Generation of Blue Noise Arrays by Genetic Algorithm

Jeffrey Newbern and V. Michael Bove, Jr.

Massachusetts Institute of Technology Media Laboratory
20 Ames Street, Cambridge, MA 02139 USA

ABSTRACT

Halftoning or quantizing by means of a threshold array is simple, fast, and easily parallelized: a matrix of threshold values is tiled across the image and each output pixel is colored white if the image value exceeds the threshold value and is colored black otherwise. The computational efficiency and locality of the compare operation makes this technique suitable for applications in printing and motion video quantization. In the past, threshold arrays have generally been used with regular-appearing patterns such as clustered-dot "classical" halftoning or Bayer's dispersed-dot patterns. Ulichney has presented a heuristic method for generating blue-noise threshold arrays which do not appear regular, and offer the visual advantages of error-diffusion without its computational costs. Such heuristic methods are capable of generating high-quality threshold arrays, but they are not flexible or controllable enough to enable tuning for particular applications or output device characteristics. We present instead a genetic method for generating a blue-noise threshold array that optimizes a set of criteria encoded in a fitness function, which can be specified to reward any desired attributes. Although the genetic method is computationally intensive, the cost is incurred only once, and the resulting array can be used for millions of images. We compare images halftoned using our arrays with other blue-noise array and error-diffusion methods, and examine the spectral characteristics of the resulting patterns.

Keywords: halftoning, printing, genetic algorithms

1. INTRODUCTION

Halftoning takes advantage of the eye's nonuniform response to spatial frequency, distributing quantization error into frequencies in which it is least visible. Halftoning by means of a threshold array is simple and fast, requiring only a comparison operation, and because it operates over a fixed neighborhood it can easily be performed in parallel. A matrix of threshold values is tiled across the image and each output pixel is colored white if the image value exceeds the threshold value and is colored black otherwise. The computational simplicity makes this an attractive method for both video and print.

The arrangement of threshold values in the array completely determines the halftone pattern at each gray level. Dispersed-dot patterns place more of their quantization error at higher spatial frequencies than do patterns with clustered dots, making them the preferred patterns for halftoning on devices which can faithfully reproduce single pixels.

The most widely used dispersed-dot patterns are those generated by Bayer's ordered dither arrays. Bayer's dispersed-dot patterns are homogeneous, but they impart a regular structure to the output which detracts from the image's content.

Blue noise arrays provide alternative dispersed-dot patterns to those of Bayer. Blue noise arrays attempt to mimic the spectral properties of error-diffusion halftoning processes; they give the benefits of dispersed-dot patterns without the regular structures present in Bayer's patterns.

Ulichney presented a method for constructing blue noise arrays based on the heuristic of redistributing pixels from tight clusters into voids without pixels. Ulichney's method is a greedy algorithm which performs an optimal application of the void-and-cluster heuristic at each gray level, but does not attempt to find a globally optimal threshold array.

---

1This research was performed in conjunction with the Print Systems Engineering group at Digital Equipment Corporation, Marlboro, MA. J. Newbern is now with SpeedSim, Inc., N. Chelmsford, MA.
Heuristic-based methods are capable of generating high-quality blue noise patterns, but they suffer from an inflexibility which prevents them from being tuned to particular halftoning applications and output devices. We present a genetic method for generating a threshold array that optimizes a set of criteria encoded in a fitness function. The genetic method achieves a high degree of flexibility by allowing substitution of an arbitrary fitness function which rewards any desired threshold attributes.

The genetic method is computationally intensive, requiring generation and evaluation of large numbers of candidate threshold arrays, although algorithmic optimizations can reduce the cost of evaluating the fitness function for each offspring. The genetic method also has the computational benefit that it can begin working from an existing array, thereby taking advantage of faster methods for generating arrays of moderate fitness. Because the cost of generating a threshold array need only be incurred once and can be amortized over millions of printed pages, it is economical to make significant computational investments in threshold array generation.

