January 2010

A joint color trapping strategy for raster images

Wang Haiyin

M. Boutin

J. Trask

J. P. Allebach

Haiyin, Wang; Boutin, M.; Trask, J.; and Allebach, J. P., "A joint color trapping strategy for raster images" (2010). *Department of Electrical and Computer Engineering Faculty Publications.* Paper 36.
http://dx.doi.org/http://dx.doi.org/10.1117/12.840426

A Joint Color Trapping Strategy for Raster Images

Haiyin Wang[a], Mireille Boutin[b], Jeffery Trask[c] and Jan P. Allebach[b]

[b]Qualcomm CDMA Technologies, 5775 Morehouse Drive, San Diego, CA 92121;
[b]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-0501;
[c]Hewlett-Packard Company, 11311 Chinden Blvd, Boise, ID 83714

ABSTRACT

Misalignment between the color planes used to print color images creates undesirable artifacts in printed images. Color trapping is a technique used to diminish these artifacts. It consists of creating small overlaps between the color planes, either at the page description language level or the rasterized image level. Existing color trapping algorithms for rasterized images trap pixels independently. Once a pixel is trapped, the next pixel is processed without making use of the information already acquired. We propose a more efficient strategy which makes use of this information. Our strategy is based on the observation of some important properties of color edges. Combined with any existing algorithm for trapping rasterized images, this strategy significantly reduces its complexity. We implement this strategy in combination with a previously proposed color trapping algorithm (WBTA08). Our numerical tests indicate an average reduction of close to 38% in the combined number of multiplications, additions, and "if" statements required to trap a page, as compared with WBTA08 by itself.

Keywords: Color trapping, color plane mis-registration, feature extraction, look-up table

1. INTRODUCTION

Multi-stage color printers, such as traditional color laser printers, operate by sequentially printing several image planes on a printing medium. For each electrostatic image plane, a different colorant (toner) is selected and then applied to the medium. Misalignment between the color planes creates gap and/or halo artifacts around the edges of each color plane. Color trapping is a process in which color edges are either expanded or shrunk to create an overlap of colors so that a small color plane misalignment (up to about 2 pixels) will not cause gap or halo artifacts. Trapping is commonly used in high-quality printing, for example in the publishing industry. Typically, this is done manually or interactively by a trained graphic artist using a professional printing application. This manual procedure is obviously tedious and difficult, especially for large scale publishing.

Some automated approaches for color trapping have previously been proposed. These can be divided into two categories based on the data representation method they use: either objects or pixels. In the case of object-based trapping methods,[1–4] the PDL instructions of the printed page are translated into a vector format suitable for detection and analysis of edges between different color regions in the page. Based on a set of trapping rules, which differ depending on which particular algorithm is being used, trapping decisions are then made at the color edges. Finally an output file containing the traps is generated in PDL format.

Pixel-based color trapping methods are applied to raster (bit-mapped) images The first pixel-based color trapping method was proposed by J. Trask[5] for hardware implementation. Building on this method, a lower-complexity method (WBTA08)[6] was suggested for software implementation. Both of these pixel-based color trapping methods tend to be less complex than object-based trapping methods, as they are applied directly on a raster (bit-mapped) image generated at a desired output resolution. The raster image is trapped in a local fashion based on the actual pixel data, which tends to yield a more straightforward algorithm. This requires each

Further author information: (Send correspondence to H.W.)
H.W.: E-mail: hw@purdue.edu, Telephone: 1 765 494 0751
M.B.: E-mail: mboutin@ecn.purdue.edu, Telephone: 1 765 494 3538
J.T.: E-mail: jeff.trask@hp.com, Telephone: 1 765 494 0751
J.P.A.: E-mail: allebach@ecn.purdue.edu, Telephone: 1 765 494 3535

