Reducing Ink Coverage in Binary CMYK Images

R. Victor Klassen
Xerox Webster Research Center, Webster, New York

Introduction

Thermal ink jet (TIJ) printers have a number of advantages over printers of other technologies, but also suffer certain drawbacks. Principal among the advantages are the ability to display single pixels, the saturated colours that are readily available with the right combinations of ink and paper, and the low cost. Water-based ink, essential to the process, also brings with it challenges in terms of safe levels of ink coverage. Because the ink is water-based, it tends to soak through the page and/or run down the page when heavy coverage levels are applied. At levels sufficiently low that the ink does not literally pour down the page, there is still a problem with ink of different colours diffusing rapidly across the inter-colour boundary. In their lesser extremes, the two main problems are inter-colour bleed, and blooming—ink of any colour bleeding into a region of white.

Here we shall discuss one possible approach to reducing problems related to excessive ink coverage: applying image processing techniques to the bitmap. This must be done in such a way as to avoid introducing any artifacts that make it obvious that tampering has been performed, and with very limited information about the desired appearance. Other approaches, such as changing the materials or print-head design, may in the long run prove better; in the short term, results may be obtained much faster using software.

Requirements

The application area puts some constraints on the design of a solution. Given that inkjet technology is inexpensive, it is likely that any processing might be embedded in a lowend machine, with limited processing power and a relatively low bandwidth connection to the (typically personal) computer that is generating its page images. Very little can safely be assumed about the page images except that they are the right size and binary.

The input must be binary unsegmented page buffers and the coverage of the output must be limited to some (as yet unknown) value. There are three other requirements worth noting: the overall hue of small regions (or large) should not change, regions that need not be changed should be left untouched, and finally, visible patterns should not be introduced into the image. It is assumed that the page has already had under colour removal applied[5], so that most, if not all of the three colour black pixels are gone.

Technically, ink coverage is the size of a single dot (one drop) divided by the area of a pixel (as given by the interpixel spacing), times the number of drops printed per pixel (up to four for a cmyk printer). For our purposes coverage is simply defined as the number of drops perpixel. Based on this definition, full binary undercolour removal brings coverage to 200% or less. For at least one materials combination a coverage level of below 160% is required for good quality.

Possible Alternatives

There is very little published about ways of handling these problems. One method, described by Chan et al.[3], is designed to reduce paper cockle and graininess of ink jet prints, using multiple dye loadings. Printing of a given scanting occurs multiple times, with three different dye loadings. Thus, those pixels requirmg the highest dye loading are printed on one pass, those requiring an intermediate dye loading are printed on another pass, and those requiring the lowest dye loading on another pass.

Chan's method takes as input contone RGB and performs RGB-CMYK conversion with full undercolour removal. Eight bit per colour input is converted to half resolution output with multiple gray shades available at each of the four output pixels corresponding to one input pixel. Redundant grey shades are used to maximize the ink coverage within the constraint that it must be less than some maximum, printer/ink/paper specific value. Standard error diffusions is used to reduce the initial 256 levels to the number of levels of grey obtainable using three dye loadings and up to four drops per loading, and also to handle the error introduced by reducing the grey level where there is not a suffficiently low coverage pattern at the required level.

Another method, described by Trask[7], limits coverage to 100%. The coverage is limited by using 2×2 super-pixels and assigning them one drop per pixel in a combination that depends on the colour required. Assuming one bit per separation input with full undercolour

Originally published in *Proc. of IS&T's 46th Annual Conference*, May 9-14, 1993, Cambridge, Massachusetts.

removal, there are eight possible colours that could be requested (including white).

Both of the above methods are primarily useful for pictorial images because of the use of superpixels. If applied to normal text or typical business graphics the artifacts produced by the use of superpixels would be unacceptable.

Perhaps the most attractive approach is to perform the contone to binary conversion in such a way as to keep coverage as close to optimum as possible, either through a clever halftoning scheme or modified error diffusion. (The Chan patent is one such approach). Such an approach presumes the pre-existence of a full colour continuous tone page image, which is beyond the scope of a very low end printer. The method described below is restricted to binary input, and presumes that no segmentation of the page has been performed.