2. A GENETIC METHOD FOR ARRAY CREATION

The genetic method begins with an initial population of 250 candidate threshold arrays, each of which is assigned a numerical score according to a fitness function. The algorithm mutates and combines the arrays in its population to create new arrays and replaces the least fit members of the population with the most fit of the new offspring, causing the average fitness of the population to rise over time. By preferentially breeding the most fit members of the population, the optimization can be made to proceed more rapidly.

Threshold arrays are represented as an $M \times N$ matrix of 8-bit threshold values, almost identical to that used for halftoning by Adobe’s PostScript implementation. The array representation satisfies the invariant that there are identical numbers of each threshold value in the array, thus ensuring that the array produces linear tone scale on an ideal output device.

Each generation, the 100 least fit members of the population are killed off and replaced by 100 new offspring. The new offspring are produced from parents chosen through a process of roulette wheel selection, in which members of the population are chosen to breed according to a fixed probability associated with their rank within the population. We assigned roulette probabilities of \( \frac{10}{250} \times \left( \lceil \frac{250 - r}{25} \rceil + 1 \right) \) to each member of the population, where \( r \) is the member's rank. This selects the most fit individual ten times more often than the least fit.

The selected parent arrays are combined and altered by mutation operators to produce an offspring array which must maintain the linearity invariant. A crossover operator produces an offspring array which combines information from two parent arrays, whereas a mutation operator produces an offspring array which is a variation on a single parent array.

Because a standard crossover operator could violate the linearity invariant, a special order-based crossover operator is used instead. The order-based crossover operator combines the threshold values of one parent with the threshold ordering information of both parents. Because all of the values in the offspring come from only of the parents, the offspring array satisfies the invariant if the parent does. Since all of the arrays in the initial population satisfy the invariant when they are created and no mutator violates the invariant, all arrays produced by the genetic algorithm will also satisfy the invariant, and will thus give linear tone scale rendition on an ideal output device.

We experimented with combinations of seven mutation operators which swap pairs of threshold values within the array:

**Point-Swap-Same:** Swaps the values of a randomly chosen pair of cells whose thresholds are both below 0.5 or both above 0.5.

**Point-Swap-Close:** Swaps the values of a randomly chosen pair of cells whose thresholds differ by no more than 8%.

**Point-Swap-Different:** Swaps the values of a randomly chosen pair of cells where one threshold is below 0.5 and the other is above 0.5.

**Smoothness-Swap:** Swaps the value of a cell which neighbors a cell which performs poorly on the smoothness component of the fitness function with the value of another cell which will improve the smoothness score for each. The pairs are found by repeated random trials with a goal which relaxes at each iteration.
Redistribution-Swap: Swaps the values of a pair of cells in such a way that the tone scale of a $4 \times 4$ region containing each cell is improved (made more linear). The linearity is determined by a coarse 16-bin histogram of the area. Since the area contains 16 cells, each bin in a histogram of a perfectly linear region should contain a single cell. If a region is not perfectly linear, then at least one bin will contain more than one cell and at least one bin will contain no cells. The mutator swaps a cell from the most crowded bin with a cell which will fill its empty bin. The other cell is taken from a region in which the cell with the needed value is in a crowded bin.

Point-Movement: Randomly swaps a point with another point fewer than three cells away in the threshold array.

Repulsion-Movement: Swaps a point with another point up to four cells away. The direction of motion is determined by a repulsion calculation which takes into account the configuration of thresholds in the neighborhood around the selected point.

After each offspring array is created, it is scored according to the fitness function and inserted into the population at its proper rank. If the array is ranked highly, the mutation operator which produced it is rewarded and it becomes more likely to be used in the future; if the array is ranked at the bottom of the population, the mutation operator which produced it is penalized and becomes less likely to be used again in the future. In this way, mutation operators which perform well can be promoted and operators which perform poorly can be inhibited until conditions change and they begin to perform well again.

3. FITNESS EVALUATION

The most important part of the optimization process is the fitness function, which assigns a numerical fitness score to each threshold array in accordance with the desirability of the array. The fitness function is evaluated each time a new candidate array is bred, and is easily the dominant factor in determining the running time of the algorithm.

