A Dot Placement Approach to Stochastic Screening Using Bitmasks

Mike Woods
Software Imaging
Oxford, United Kingdom

Abstract Heading

FM or stochastic screening is a popular approach to halftoning for many applications. The error diffusion algorithm delivers extremely good screen quality but at the price of a computationally-intensive runtime process. Point processes, using either dither arrays or bitmask sets, have efficient runtime requirements but often produce halftones of much lower quality. The generation of such screens usually involves starting with a random pattern and applying some simulated annealing process to gradually improve its characteristics. This paper proposes a method for generating stochastic patterns that employs a dot placement algorithm in which each dot is placed in a position "appropriate" for producing good stochastic output. The algorithm is then enhanced by applying a smoothing step at the end of each halftone pattern generation to adjust any dots that, due to the placement of later dots, are now in sub-optimal positions. Although the algorithm can be used to produce dither arrays, it is primarily aimed at generating bitmasks where the additional degree pattern freedom is exploited to improve pattern smoothness. The algorithm also permits second order stochastic patterns for use with imprecise print devices such as electro-photographic printers.

Introduction

In 1973 Bayer proposed an algorithm for rendering continuous tone images to bi-level devices that claimed to give optimum dither patterns.[1] However, although the dispersion of dots within Bayer's patterns might be theoretically optimal, the strong periodic artifacts are far from satisfactory. Three years later Floyd and Steinberg proposed the "error diffusion" algorithm.[2] By delivering non-deterministic, dispersed dot patterns it offered a marked improvement over Bayer's artifacts, albeit at an increased processing cost. In 1987 Ulichney[3] applied the term "blue noise" to such patterns on the basis of their power spectra and suggested ways to improve error diffusion to bring the power spectra closer to the presumed ideal. Since then many others have proposed ways to improve error diffusion output (for example, by using hexagonal sampling,[4] or adding edge enhancement[5]). Such enhancements have allowed error diffusion algorithms to deliver very high quality halftone patterns. However, error diffusion remains fundamentally a computationally-expensive algorithm compared to a point process.

In 1991 Sullivan[6] proposed halftoning using a set of stochastic bit patterns, each of which is generated by starting with a random pattern and applying a software annealing process to gradually transform the noise from white to blue (utilizing the DFT domain to measure the quality of the pattern). While this technique produces good patterns for any given halftone, the lack of any positional relationship between halftone levels leads to false contours when processing image data.

The following year Mitsa and Parker suggested a similar technique to Sullivan but adapted for generating dither arrays.[7] The use of the dither array prevents the contouring issue of the bitmask set but imposes a constraint on dot placement that makes it difficult to achieve good blue-noise patterns throughout the dynamic range. Ulichney also demonstrated the generation of blue-noise patterns in dither arrays using a "void-and-cluster" method.[8] Still using an annealing process, the method employs a Gaussian filter to identify holes and dot clumps and then transfers dots between. (Spaulding et al. have produced a useful review of these techniques.[9])

In comparison to the above halftoning methods that take an evolutionary approach to generation, the algorithm presented here takes its inspiration from error diffusion. The patterns are built up one dot at a time with the placement decision being influenced by current dot positions. (That said, the algorithm also incorporates a smoothing step for each halftone pattern, though still using the same technique as for initial placement.)

Although the basic algorithm can be applied to the generation of dither arrays, it is preferred to create bitmask sets. This allows an extra degree of freedom in the pattern generation that can improve overall halftone quality (though care is needed to avoid introducing too much noise into images by the issue of false contouring).

Algorithm Outline

We wish to create a set of N bitmask patterns, each of which is of dimensions $W \times H$. (There being one bitmask pattern for each represented halftone output.) The patterns will be created from lightest to darkest. (For simplicity of presentation, it is assumed that a bit value of 0 is a white pixel and a bit value of 1 is a black pixel.)

In addition to a matrix to hold the bitmask under construction an additional matrix of equivalent dimensions is maintained. Called the Noise Map it is filled with random, floating-point values in the range -1 to +1. These values are filtered to eliminate duplicates.

To create a bitmask, the pattern is initialized to that of the previous completed bitmask (with the first bitmask being initialized to an empty pattern). The required extra dots for this halftone level are placed at suitable positions. Finally, a smoothing step is applied to the pattern to adjust dots that are in sub-optimal positions.