Color Imaging XV: Displaying, Processing, Hardcopy, and Applications, edited by Reiner Eschbach, Gabriel G. Marcu, Shoji Tominaga, Alessandro Rizzi, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7528, 75280C · © 2010 SPIE-IS&T
CCC code: 0277-786X/10/$18 · doi: 10.1117/12.840426

color separation to be stored as an individual plane in a frame buffer. The planes of the frame buffer are then trapped pixel by pixel, and the results are used to determine respectively the final trapped colors for each plane. Since the interaction and the color of edges in the frame buffer can be figured out by the Raster Image Processor (RIP) interpreter, the color transitions are simple to find. However, both the above pixel-based methods trap the image pixels independently without considering the neighboring pixels.

We propose a color-trapping strategy which can be combined with any of the two existing pixel-based color trapping methods[5,6] to decrease its computational cost. This strategy makes use of the edge properties to trap the edge pixels more efficiently. We implement this strategy in combination with WBTA08. Our numerical tests indicate an average reduction of close to 38% in the combined number of multiplications, additions, and "if" statements required to trap a page, as compared with WBTA08 by itself. A non-optimized software implementation indicates an average running time reduction by about 11%.

The remainder of this paper is organized as follows. Sec. 2 summarize the color trapping strategy used in existing pixel-based color trapping methods. Our proposed strategy is presented in Sec. 3. Our experimental results are presented in Sec. 4 and we conclude in Sec. 5.

2. BACKGROUND ON RASTERIZED IMAGE TRAPPING

Existing color trapping algorithms for raster images (JTHBCT03 and WBTA08) process every pixel in the image one by one. To process a selected pixel, a window of print image data around the pixel is selected, and the data in the window is used to decide whether the selected pixel needs to be trapped and how. This is done in four steps.

Step One: Preprocessing. The color values of the pixels in the window are pre-processed to reduce storage requirements and ensure robustness of the process. The preprocessed window consists of pixels categorized into one of three colors (A, B and O). Color A represents all pixel colors that are close to that of the selected pixel.

Step Two: Edge Detection and Classification. The processed window data is analyzed to detect the presence of an edge in the neighborhood of the selected pixel. If no edge is detected, the selected pixel is categorized as *non-trappable*. If an edge is detected, the type of edge is determined. In most cases, two types are considered: 1) edges passing through the selected pixel, and 2) edges one pixel away from the selected pixel. The selected pixel is then classified as either *edge1*, *edge2*, or *non-trappable*, respectively.

Step Three: Trapping Parameter Calculation.

If the pixel is trappable (i.e. if it is categorized as either *edge1* or *edge2*), the trapping parameters are computed. This step involves computing the approximate color density of the original window and using this value to obtain the amount of trapping to be applied to the selected pixel.

Step Four: Calculation of Trapped Value for the Selected Pixel. The final color plane values of the selected pixel are calculated using the trapping parameters. The trapped value for each of the colors of the selected pixel are a linear interpolation of the corresponding two color values forming the edge. As a general rule, in order not to change the outline of the color object, the darker color that forms the contour should remain unchanged after trapping. As a result, the lighter color regions are usually extended into the darker ones.

3. JOINT COLOR TRAPPING IN THE NEIGHBORHOOD OF EDGES

Existing trapping methods for rasterized images trap every edge pixel independently: once an edge pixel is trapped, the next pixel is processed without making use of the edge type or trapping parameters of the previous pixel. Therefore, every pixel trapped goes through the same process described in Sec. 2. However, for the pixels around an edge, their edge types and trapping parameters are not independent of each other. Therefore, a computational advantage may be gained by trapping certain pixels together, instead of sequentially and independently. More precisely, the sequential edge type determination strategy can be replaced by a two-step strategy: 1) Determine the edge type of some pixels using an existing method, and 2) Deduce the edge type of the other pixels using some simple rules. Similarly, the sequential trapping parameter calculation strategy can be replaced by a two-step strategy: 1) Determine the trapping parameters for some edge pixels using an existing approach, and 2) Deduce the trapping parameters for the other edge pixels using some simple rules. In order to implement these strategies, we first study some important edge properties.