A Coverage Reduction Technique

The algorithm that seems most natural, given the requirements above, is as follows:

For each pixel or small region,
1. Compute the coverage in the neighborhood of that small region
2. If the coverage is too high
   2.1. Selectively turn off some of the pixels in the region

At the single pixel level coverage can be 0%, 100%, 200%, 300% or 400%; if full undercolour removal is employed, coverages are only 0°/4 100% or 200%. In order to begin to consider intermediate coverage levels one must use a region of the image and average the coverage in some way. There is a tradeoff to be made here. Large regions allow high precision in measuring the coverage, but smaller regions tend to be important for intercolour bleed. Preliminary tests indicated an 8×8 window to be reasonable for finding the overall coverage inducing bleed in the 4×4 square in its center. This is by no means a carefully optimized value: it seemed sufficiently large for a good starting point, and small enough to be sensitive to variations in the density onthe page. It is also relatively efficient (the cost of filtering an image being proportional to the area of the footprint of the filter). The initial estimate has been sufficiently effective that further study seemed of little value.

Computing coverage in an 8×8 window, always aligned on a 4 bit boundary can be done efficiently using some small tables. Coverage should be computed for all of the separations and summed. Computing coverage in this way was described by Carpenter[2] in the context of computing approximate coverage of polygons in super-sampling grids for antialiasing purposes. It is possible that some colours bleed more than others; if so, some separations might be weighted more heavily in the sum.

Step 2.1, that of selectively turning off some of the pixels, is most important as far as meeting the requirements of preventing hue shifts. Pixels of different separations are turned off in the same proporation as they occur in the original image.

An algorithm much like error diffusion can be used to set the coverage to the desired level. Suppose all of the pixels in two separations are turned on in a subregion. Visit all of the pixels in turn, keeping an error variable for each separation. Turn on pixels that error diffusion would turn on in a region of constant (correct) coverage.

In the usual case there will be some coverage in each separation in a window. Assuming the total coverage is too high, it is necessary to turn off a fraction of the pixels in each separation, and the same fraction in all separations, except black. The fraction to be left on is simply the ratio of the desired coverage to the measured coverage. This can be exactly retained as a rational number, *d/m*. A simple algorithm for ensuring that *d* of every *m* pixels are turned on is:

```
f =-m
for each pixel that is on in the original
    f=f+d
    if (f>=0)
      turn on this pixel
      f=f-m
```

The fraction *f* is the (fractional) number of pixels that should have been turned on since the last one was turned on relative to m, or alternatively the fraction of the way to the next pixel that should be turned on. Pixels that are turned on are spaced as uniformly as possible along the (as yet unspecified) path that gives the order in which they are visited. If the control variable (*f*) is initialized only at the start of the page, and not in each window, the average over a larger region will be correct.

The initial setting of *f* to zero is arbitrary; to reduce intercolour correlations it may be advantageous to use various values between 0 and m for the initial settings of the control variable for the different separations.

The student of computer graphics will recognize Bresenham's algorithm for drawing lines[1]. It is also at the heart of any error diffusion algorithm.

We have not yet discussed the order in which pixels are visited. Some pixels should be visited first on one separation while others are visited first in another separation. This is to avoid biasing the results in favour of certain colours. Also, a good path has a high fraction of pixels close together on the page being close together on the path. Considering only four-connected neighbors, one can rank a few alternatives. If it is satisfactory to use a single 8×8 window to determine the coverage of an entire 4×4 region, it is desirable to visit all the pixels in that region before moving on to another such region. It saves buffer space, and it reduces the mean distance along the path to four-connected neighbors (relative to using scanting order). Figure 1 shows some alternative paths through a 4×4 window. Of these, (e) is the preferred option. It is derived from the Hilbert curve[6]. Similar paths have been used to minimize patterning in error diffusions. Besides having the highest fraction

four-connected neighbors close together on the curve, it has the added advantage of being the most irregular of the patterns, and therefore least likely to interact with halftone or error diffusion-induced patterns in the input image. From this standpoint, (b) and (d) are most likely to create visible patterns at certain intensities where one pixel in four is affected, with (b) being worse than (d).