To place a dot, a weight value is first calculated for each 0 pixel. A set of candidates is then chosen where each candidate has a weight value within a defined tolerance of the minimum weight value. From this set is chosen the pixel position that has the least corresponding noise value from the Noise Map. That pixel is set to 1.

The Weight Function

The weight value of a pixel ($W(x,y)$) is the sum of the weight (w) exerted upon it by nearby dots (pixels of value 1). The actual equation used to calculate w is a matter of choice but in tests Eq. 1 has been found to give good results. (The Gaussian filter used by Ulichney[8] would be an alternative.)

$$w = \frac{1}{D} - \frac{1}{R} \tag{1}$$

In Eq. 1 D is the distance between the pixel and the dot. In calculating D it is necessary to consider that the resulting patterns are repeated in a tile arrangement. For example, the delta along the horizontal axis is calculated by Eq. 2.

$$\Delta x = \begin{cases} |x_2 - x_1| & |x_2 - x_1| \le \frac{W}{2} \\ W - |x_2 - x_1| & |x_2 - x_1| > \frac{W}{2} \end{cases}, \tag{2}$$

For small patterns it may be reasonable to include all dots in the $W(x,y)$ calculation but for realistic sizes it is reasonable to restrict the area of interest to a neighborhood of the pattern centered on the pixel in question. In Eq. 1 this neighborhood is assumed to be a circle of radius R. Hence, the exerted weight (w) tends to zero at the edge of the neighborhood.

The sizing of the neighborhood is clearly an important aspect of the algorithm. A smaller neighborhood reduces processing time but can introduce unevenness in the macro structure of the patterns. Also in light and dark tone patterns the minority pixels are widely dispersed so the neighborhood needs to be large enough to ensure a reasonable sampling. In comparison the mid-tones do not require a particularly large neighborhood because the dot density is high. It will also be appreciated that there is a finite limit to R as it cannot exceed half the dimensions of the bitmask, otherwise some dots will be counted twice in the sum (which will distort the patterns).

By experiment it has been found that good results may be achieved if R is varied from ½ the dimensions at the tone extremes to ¼ the dimensions at the mid-tone.

It will be appreciated that the first dot of the first bitmask will be placed at an arbitrary position within the mask (though as the mask is tiled this is irrelevant). If the foregoing algorithm is considered, it will also be apparent that the second dot will be placed randomly within the area of the mask not covered by the first dot's weight neighborhood. Although it may be argued that a better algorithm is needed for this second dot this is only an issue at very small mask sizes (such as 16x16 or 32x32). At more practical sizes the extra noise of the second dot placement is insignificant (particularly after pattern smoothing is applied).

The above algorithm has been found to produce very pleasing stochastic patterns. However, for certain output devices the mid-tones can appear "patchy" due to what might be termed "checkerboard" artifacts. (Such artifacts may be observed in classical error diffusion output[3].) For these cases it is necessary to introduce a small amount of extra noise into the patterns. This is easily achieved by modifying the weight value according to the corresponding noise value from the Noise Map ($N(x,y)$). This is defined in Eq. 3. This includes a function $f(t)$ that scales the noise value according to the current tone.

$$W(x,y)' = W(x,y) \cdot (1 + N(x,y) \cdot f(t)) \tag{3}$$

A suitable definition for $f(t)$ is still a matter for experimentation.

Pattern Smoothing

As has already been mentioned, when all dots have been placed for a bitmask pattern there is then performed a smoothing step. This is advantageous as although dots are placed so as to form good patterns, later placements can mitigate against earlier decisions.

The smoothing step involves repeatedly examining the pattern for a dot that is inappropriately placed and moving it to a better position. The step completes when no suitable dot can be identified.

To test a dot it is experimentally removed from the pattern and then placed back again according to the dot placement algorithm. If its position changes the delta in its weight value is stored, and the dot is replaced in its original position. When all dots have been thus tested a short-list is drawn up of those dots whose weight value delta is within a defined tolerance of the maximum delta. From this list, as with dot placement, the dot with the least corresponding noise value is chosen.