Figure 1. The edge map of an image. Black pixels represents non-trappable pixels, white pixels represents *edge2* pixels, and gray pixels represents *edge1* pixels.

3.1 Edge Properties

(a) Edges are continuous.

Since edges correspond to the boundary of objects in an image, edge pixels tend to form lines of a similar edge type. Indeed, as illustrated in Fig. 1, *edge1* pixels form continuous line segments immediately next to the object, while *edge2* pixels also form continuous line segments that are contiguous to the *edge1* segments. Moreover, contiguous to an *edge2* line segment is typically a connected line segment of non-trappable pixels.

Edge pixels thus tend to form groups at the junction of two areas of different colors, and many of these groups form either a vertical or a horizontal line segment. They can therefore be easily traced by following the pixels of a raster image in either the vertical or the horizontal direction. The edge direction (either vertical or horizontal) can be inferred from the orientation of an *edge2* line segment, and the position of the edge with respect to the *edge2* line segment (either left/right of a vertical segment, or above/below a horizontal segment) can be inferred from the position of the non-trappable pixel line segment next to the *edge2* line segment.

(b) Edges have two sides.

As we just mentioned, each *edge2* pixel is typically adjacent to an *edge1* pixel immediately next to the boundary of an object. Since object boundaries have two sides, two *edge1* pixels are typically found at each side of the boundary next to each other (except when one of them is white), each connecting with an *edge2* pixel. This gives rise to an *edge2-edge1-edge1-edge2* (E2112) pattern across the edge, with a pair *edge2-edge1* on each side of the edge. We call this a *double-sided edge pattern*. Note that when there is an edge between a colored area and a white area, there is only a single-sided edge pattern, with *edge2-edge1* appearing on the non-white side of the edge, as white pixels are always non-trappable.

As can be observed from Fig. 2, when the edge types of the pixels are determined using the edge classification method of WBTA08,[6] most of the edges exhibit the E2112 pattern, either horizontally or vertically. More specifically, almost all vertical or horizontal edges of length greater than 1 that are separating two non-white areas exhibit the E2112 pattern. Therefore, once an *edge2* pixel is detected next to an edge, the edge type of the remaining pixels in the pattern can be automatically inferred. The only question left is how to determine the edge direction (either vertical or horizontal) and its relative location (left/right or above/below the *edge2* pixel) in order to finalize the pattern placement with respect to that first *edge2* pixel. The edge continuity properties discussed in the previous subsection allow us to answer that question: if a horizontal (respectively vertical) row of *edge2* pixels is detected, then the edge must be horizontal (respectively vertical). Meanwhile, the relative placement of the contiguous row of non-trappable pixels next to this row of *edge2* pixels determines the placement of the *E2112* pattern.

