Comparative Study of Search Strategies for the Direct Binary Search Image Halftoning Algorithm

Sagar Bhatt and John Sabino, Rice University, Houston, TX; John Harlim, University of Maryland, College Park, MD; Joel Lepak, University of Michigan, Ann Harbor, MI; Robert Ronkese, University of Delaware, Newark, DE; and Chai Wah Wu, IBM T.J. Watson Research Center, Yorktown Heights, NY

Abstract

We propose strategies for reducing the operations required by a high-quality halftoning algorithm, Direct Binary Search (DBS) and compare the errors in the resulting halftones from the proposed methods, standard DBS, dithering, and error diffusion.

Introduction

Halftoning is the process of approximating an image containing a continuum of colors by an image containing a few colors. Most modern digital printers rely on halftoning because they cannot vary the intensity of their inks and use only a handful of colors. A black-and-white printer, for example, must use only black ink and white paper to faithfully recreate a grayscale image. Any gray color we see in an image printed by a black-and-white printer is an illusion. At an appropriate distance, the eye perceives closely-spaced black and white blotches as a patch of gray. The exact way in which the eye produces this gray is not completely understood, but a blurring process is evident. A complete model of the human visual system (HVS) is an open research problem and must take many things into account: for example, how the eye blurs together nearby pixels, the sensitivity of the eye to changes in texture and luminescence, the attenuated response of the eye to changes along diagonal directions, and how optical illusions deceive the brain. Several models [4] have been proposed, but a particularly effective and simple one is the Gaussian filter model. The role of the HVS in the halftoning problem leads to the following mathematical formulation.

The Halftoning Problem *Given a full color image I, find a halftone image O that minimizes $\|G(O) - G(I)\|$, where $G(X)$ is the perceived image when viewing X.*

We represent an image as a matrix of pixel values. In the Gaussian filter HVS model, $G(\cdot)$ is a two-dimensional convolution operator with a Gaussian kernel. The norm depends on the setting, but considering the image as a vector, the l_p norm for $p = 2, \infty$ is commonly used.

A particularly restrictive constraint in designing a practical halftoning algorithm is that the method must perform extremely fast. This is especially true in high-speed, high-resolution printers, as there could be more than a hundred million pixels on a page. For these applications, it is desirable that the algorithm produces an acceptable halftone using kN operations, where N is the number of pixels in the original image, and k is a small constant ($k \lesssim 10$). Several extremely fast halftoning algorithms exist, but their halftones is often unsatisfactory. We review a slower method called Direct Binary Search (DBS) [1] and compare it with some of the faster methods: dithering [2, 3] and error diffusion [5–7]

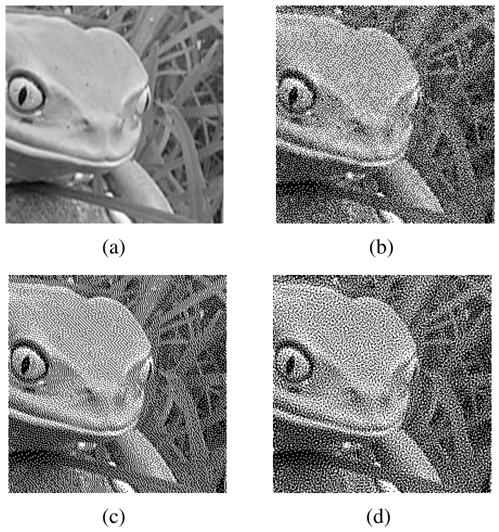

Figure 1. Results of halftoning a 512×512 pixel grayscale image (a) with: (b) dithering, (c) error diffusion, and (d) Direct Binary Search. Cropped images shown. (Note: Your printer may employ a halftoning scheme to image (a) so that it appears to have a finer resolution.)

(see Figure 1). While DBS is considered one of the best halftoning algorithms, its computational cost is too high for practical halftoning in an online environment. The purpose of this paper is to study and compare search strategies of DBS that preserve most of the quality of DBS halftones but with fewer operations.