As the patterns are held as bitmasks it is perfectly possible to move all the dots during the smoothing step. However, while this produces very pleasing stochastic patterns it introduces an unacceptable level of false contouring in images. Therefore, the smoothing step distinguishes between dots that have been placed for the

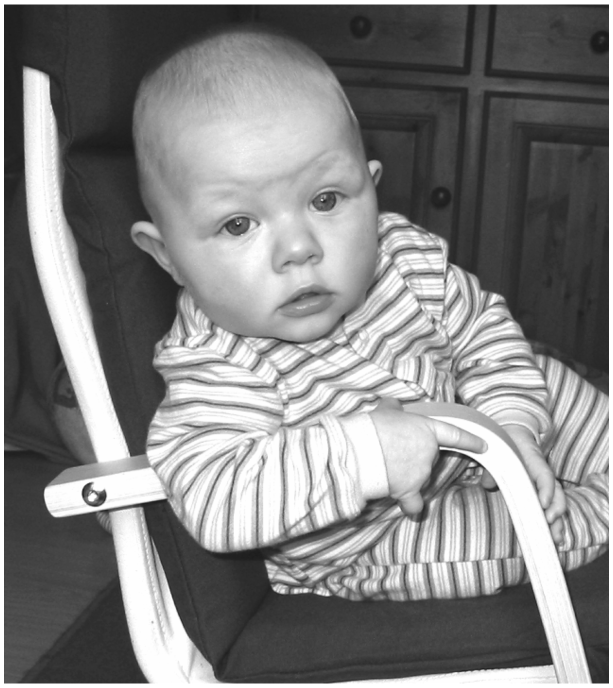

Figure 1. Original image

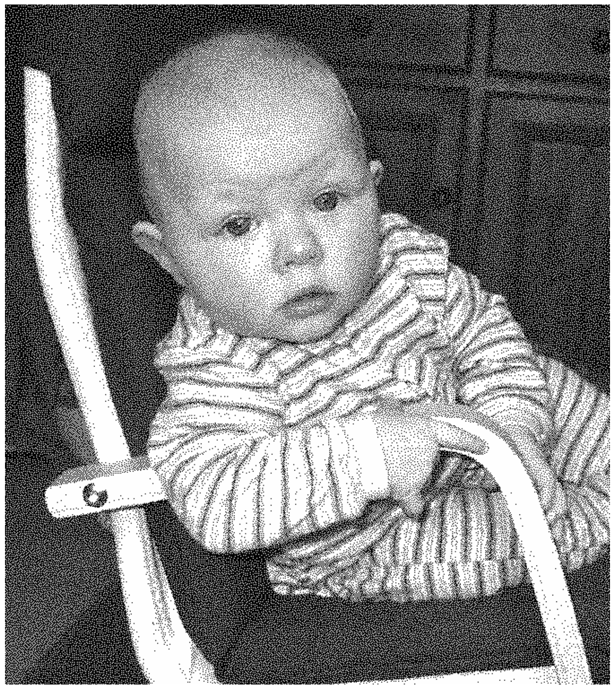

Figure 2. Halftoned image using stochastic bitmask

current bitmask pattern and those inherited from the previous bitmask. The former may be moved at any time, the latter only if the move offers a significant advantage. Rules for determining precisely when the advantage is sufficiently significant are still a matter of research. Interesting results have been achieved when allowing a small proportion of inherited dots to move and by allowing such moves only in the early stages of the smoothing step, but further work is required to quantify the cost of the additional contouring noise.

Non-square Bitmasks

In the algorithm overview it was indicated that the bitmask need not be square, but for simplicity this was ignored in the definition of the dot placement algorithm. The matter will now be dealt with.

Given that one of the features of a stochastic screen is to appear aperiodic, there is no general advantage in supporting non-square bitmasks. However, many devices operate with non-square resolution aspect ratios. To create a physically square tile for such devices the bitmask of necessity must be non-square. To work well in this scenario the algorithm needs to accommodate the non-square nature of the pixels.