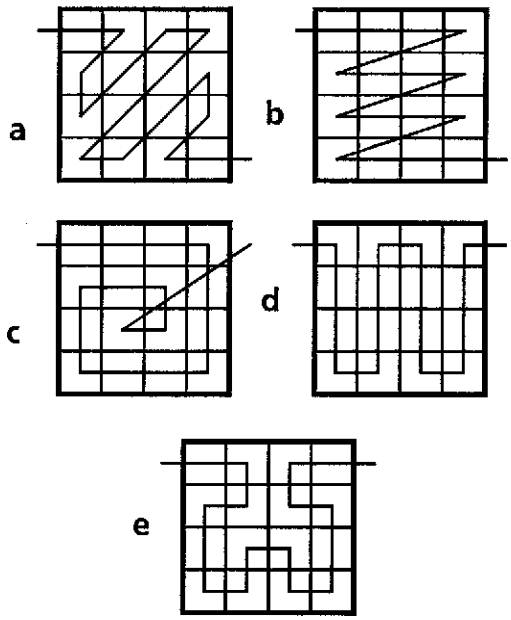

Figure 1. Five possible orders of visiting the pixels in a 4 × 4 square, ranked in order by decreasing mean distance to four-connnected neighbors. If eight-connected neighbors were counted, (a) would hae ranked higher in the list, but (e) would still be the winner.

In order to eliminate bias (from always considering one separation first), the pixels should be considered with respect to different separations in different orders. There are two ways of doing this. One way is to cycle through the possible orderings in which the separations can be treated, visiting pixels in the same order for each separation, the other is to change the order in which pixels are visited for each separation we have followed the second approach.

All of the patterns in Figure 1 can be flipped about the horizontal midline to produce a pattern with the same features. 90° rotations create discontinuities at the cell boundaries, but only there. Since black is highly unlikely to have any pixels removed (in areas with much black the coverage will be 100%) and yellow is unlikely to show small effects resulting from the discontinuities in the path, these are the best choices for the discontinuous patterns. Alternatively one might choose pattern (e) for magenta and cyan, and (d) for yellow and black, yielding four patterns without discontinuities. While perfectly good arguments exist for the various alternatives, the simple choice of rotating pattern (e) for the four separations has been used with sufficient success to obviate the need for further experimentation.

Summary

A technique has been described for reducing the amount of ink coverage in images destined for printing on TIJ printers. The method avoids hue shifts by removing the same fraction of ink in all separations. Acceptable regions are left unaffected, in two senses: large regions of low coverage are not changed in any way, and individual pixels with no more than one separation to be printed are not touched. After regions of excessive coverage are found, pixels in such regions are considered for deletion, in an order dictated by a small piece of a Hilbert's curve. This reduces the potential for introducing patterns, while at the same time spreading out the deleted pixels relatively uniformly over the region.

Simply reducing the maximum level of ink coverage in a print has led to substantial improvements in image quality. While it does not fix all problems, for some materials combinations it makes every page printable, with the intended appearance.

References

1. J. Bresenham 'Algorithm for computer control of a digital plotter', *IBM Systems Journal*, **4** (1), 1965, pp. 25-30.
2. L. Carpenter, 'The A-buffer, anantialiased hidden surface method', *Computer Graphics* **18** (3), 1984, pp. 103-108.
3. C. S. Chan, J. G. Bearss, T. M. Nelson, Method and system for enhancing the quality of both color and black and white images produced by ink jet printers, U.S. Patent 4, 930,018, May 1990.
4. R. W. Floyd, L. Steinberg "An adaptive algorithm for spa tial gray scale"", *Proc. Society for Information Display*, **17** (2) 1976.
5. See eg., J. D. Foley, A. vanDam, S. K Feiner, J. F. Hughes, *Computer Graphics Principles and Practice*, 2nd ed, Addison Wesley 1990, p. 599.
6. D. Hilbert Uber die stetige Abbildung einer Linie auf ein Flachenstuck, *Math. Annelen* **38**, 1891, pp. 459-460.
7. J. L. Trask, "Method for enhancing the uniformity and consistency of dot formation produced by color ink jet printing", U.S. Patent 4,999,646, March 1991.
8. I. H. Whitten and R. M. Neal, "Using Peano curves for bilevel display of continuous-tone images", *IEEE Computer Graphics and Applications*, **2**, (5), (1982), pp. 47-52.