Direct Binary Search

For simplicity, we present our formulation for grayscale halftoning; a formulation for color images readily follows. We shall represent grayscale intensities by numbers in the interval $[0, 1]$, varying smoothly from white at 0 to black at 1. We define an **image** as a finite two-dimensional array of grayscale intensities. We refer to entries in the array as **pixels** and each pixel has a set of coordinates (i, j). In the rest of this paper, we denote I to be a grayscale input image and O to be a black-and-white output halftone.

Direct Binary Search is a computationally expensive algorithm that requires several passes through an image before converging to the final halftone. DBS proceeds by generating an initial halftone image (possibly using dithering or another fast method), and then performs a local improvement in the halftone by **swapping** (switching the colors of nearby pixels) and **toggling** (changing the parity of an individual pixel). An outline of a basic version of the algorithm is shown in Algorithm 1.

Algorithm 1 Standard DBS

```
Generate an initial halftone $O$ (typically by dithering)
Compute the error $\mathscr{E} := \|\tilde{e}\|_F$
repeat
    for each pixel $(i,j)$ do
        Compute the change in $\mathscr{E}$ caused by
        1. Swapping pixel $O(i,j)$ with a neighbor
        2. Toggling pixel $O(i,j)$
        if either action decreases $\mathscr{E}$ then
            Perform the better one
        end if
    end for
    Update $\mathscr{E}$
until ending criteria are met
```

For a more accurate approximation, a model of the printer spot profile (the actual appearance of the ink dots on the printed page) should also be taken into account. In this paper, we follow [4] by assuming that the spot profile is small enough (compared to the blurring effect of the HVS) to not affect the perception of the image.

The error $\mathscr{E}$ is defined to be

$$\mathscr{E} = \|\tilde{e}\|_F = \sqrt{\sum_{i,j} \tilde{e}(i,j)^2},$$

where the perceived error $\tilde{e}$ is defined by

$$\tilde{e}(i,j) = G(O)(i,j) - G(I)(i,j) = \tilde{p} \star\star (O-I)(i,j).$$

Here $\|\cdot\|_F$ is the Frobenius matrix norm with $\star\star$ denoting two-dimensional convolution and $\tilde{p}$, the HVS filter, is stored as a $P \times P$ matrix. In our case studies, we use $P = 11$, with $\tilde{p}(i+5, j+5) = \exp\left(-(i^2+j^2)/5\right)$.

One can use different criteria for terminating DBS depending on the circumstances. If runtime is not a concern, the algorithm can iterate until no changes are made in an iteration. This would be appropriate for situations such as screen generation for dithering. Another criteria is that the algorithm can terminate when the relative error reduction for an iteration falls below a tolerance. For this paper, the standard DBS algorithm and each of the variants we describe below employ this ending criteria with a tolerance of 0.01. Figure 1 shows a result of implementing DBS using this model.

Throughout the paper, **processing** a pixel should be interpreted as checking for an acceptable swap or toggle and performing the action if one is found. We say that a **trial** has been performed each time that the potential error change of a swap or toggle has been evaluated. In each trial, DBS evaluates the change in error $\Delta\mathscr{E}$ for each possible swap or toggle. As described in [1], $\Delta\mathscr{E}$ can be computed using a few operations in terms of

$$\tilde{p} \star\star \tilde{e} \text{ and } \tilde{p} \star\star \tilde{p}, \qquad (1)$$

When a swap or toggle occurs, the $(2P-1) \times (2P-1)$ components of $\tilde{p} \star\star \tilde{e}$ centered at each altered pixel are updated. Notice that a swap costs twice as much as a toggle since the former requires updating twice as many components as the latter. Updating these components is much more expensive to perform than a trial, especially for large P. For this reason, we concentrate primarily on reducing the number of changes (swaps or toggles) a DBS algorithm needs to make to the initial halftone. We do this by judiciously choosing the set of pixels to consider for trials, swaps and toggles.

New Strategies for Implementing DBS

We will describe now several variations of DBS. Hereafter, we call the DBS reviewed earlier as the standard DBS. We first discuss several strategies for sorting the pixels to give priority to regions of the halftone with higher error. We also consider only accepting swaps or toggles that reduce the error at least as much as a fixed percentage of the mean error reduction of previously accepted changes. A final approach is to refine the set of processed pixels based on the changes accepted, attempting to isolate the high-error regions of the halftone.