In point of fact only minor extensions to the algorithm are required. If the dimensions of the bitmask are assumed to produce a square in the output image, it is simple to define a scaling to convert between pixel co-ordinates and isotropic co-ordinates at the lower resolution. This mapping is given by Eq. 4.

$$\begin{aligned} x_o &= x \cdot \frac{\min[W,H]}{W} \\ y_o &= y \cdot \frac{\min[W,H]}{H} \end{aligned} \qquad (4)$$

When calculating the weight value of a pixel ($W(x,y)$) the isotropic co-ordinate space is used for *D* and *R*. All other parts of the algorithm can work in pixel space.

Second-Order Patterns

Many classes of output device are not capable of reliably reproducing isolated, single pixel dots (for example, electro-photographic printers). For such devices dispersed dot halftoning, such as stochastic screening, often yields disappointing results. While mid-tones are generally reproduced with acceptable quality the light and dark tone ends of the halftone range tend to under- and over-saturate, respectively. In extreme cases, more than half the dynamic range is lost.

The proposed algorithm has been adapted to accommodate such devices and has produced promising results on laser printers. However, at time of writing the work is not ready for publication. It is hoped to include more details in the oral presentation.

Likewise, many print devices are capable of more than bi-level output. Again, the proposed algorithm and resulting bitmasks have been applied to halftoning on such devices with very pleasing results.

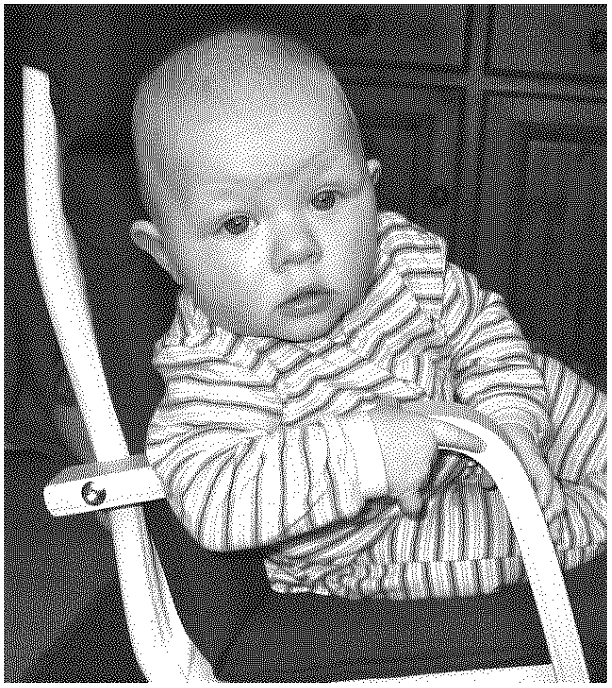

Figure 3. Halftoned image using serpentine Floyd-Steinberg error diffusion

Results

Results may be seen in figures 1 to 4. Fig. 1 shows the original grayscale image. Fig. 2 gives the image when halftoned with a 256×256 bitmask. For comparison, figures 3 and 4 show the same image when halftoned using serpentine Floyd-Steinburg error diffusion and Bayer's algorithms, respectively.

With the low-resolution image used it is possible to discern more noise in the bitmask output than that of the error diffusion. This is largely a consequence of the compromise of dot positioning in the bitmask so as to avoid false contours. Because of its algorithmic nature error diffusion is able to smoothly vary the dot frequency without such limitations. (See section 5 of Spaulding et al.[9]) In actual print scenarios this difference is much less observable.

Comments on Performance

As presented the bitmask generation algorithm is computationally expensive. In particular the smoothing step is extremely time-consuming for large bitmasks. As this is generation-time, not runtime, it may be argued that the time is irrelevant. However, in the real world development resources are often constrained so generation time can still have an impact on an algorithm's utility.

Fortunately it is not difficult to optimize the algorithm to greatly improve performance. Key to this is the creation of the Weight Map. This allows the weight value of each pixel to be stored rather than generated each time it is needed. As dots come and go from the pattern the Weight Map is updated, but as a pixel's effect is restricted to its neighborhood, the cost of updating is far less than the cost of recalculation. (Indeed, there is an order of magnitude difference.)

Further improvement can be made by pre-calculating all values for w within the neighborhood.

The result of these (and other optimizations) is generation times for 256×256 bitmask sets in around 45 minutes on an average-performance PC.

