Halftoning by Back Error Compensation

*Gabriel Marcu and Satoshi Abe**
Array Corporation, Shibuya-ku, Tokyo, Japan
**Faculty of Computer Science, University of Tokyo*
Tokyo, Japan

Abstract

This paper describes a halftone method based on the minimization of a decision function integrates the errors produced in the previously computed pixels. The decision function is computed by weighting the pixel errors, according to a matrix corresponding to the HVS. The current pixel is the minimization variable, and value that minimizes the decision function over the output range is selected as the output pixel value. The method is a slightly modified version of the method presented in [1], concerning the computation of error term. It is demon-strated that, in the modified form, the minimization criteria is equivalent to the error diffusion for a binary output, but the error diffusion is the method that requires minimum computation. The method presented here conducts to more linear transfer function than the method presented in [1]. It is also shown that the selection of the weighting matrix is not crucial and similar results can be obtained with different weighting coefficients.

In the second part of the paper an error diffusion that works independently on the processing direction is presented. In this method, all the symmetric neighbor pixels around the kernel center are involved in computation. The processed neighbors contribute to compute the modified input and the unprocessed pixels absorb the diffused error of the current pixel. Results of both methods are presented.

Introduction

The error diffusion is a very popular technique for digital halftoning.[1,2,3] In [4] we introduced a variant of error diffusion based on a decision function. There, the error term was computed as a simple difference between input and the output pixels, and that paper emphasized that the characteristic of the transfer function is not linear. In this paper, we introduce a correction for computation of the error term in [4]. This correction improves the linearity of the transfer function of the algorithm. It is demonstrated that, in the modified form, the algorithm is equivalent to the error diffusion for a binary output. In fact the error diffusion by thresholding is a better solution since it reduces to minimum the number of computation required by decision function implementation. It is also shown that the selection of the weighting matrix is not crucial and similar results as those presented here can be obtained with different weighting coefficients.

Back Error Compensation Based on Decision Function

We summarize the method presented in [1] where the method was introduced for the general case of an arbitrary color palette, {**c**}, with N elements, each color being a vector with S components, **c** = (c_1, c_2, c_S). For each pixel in the original image, the method selects that color from the output set, that minimizes the error of color rendition in a local array, **A**, around the current investigated pixel. Figure 1 depicts an example of a local array corresponding to the current pixel (j,k), when the processing direction of the image is from up to down and from left to right.

	$i=-2$	$i=-1$	$i=0$	$i=1$
$j=-2$	$(m-2,n-2)$	$(m-2,n-1)$	$(m-2,n)$	$k-2,n+1)$
$j=-1$	$(m-1,n-2)$	$(m-1,n-1)$	$(m-1,n)$	$(k-1,n+1)$
$j=0$	$(m,n-2)$	$(m,n-1)$	$(\mathbf{m,n})$	

processing direction → current pixel

*Figure 1. Example of local area **A(m,n)***

For a pixel (m,n), the decision function, f(m,n), is:

$$f(m,n) = \| \mathbf{in}(m,n) - \mathbf{out}(m,n) - \mathbf{E}(m,n) \|, \quad (1)$$

where **E**(m,n) is the vector function:

$$\mathbf{E}_{(m,n)} = \sum_{\substack{(j,k)\in\mathbf{A}(\mathbf{m},\mathbf{n}) \\ (j,k)\neq(m,n)}} w(j,k).\mathbf{err}(j,k), \quad (2)$$

The term w(i,j) represents the weight mask of the error terms inside the local array, **A**. As it was introduced in [1], term **err**(i,j) represents the vector error defined as:

$$\mathbf{err}(j,k) = in(j,k) - \mathbf{out}(j,k), \quad (3)$$

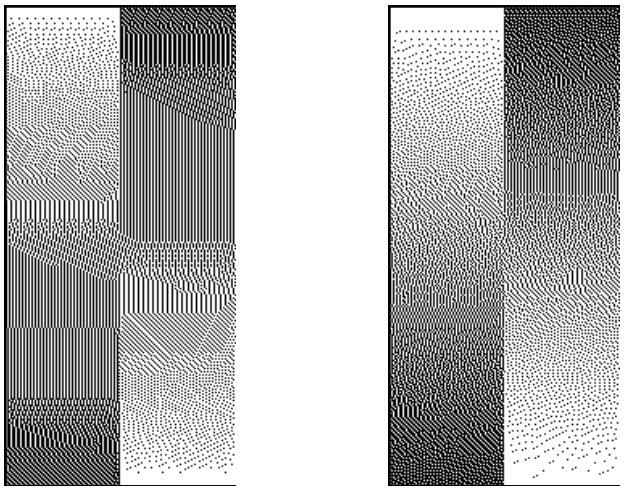
for all the elements denoted by (j,k) inside the local array, **A**, and **in**(j,k) and **out**(j,k) are the values of the input and output image. Note that the input pixel **in**(m,n) is included in the function (1) and the output value **out**(m,n) represents the minimization variable of f(m,n). For (j,k) ≠ (m,n), the **err**(j,k) values are computed in the previous processing step using the relationship (2).

Correction of the Error Term

In this paper we replace the equation (2) by the relationship (3) where the error term is the value of the vector decision function:

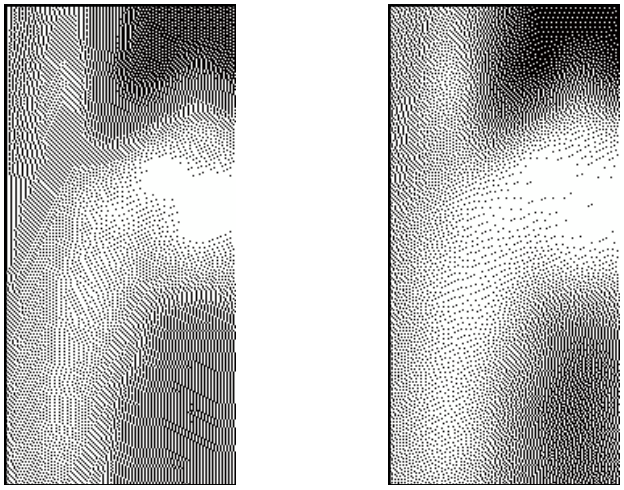
$$\mathbf{err}(m,n) = \mathbf{in}(m,n) - \mathbf{out}(m,n) - \mathbf{E}(m,n). \quad (4)$$

Figure 2 presents a test wedge processed with the method introduced in [1] (2.a) and with the correction introduced in this paper (2.b). Figure 3 presents a sample image processed with the method introduced in [1] (2.a) and with the correction introduced in this paper (3.b). In [1] the pattern effect is reduced by enlarging the local area, but here the weights of Floyd and Steinberg are used. The equation (4) attenuates the pattern effect and conducts to more linear transfer function.



(a) before correction (b) after correction

Figure 2. Test wedge processed using back error compensation before (a) and after (b) correction of error term.



(a) before correction (b) after correction

Figure 3. Sample image processed using back error compensation before (a) and after (b) correction of error term.

Equivalence to Conventional Error Diffusion

In the following we will demonstrate that the equation (1), (2) and (4) are equivalent to conventional error diffusion. In case of black and white images, the equation (1) (2) and (3) are reduced to:

$$f(m,n) = | in(m,n) - out(m,n) - E(m,n) |, \quad (1a)$$

$$E(m,n) = \sum w(j,k) \bullet err(j,k), \quad (2a)$$

$$err(m,n) = in(m,n) - out(m,n) - E(m,n). \quad (3a)$$

Assuming the binary output, the f(m,n) becomes minimum for one of the two possible output values, 0 or 1. Assuming the value 1 gives the minimum value of f(m,n). In this case, we have:

$$| in(m,n) - 0 - E(m,n) | < | in(m,n) - 1 - E(m,n) |,$$

or the equivalent form:

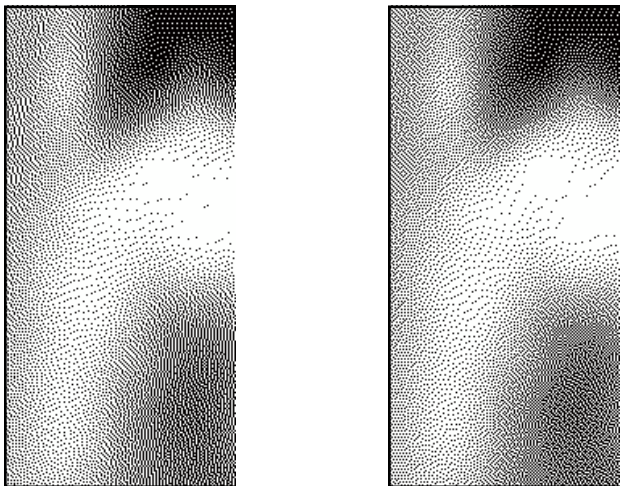
$$in(m,n) - E(m,n) < 0.5,$$

that represents the threshold comparison of the conventional error diffusion. Same result can be derived from the case that 1 is selected as the output value.

It is easy to see that the error diffusion requires the minimum number of computation compared with the implementation that uses the decision function. For the multilevel error diffusion, the modified input is computed first, then based on this value, the threshold value is selected. The minimum of the decision function requires the computation of f(m,n) for all the values in the output range, increasing significantly the computation time.

Gaussian Weighting Mask

Empirically it was verified that the weighting coefficients are not critical to the middle tone of the image. Figure 4 shows the same region of a sample image rendered using the Floyd & Steinberg coefficient mask (4.a) and a symmetric Gaussian mask clipped to the error diffusion area (4.b), for a raster processing direction, as it is illustrated in Figure 5.



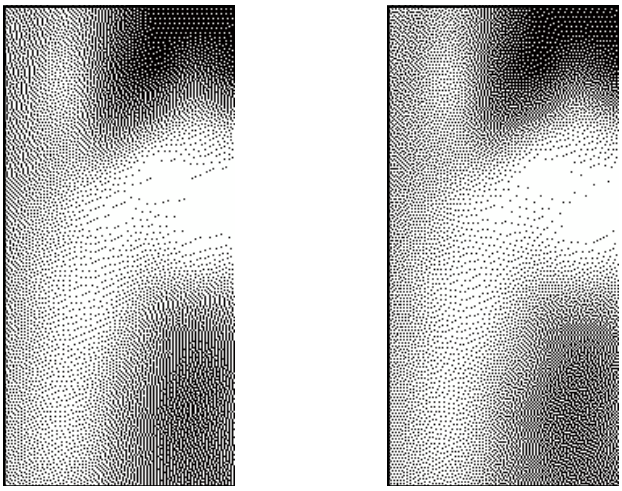
(a) Floyd-Steinberg (b) Gaussian

Figure 4. Sample image rendered with two weighting coefficient sets.

In fact, some of the region of the test sample displays a better arrangement of dots when the Gaussian coefficients are used instead of Floyd and Steinberg coefficients. It can be verified that other set of coefficients gives also similar results. Figure 6 shows the sample image processed for asymmetric packed weights (6.a) and clipped (unpacked) weights (6.b). This empirical observation is in concordance with recent theoretical results5 that adapt the weighting mask to the gray level of the input.

1	3	1	off	off	off			
3	×	3 + off	×	on =	×	3		
1	3	1	on	on	on	1	3	1

Figure 5. Clipping weighting mask to the error diffusion area for raster processing direction.



(a) packed (b) clipped

Figure 6. Error diffusion using packed and clipped weights presented in Figure 7.

1	2	1		1	2	4		
2	12	2	=	5	12			
1	2	1						

(a) packed weights

1	2	1		1	2	1		
2	12	2	=	2	6			
1	2	1						

(b) clipped weights

Figure 7. Packing and clipping the Gaussian weights

Symmetric Kernel Error Diffusion

Based on the Gaussian symmetric weighting mask, we derive a variant of error diffusion that process the pixel independently of the advancing direction of error diffusion. Initially, all the pixels of the image are marked as not processed, msk(j,k) = 0, ∀j,k. The pixel to be processed is considered the center of a symmetric low pass filter kernel, **A**. An example of filter kernel is presented in the left side of Figure 7. Each processed pixel is marked with msk(m,n) = 1.