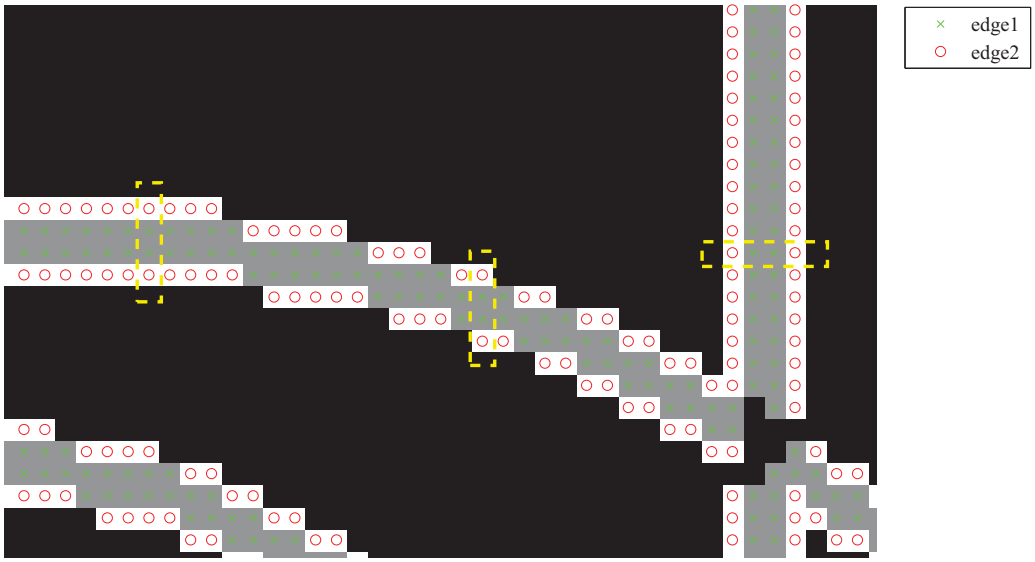

Figure 2. Close-up view of the *edge2-edge1-edge1-edge2* edge pattern. One can see that most of the edges give rise to this pattern.

(c) Edges are symmetric.

Edge features are geometric: their properties are independent of the actual position and orientation of the two regions they separate. The trapped values of the edge pixels interpolates the color values of these two regions. As a conclusion, this interpolation process should not depend on the position of the two regions. In particular, exchanging the two regions should not change the parameters of the traps. In other words, trapping on both sides of the edges should be a symmetric process.

However, this is not true for our previous three color trapping algorithms. Instead, following the approach of JTHBCT03, the trapped values of a selected edge pixel is a linear interpolation between the color values c_1 of the selected pixel (c_1 = color A) and the color values c_2 of the first color B pixel of the corresponding color-quantized 5×5 window. (See 1 and 2 in Fig. 3, respectively, for examples of color A and first color B pixels). More precisely, if $c_1 = (C_1, M_1, Y_1, K_1)$ and $c_2 = (C_2, M_2, Y_2, K_2)$, the trapping parameters δ_X, for each color $X = C, M, Y, K$, are obtained by using the values of c_1 and c_2 to index into a set of LUTs as described in.[5] The trapped values of the pixel are obtained by setting

$$X_1^{trapped} = X_1 + \delta_X(X_2 - X_1), \qquad \text{for } X = C, M, Y, K. \tag{1}$$

For brevity, we say that we *trap the color pixel* c_1 *according to* c_2 (or trap c_1/c_2). As the same canonical order is used to scan the pixels of the 5×5 window, the two *edge1* pixels facing each other across an edge often end up being trapped according to different color values. Consequently, the trapping parameters used for trapping the two sides of an edge are not always symmetric.

Instead of taking the color of the first color B pixel as the reference color c_2 to trap an *edge1* pixel, we propose to use the color of its direct neighbor across the edge (e.g., 3 in Fig. 3). When the edge separates two non-white regions, this corresponds to another *edge1* pixel. Moreover, for symmetry, we propose to trap both of these *edge1* pixels together as a pair, and their final trapping parameters (δ_X as in Eqn. 1) can be obtained from the same set of intermediate control parameters (see Subsec. 3.2.1 for details).

3.2 Details of Joint Color Trapping Algorithm

Our proposed algorithm is summarized in Fig. 4. We now describe it in detail.