We used a fitness function which rewards visual uniformity and isotropic distribution of points in every gray pattern, and penalizes low-frequency artifacts at 25%, 50%, and 75% grays. The fitness function is calculated as the product of three components: smoothness, repulsion, and partitioning.

The smoothness component rewards visual uniformity in the midtones of the grayscale. It roughly models the eye as a Gaussian low-pass filter and measures the mean squared error between the desired gray level and the eye-filtered halftoned pattern produced by the threshold array for gray levels at about 2% intervals in the middle of the grayscale.

The repulsion component rewards isotropic distribution of minority pixels (white pixels in a pattern below 50% gray or black pixels in a pattern above 50% gray) in the highlight and shadow regions of the grayscale. At each gray level, $g$, the repulsion routine computes $r_g$, the average over all minority pixels of the sum of $e^{-d^2/25}$, where $d$ is the shortest straight-line distance between the pixel and each other minority pixel in a $19 \times 19$ cell neighborhood of the halftone pattern around the pixel. A $19 \times 19$ neighborhood is used because it is large enough to yield meaningful results for gray levels all the way out to 1/256 or 255/256 gray, when only one in 256 pixels will be a minority pixel. A measure of the deviation from uniformity is obtained by taking the weighted average of $r_g$ normalized against the $r_g$ obtained by using Bayer's ordered dither to generate the halftone patterns. Any normalized $r_g$ greater than 10 is replaced by $7.7 + \log(r_g)$ in the calculation of the weighted average, to prevent the optimization from being dominated by large deviations which may be acceptable under some circumstances.

The normalized $r_g$ values are weighted to allow greater deviations in the extreme regions of the tone scale, where the small number of minority pixels lead to large deviations from Bayer's values, even for acceptable threshold arrays. The weighting also places less emphasis on the repulsion values in the midtones, where desirable halftone patterns are better determined by other means.

The partitioning component penalizes undesirable artifacts and textures in the halftone patterns at 25%, 50%, and 75% gray levels. Undesirable patterns at 50% gray include horizontal, vertical, and diagonal lines, checkerboards and variations on checkerboards. Undesirable patterns at 25% and 75% gray include adjacent minority pixels.

The checkerboard pattern is an optimal solution for representing 50% gray on an ideal output device, however, arrays which generate this pattern produce unsatisfactory patterns at other gray levels around 50%, because any
variation from a perfect checkerboard produces a visible low-frequency defect in the pattern. For this reason, the
partitioning component of the fitness function is used to ensure that the partitioning of pixels at 50% gray into those
which are black and those which are white does not form a perfect checkerboard or a checkerboard variant with
similar visual characteristics.

The partitioning calculation classifies each pixel as OK, Bad or Worst based on its role in the halftone patterns
at each of the three gray levels, and computes a fitness score which penalizes patterns with greater numbers of Bad
and Worst pixels.

Calculation of the fitness function, particularly the repulsion and smoothness components, is computationally
intensive, but the computation can be made more efficient using standard optimizations. Lookup tables speed up the
repetitive calculations of weights in the repulsion and smoothness calculations. Symmetry considerations allow the
repulsion component for all gray levels to be calculated with a single neighborhood operation per threshold value.
Because the value of the smoothness component will not change when two thresholds which fall within the same
2% interval are swapped, re-calculation can be avoided in this case. Similarly, re-calculation of the partitioning
component can be avoided when two thresholds which occupy the same 25% interval are swapped. In the later stages
of optimization, these small changes occur with increasing frequency as fine adjustments are made to the threshold
array. Finally, when a point swap occurs, the repulsion component can be updated in constant time, avoiding the
$O(M \times N)$ cost of re-calculation.

Schermesser and Bryngdahl\textsuperscript{6} have recently presented a halftoning method based on a numerical texture metric.
While their method optimizes over an entire image, their metric might be adapted to the fitness function in a genetic
threshold array generation method.