Sorting

Pixels are processed in an arbitrary order in standard DBS. We believe that processing high-error pixels first tends to give a more rapid error reduction. In fact, after sorting, just one iteration of DBS is needed to reach a satisfactory overall error level.

For simplicity of notation we assume that we are sorting entries (i,j) based on $|\tilde{e}(i,j)|$. In actual implementation, however, the sorting is based on components of $\tilde{p} \star\star \tilde{e}$ (see [1]). We find no substantial difference between using these two quantities as a sorting criterion. Unfortunately, sorting costs $O(NlogN)$, where N is the number of pixels. To reduce this cost, we propose the following.

Local Sort

Errors are perceived locally, i.e. $\tilde{p}$ has a finite extent, so it is not essential to sort the entire image to find problem pixels in the initial halftone. The image can be broken into constant-size blocks and each block sorted separately without significant degradation of performance. An outline of this revised DBS algorithm is shown in Algorithm 2. Tests on sample images using a block size of only 4×4 have performed nearly as well as globally sorting the pixel locations. Note that the sorting cost now becomes $O(N)$.

Algorithm 2 Local Sort DBS

```
Split the image into $b \times b$ size blocks
Sort each block based on $|\tilde{e}(i,j)|$
repeat
    for $r = 1 \ldots b^2$ do
        for each block do
            Process the pixel with the $r$th highest error
        end for
    end for
until ending criteria are met
```

Regular Spacing

This method is a very inexpensive approximation to Local Sort. Instead of sorting each of the blocks, we only sort one of them and assign this order to all the other blocks. The resulting effect is that pixels processed consecutively are "spaced out" instead of being contiguous. Hence, this method avoids making many consecutive changes in small neighborhoods.

Threshold Refinement

Instead of giving higher priority to pixels at which the error is higher, another strategy is to emphasize the pixels where the most improvement can be made. In threshold refinement, a swap is only made if it reduces the error at least as much as some fraction of the average error reduction of the swaps accepted so far in that pass through the image. Toggles are performed without regard to the threshold because there tend to be far fewer toggles than swaps, toggles are half as expensive as swaps, and a toggle may be necessary to bring the average gray level of the halftone in a given region into agreement with the original image. The primary advantage of threshold refinement is that the expensive updating procedure for $\tilde{p} \star\star \tilde{e}$ is restricted to the pixels that give a substantial reduction in error. To reduce the cost of averaging, we kept a running average in our implementations.

Algorithm 3 Threshold Refinement DBS

```
Input: Threshold weight $\beta$ (A typical value is $\beta = 1/2$.)
$\overline{\Delta\mathscr{E}} \leftarrow 0$ (The bar denotes average.)
for each processed pixel do
    if a toggle reduces the error then
        Perform the toggle
    else
        if the best trial swap gives $\Delta\mathscr{E} < \beta\overline{\Delta\mathscr{E}}$ then
            Perform the swap
            Update $\overline{\Delta\mathscr{E}}$
        end if
    end if
end for
```

Search Set Refinement

One may also attempt to only process regions where changes have been made previously, the idea being that these regions tend to have higher error and are concentrated near the edges. Given an initial set of pixels, Search Set Refinement (SSR) expands and contracts the set according to the swaps and toggles accepted. When a change to a pixel is made, the pixel is kept in the set and its neighbors are added. Otherwise, the pixel is removed from the set. This technique is outlined in Algorithm 4. In our implementations, we refer to the size of a neighborhood by its radius, i.e., the maximum of the vertical and horizontal distances a pixel in the neighborhood is from the center pixel.

Results

Here, we compare the performance of all proposed algorithms described in the previous section. We test each algorithm on 150 images with sizes ranging from 100,000 pixels to 10 million pixels. The implementations are:

- Standard DBS with neighborhoods of size 3×3 (i.e. radius 1).
- Local Sort with 4×4 blocks.
- Regular Spacing with 4×4 blocks.