Regarding runtime performance, it has been found that halftoning using bitmasks is approximately ten times faster than a commercially available error diffusion algorithm (which is more sophisticated and of higher quality than the Floyd-Steinburg algorithm) for bi-level output and about four times faster for multi-level output. These timings are also comparable with halftoning using threshold arrays.

Conclusion

A new method for generating stochastic screens has been presented. This method is particularly well-suited for halftoning using bitmasks. Its simple dot-placement algorithm has been shown to be effective in producing pleasing results. Its low run-time requirements make it a suitable choice for print scenarios where error diffusion is too expensive or where the slight quality advantage of error diffusion is not required.

It is also noted that within the presented method there is much scope for enhancing the resulting pattern quality through experimentation.

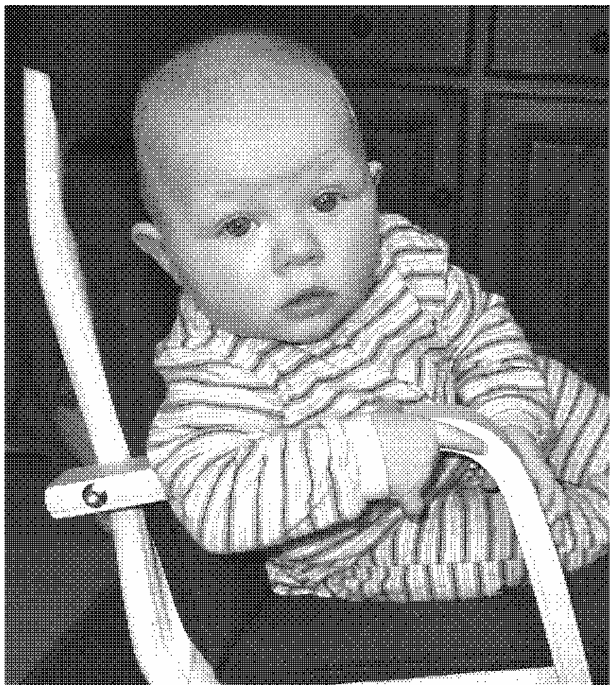

Figure 4. Halftoned image with Bayer pattern

References

1. B. E. Bayer, An Optimum Method for Two-Level Rendition of Continuous-Tone Pictures, *International Conference on Communications,* Vol. 1, pp. 26-11-26-15 (1973)
2. Robert W. Floyd and Louis Steinburg, An Adaptive Algorithm for Spatial Greyscale, *Proc. SID*, Vol. 17(2), pp. 75-77 (1976)
3. Robert A. Ulichney, *Digital Halftoning*, MIT Press 1987
4. R. L. Stevenson and G. R. Arce, Binary display of hexagonally sampled continuous-tone images, *Journal of the Optical Society of America A*, Vol. 2(7), pp. 1009-1013 (1985)
5. Reiner Eschbach and Keith T. Knox, Error-diffusion algorithm with edge enhancement, *Journal of the Optical Society of America A*, Vol. 8(2), pp. 1844-1850 (1991)
6. J. Sullivan, L. Ray and R. Miller, Design of Minimum Visual Modulation Halftone Patterns, *IEEE Trans. On Systems, Man and Cybernetics*, Vol. 21(1), pp. 33-38 (1991)
7. Theophano Mitsa and Kevin J. Parker, Digital halftoning technique using a blue-noise mask, *Journal of the Optical Society of America A*, Vol. 9(11), pp. 1920-1929 (1992)
8. Robert Ulichney, The void-and-cluster method for dither array generation, *Human Vision, Visual Processing and Digital Display IV, Proc. SPIE*, Vol. 1913, pp. 332-343 (1993)
9. Kevin E. Spaulding, Rodney L. Miller and Jay Schildkraut, *Recent Progress in Digital Halftoning II*, IS&T, Springfield, VA, 1999, pp. 225-247

Biography

Mike Woods is Chief Scientist at Software Imaging, the world's largest independent supplier of printer driver technology to Original Equipment Printer Manufacturers. With over 20 years experience as a professional programmer, he was chief architect of the company's award-winning PrintMagic driver system, which was used by EPSON for their successful Stylus Color printers. He maintains a special interest in color halftoning. He holds an MSc in computer studies from Brunel University, London.