4. RESULTS

The genetic method was applied to the task of generating $32 \times 32$ blue noise arrays for an ideal output device.
Figure 1 shows the result of halftoning a grayscale ramp using a void-and-cluster blue noise array,\textsuperscript{3} Floyd and Steinberg
error diffusion\textsuperscript{7} and our genetic method. The GA thresholds provide a blue noise pattern comparable to that of the
void-and-cluster thresholds in the highlights and shadows, but are less grainy in the regions around 25% and 75%
gray and have fewer low-frequency artifacts around 50% gray. Both the void-and-cluster and the GA thresholds are
free of the contours and directional textures which mar the image halftoned with the error diffusion technique. Those
textures can be reduced by introducing random variations into the error diffusion process.

Figure 2 compares a test image halftoned with the same techniques. Again, the error diffusion technique introduces
unsightly artifacts which are not present in either the void-and-cluster image or the GA threshold image. The error
diffusion image is sharper than either of the others, and the genetic threshold image is slightly sharper than the
void-and-cluster image.

Figure 3 shows power spectra for 50%, 60%, 75%, and 90% gray patterns halftoned using blue noise arrays
created by the void-and-cluster method and the genetic method and patterns halftoned by error diffusion. The
spectral estimates were generated by first calculating the two-dimensional Discrete Fourier Transform of a $128 \times 128$
halftoned pattern and then calculating the annular average of the squared and normalized frequency samples in order
to obtain a one-dimensional power spectral density estimate. The DC term has been omitted to show the remaining
data at a readable scale.

Because of the periodic nature of threshold arrays, it is difficult to obtain power spectral density estimates with
low variance. The computation of Figure 3 used thirty-four annuli to give both informative resolution in the frequency
domain and acceptably low variance in the frequency samples.

The patterns produced by the GA thresholds show slightly less energy in the low-frequency region of the spectrum
than the void-and-cluster patterns, and the error diffusion process produces better blue noise spectra than either of
the threshold arrays.

Experiments indicate that each of the three fitness function components adds a valuable constraint to the genetic
search. Omitting the smoothness component leads to arrays with unacceptable variations in texture and density,
mitting the repulsion component fails to produce isotropic halftone patterns in the highlight and shadow regions of
the grayscale, and omitting the partitioning component adds more low-frequency content to the halftoned patterns.
Figure 1: Comparison of grayscale ramps using GA thresholds to images using void-and-cluster thresholds and error diffusion.
Figure 2: Comparison of test photo using GA thresholds to images using void-and-cluster thresholds and error diffusion.
Figure 3: Comparison of spectral properties of blue noise arrays generated using the genetic technique with arrays created using the void-and-cluster method and patterns produced by error diffusion. The error diffusion spectrum is not shown for 50% gray because it consists of an impulse at $f_g = \frac{1}{\sqrt{2}}$ which dwarfs the spectral components of the other two images.
Figure 4: Threshold array generated by optimizing all of the components of the fitness function.
Figure 5: Threshold array generated by optimizing only the repulsion component.
Figures 4 and 5 provide an example of the effects of fitness function components on the threshold arrays generated using the genetic method. The threshold array generated using only the repulsion component produces very good blue noise patterns at 10% and 90% grays, but has some low-frequency artifacts at 25%, 50%, and 75% grays which are not present in the patterns produced using all three components of the fitness function.

Of the eight mutation operators used by the algorithm, the order-based crossover operator performed the best. The point-swap mutator which swapped points with close thresholds performed well late in the run, when small adjustments to the array are most successful. The point-swap mutator which swapped minority with majority pixels was found to be of very little use. The three directed mutators were useful but exhibited only mediocre performance.

5. CONCLUSION

We have demonstrated a method for generating high-quality blue noise threshold arrays. Because the genetic method evolves threshold arrays according to qualities encoded in a fitness function, it offers a greater degree of flexibility to manipulate the threshold array than do heuristic-based methods. Comparison of arrays generated by the genetic method against other blue noise arrays and against the error diffusion technique, showed that our arrays have slightly better spectral properties than void-and-cluster arrays and greater controllability and efficiency than error diffusion.

REFERENCES