Algorithm 4 Search Set Refinement DBS

```
Start with an initial set $S$ of pixels (typically a sparse set of
uniformly distributed pixels)
repeat
    for each pixel in $S$ do
        Process pixel
        if pixel was changed then
            Add its neighborhood to $S$
        else
            Remove pixel from $S$
        end if
    end for
until ending criteria are met
```

- Search Set Refinement (SSR) with radius one and an initial set of uniformly distributed pixels chosen by selecting one pixel from each 4×4 block.
- Floyd–Steinberg error diffusion.

All variations of standard DBS are implemented with Threshold Refinement using $\beta = 0.5$. SSR with radius zero is also applied in both Local Sort and Regular Spacing (with the initial search set being the entire image).

The accuracy of each algorithm is measured by

$$\text{Relative Error} = \frac{\text{Error - Error of standard DBS}}{\text{Error of standard DBS}}.$$

Halftones with relative errors up to 30% tend to be visually indistinguishable from halftones produced by standard DBS. Figure 2 plots the relative error as a function of the image size. Our results show that Local Sort produces the least error, with an average of 21% relative error. Notice that the average relative error of SSR, 41%, is about the same as the average relative error of error diffusion, 44%. However, error diffusion has a large variance of relative error compared to our proposed schemes. This is not surprising since our schemes are based on DBS, which minimizes the perceived error, whereas error diffusion does not. See Figure 5 for examples of halftone images produced by Local Sort, Regular Spacing, and Search Set Refinement with radius one.

The efficiency of each proposed scheme is also compared to the standard DBS algorithm. As mentioned before, the main cost is updating $\tilde{p} \star\star \tilde{e}$, which costs twice as much for a swap than for a toggle. The total computational cost is thus on the order of twice the number of swaps plus the number of toggles. Figure 3 shows the ratio of this quantity to the total number of pixels in an image as a function of the image size. This ratio represents the fraction of altered pixels in an image. The standard DBS alters approximately 40% of the pixels in each image, while our proposed schemes altered only about 10% of the pixels. In Figure 4, the total number of trials of our proposed schemes is roughly less than 5 per pixel, about four times less than that of the standard DBS, roughly 20 per pixel. These results indicate that after the initialization of $\tilde{p} \star\star \tilde{e}$, the number of operations in processing the pixels in these schemes is about 25% of the corresponding operations in standard DBS.

Conclusions and future work

We have presented several methods to reduce the number of operations in the DBS algorithm while preserving much of the

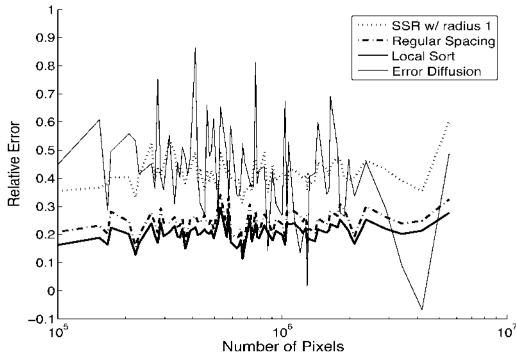

Figure 2. *Relative error with respect to standard DBS as a function of image size.*

