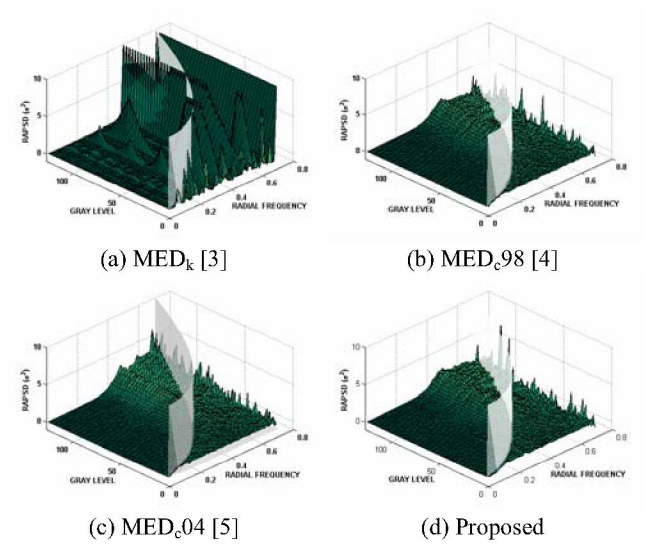
[4] Y.H. Chan, "A modified multiscale error diffusion technique for digital halftoning," IEEE Signal Processing Letters, Nov 1998, Vol.5. pp. 277-280

[5] Y.H. Chan and S. M. Cheung, "Feature-preserving multiscale error diffusion for digital halftoning," Journal of Electronic Imaging, Jul 2004, Vol.13, No.3, pp.639-645

[6] R.W. Floyd and L. Steinberg, "An adaptive algorithm for spatial grey scale" Proc. Soc. Inf. Display, 17:75-77, 1976.

[7] E.Peli, ``Multiresolution, error-convergence halftone algorithm," J. Opt. Soc. Am. A, 1991, 8(4), pp.625-633.

[8] R.A. Ulichney, Digital Halftoning. Cambridge, MA:MIT Press, 1987.



(a) MED_k [3] (b) MED_c98 [4]

(c) MED_c04 [5] (d) Proposed

Figure 5. Performance of various MED algorithms in terms of RAPSD

Operation	Operations / pixel			
	Ours	[3]	[4]	[6]
ADD	8.89	13.95	13.47	4.99
CMP	6.65	12.92	34.46	1.00
MUL	0.93	0.96	0.96	3.99
Total	**16.47**	**27.83**	**48.89**	**9.98**

Table 1. Average computational complexity of various algorithms

(b) Proposed (c) MED_k [3] (e) MED_c98 [4]

Figure 6. Halftones produced with various MED algorithms