For each pixel, the algorithm computes selectively the accumulated error, E(m,n), considering only the processed pixels in the kernel,

$$\mathbf{E}_{(m,n)} = \sum_{\substack{(j,k)\in \mathbf{A(m,n)} \\ (j,k)\neq(m,n)}} w(j,k).\mathbf{err}(j,k).msk(j,k), \qquad (5)$$

and then modifies the input value:

$$in'(m,n) = in(m,n) - E(m,n).$$

The algorithm selects the output value by thresholding the modified input:[3]

$$out(m,n) = step\ [in'(m,n) - T].$$

$$Err(m,n) = in'(m,n) - out(m,n);$$

Finally, the error between the modified input and the computed output is weighted and diffused to the remained unprocessed pixels in the kernel, regardless the processing sequence:

for $((j,k) \in \mathbf{A(m,n)}, (j,k) \neq (m,n))$:

in $(j,k) = in(j,k) - Err(m,n).w(j,k).msk(j,k);$

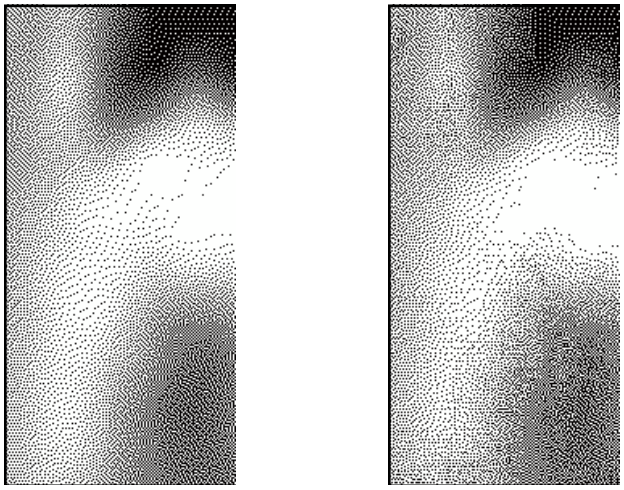
The error diffusion algorithm can start anywhere in the image. The processing sequence can be selected arbitrary, because a kernel pixel is automatically considered either to contribute with an error term for modified input or to absorb the accumulated error diffused from the center. The weighting matrix is unitary derived from a two dimension Gaussian function. The coefficients have symmetric values with respect to the kernel center. The computation of a pixel involves all the pixels of the symmetric kernel: part of them are used to integrate the errors of processed pixels and the remaining part serves to diffuse the error to the unprocessed pixels.

For raster processing from up to down and left to right, the process presented here is reduced to conventional error diffusion. In this case the processed and unprocessed pixels are separated in two compact area as it is illustrated in Figure 8. Packing the symmetric filter kernel of processed area over the unprocessed coefficients will result in the Floyd and Steinberg weighting mask.

1	3	1		1	3	5		
3	16	3	=	7	16			
1	3	1						

Figure 8. Packing the Gaussian kernel into the Floyd and Steinberg weighting mask.

Figure 9 illustrates the described algorithm processing the image in raster order (9.a) and increasingly gray level order (9.b). Combination with other techniques is also attractive and will be left for further discussion.



(a) raster order (b) gray level order

Figure 9. Symmetric error diffusion kernel processed in: raster order (a) and gray level order (c).

Conclusions

In the first part it was presented a correction to a previously proposed variant of error diffusion.[4] The corrected version have a more linear transfer function. I was demonstrated that the presented version is equivalent to conventional error diffusion. In the second part of the paper a symmetric error diffusion algorithm is presented. The proposed method enables an arbitrary direction of processing.

References

1. R. Floyd, L. Steinberg, "An adaptive algorithm for spatial gray scale", *SID Int. Symp. Digest of Technical Papers*, 1975, V4, p. 36.
2. A. Ulichney, *Digital Halftoning Techniques*, McGraw Hill, 1987.
3. K. T. Knox, R. Eschbach, "Threshold modulation in error diffusion", *J. Electronic Imaging*, V**2**, July 1993, p. 185–192.
4. G. Marcu, S. Abe, "An error diffusion method for color reproduction in ink jet printing", *Recent Progress in Digital Halftoning*, Reiner Eschbach, Ed., IS&T, 1994, p. 25–27.
5. P. W. Wong, "Adaptive error diffusion and its application in multiresolution rendering", *IEEE Trans. On Image Processing*, V**5**, N7, July 1996, p. 1184-1196.

❋ Previously published in *NIP12*, pp. 132–135, 1996.