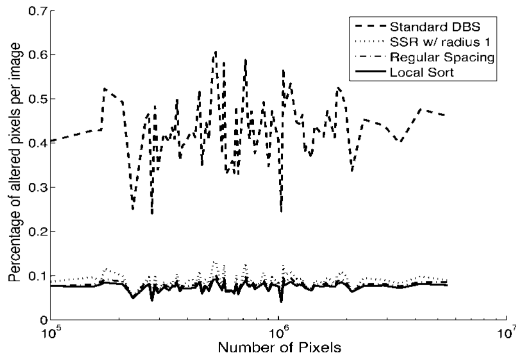

Figure 3. *Fraction of altered pixels per image as a function of image size.*

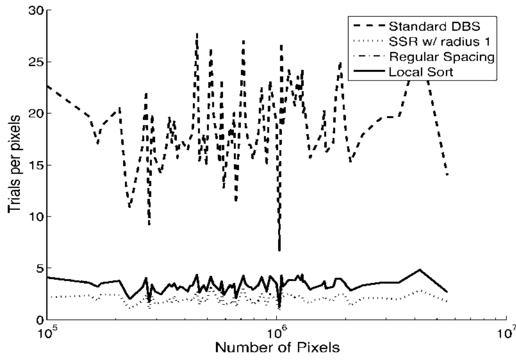

Figure 4. *Trials per pixel as a function of image size.*

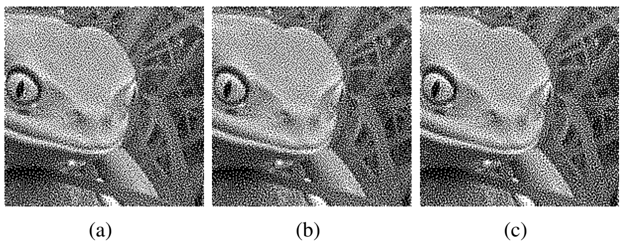

Figure 5. *Results of halftoning a grayscale image (Figure 1(a)) with the proposed schemes: (a) Local Sort, (b) Regular Spacing, and (c) SSR with radius one. In all of these schemes, Threshold Refinement is applied with $\beta = 0.5$. SSR with radius zero is also applied in schemes (a) and (b).*

quality of its halftones.

Future work will include more extensive testing of the methods, including optimization of algorithm parameters and incorporating techniques for reducing the number operations per trial as described in [1,3,8]. The goal is to design halftoning algorithms with halftone quality close to DBS and running times comparable to error diffusion.

Acknowledgment

This project is part of the Mathematical Modeling in Industry IX Workshop for Graduate Students organized by the Institute for Mathematics and Its Applications (IMA) at the University of Minnesota, August 1-10, 2005. The authors would like to thank the IMA for its generous funding, excellent computing facilities, stimulating environment and supportive personnel throughout the workshop.

References

[1] Jan P. Allebach. "DBS: retrospective and future directions", Proc. SPIE, vol. 4300, pp. 358–376, 2001.

[2] Kevin E. Spaulding, Rodney L. Miller, and Jay Schildkraut. "Methods for generating blue-noise dither matrices for digital halftoning", *Electronic Imaging*, **6** (2), pp. 208-230, 1997.

[3] C. W. Wu, G. Thompson, and M. Stanich, "A unified framework for digital halftoning and dither mask construction: variations on a theme and implementation issues," NIP 19: IS&T's International Conference on Digital Printing Technologies, pp. 793-796, 2003.

[4] Sang Ho Kim and Jan P. Allebach. "Impact of HVS Models on Model-based Halftoning", Proc. SPIE, vol. 4300, pp. 422-437, 2001.

[5] Robert W. Floyd and Louis Steinberg. *An Adaptive Algorithm for Spatial Grayscale*. Proceedings of the Society for Information Display **17** (2) pp. 75-77, 1976.

[6] J.F. Jarvis, C.N. Judice and W.H. Ninke, *A Survey of Techniques for the Display of Continuous Tone Pictures on Bilevel Displays*. Computer Graphics and Image Processing **5** pp. 13-40, 1976.

[7] P. Stucki, *MECCA - A Multiple Error Correcting Computation Algorithm for Bi-level Image Hard Copy Reproduction*. Research report RZ1060, IBM Research Laboratory, Zurich, Switzerland, 1981.

[8] C. W. Wu, M. Stanich, H. Li, Y. Qiao and L. Ernst *Fast Error Diffusion and Digital Halftoning Algorithms Using Look-Up Tables*, NIP22: IS& T's International Conference on Digital Printing Technologies, 2006.

Author Biography

C. W. Wu received his B. A. in Cognitive Science from Lehigh University, his M. A. in Mathematics and Ph. D. in Electrical Engineering from the University of California, Berkeley. He is currently a Research Staff Member at IBM T. J. Watson Research Center. He has authored over 100 papers, was issued over 40 US Patents and is a Fellow of the IEEE. His research interests include digital halftoning, multimedia security and synchronization in networks of nonlinear dynamical systems.