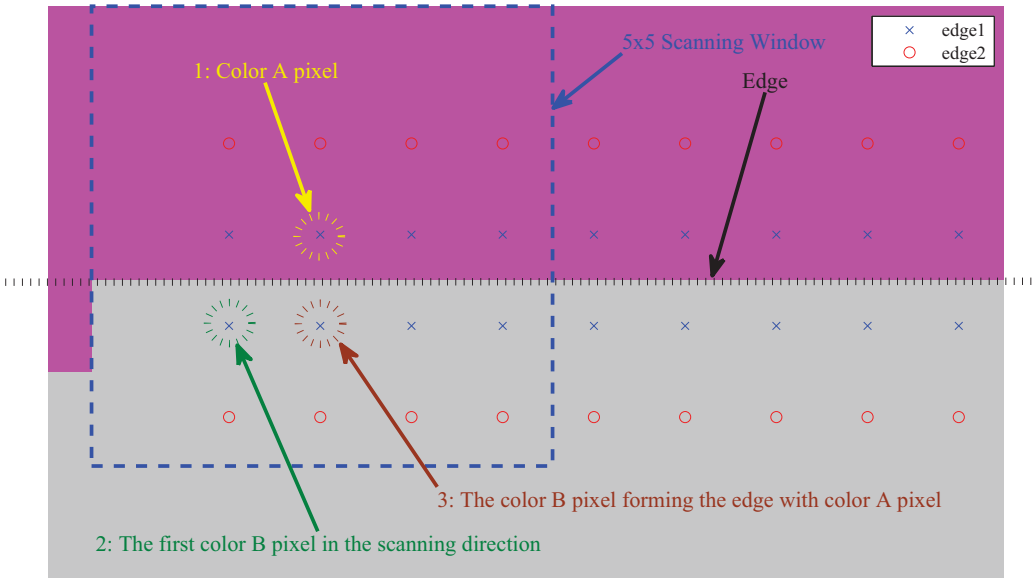

Figure 3. Color pixel arrangements around an edge.

3.2.1 Page Scanning Strategy

To facilitate the detection of edges, we use the following page scanning strategy as illustrated in Fig. 5. Given an input rasterized image, we begin by scanning it in raster order pixel by pixel using a 5×5 scanning window. Until an *edge2* pixel is found, every pixel is categorized and, if needed, trapped, using the method described in WBTA08. Meanwhile, an edge-filling map is maintained to keep track of the pixels being scanned and trapped.

Once a "fillable seed" *edge2* pixel (i.e., satisfying the edge properties of the first *edge2* pixels detected in a double-sided edge pattern) is identified, we start applying the edge type filling rules described in Subsec. 3.2.1 and the edge pixel trapping strategy described in Subsec. 3.2.1. If the presence of a horizontal edge is determined, the pixels directly below the *edge2* pixel in the next three rows end up being categorized and trapped. Similarly, if a vertical edge is detected, the pixels in the next three columns to the right of the *edge2* pixel end up being categorized and trapped. Meanwhile, the locations of all the pixels processed are recorded in the edge-filling map. We then continue scanning the image in raster order. Each pixel that is already recorded in the edge-filling map is skipped, as it has already been scanned and/or trapped.

(a) Edge-type filling rules Based on the previously described edge properties, we define the following edge-type filling rules.

(i) Horizontal edge-filling rule Given an *edge2* pixel that is not already part of an edge pattern (what we call an *isolated* edge pixel), we determine whether this pixel is next to a horizontal edge by checking if it is connected to other *edge2* pixels. If there are more than one *edge2* pixels connected horizontally with all non-trappable pixels above them, we conclude that an horizontal edge has been found. We thus fill the *E2112* pattern vertically starting from the first isolated *edge2* pixel and trap the four pixels included in the edge pattern according to the trapping strategy described in the next subsection. This process is illustrated in Fig. 6.

(ii) Vertical edge-filling rule If no horizontal edge is found, we search for vertical edges that are not part of the other edge patterns. That is, we look for a set of vertically connected *edge2* pixels with a contiguous column of non-trappable pixels on its left. When such a pixel configuration is found, we assume there is a vertical edge to the right of the *edge2* pixel row, and deduce the edge types of the rest of the *E2112* patterns. For trapping these pixels, we proceed vertically, starting from the first isolated *edge2* pixel and according to the trapping rules describe in the next subsection. This vertical edge filling process is illustrated in Fig. 7.

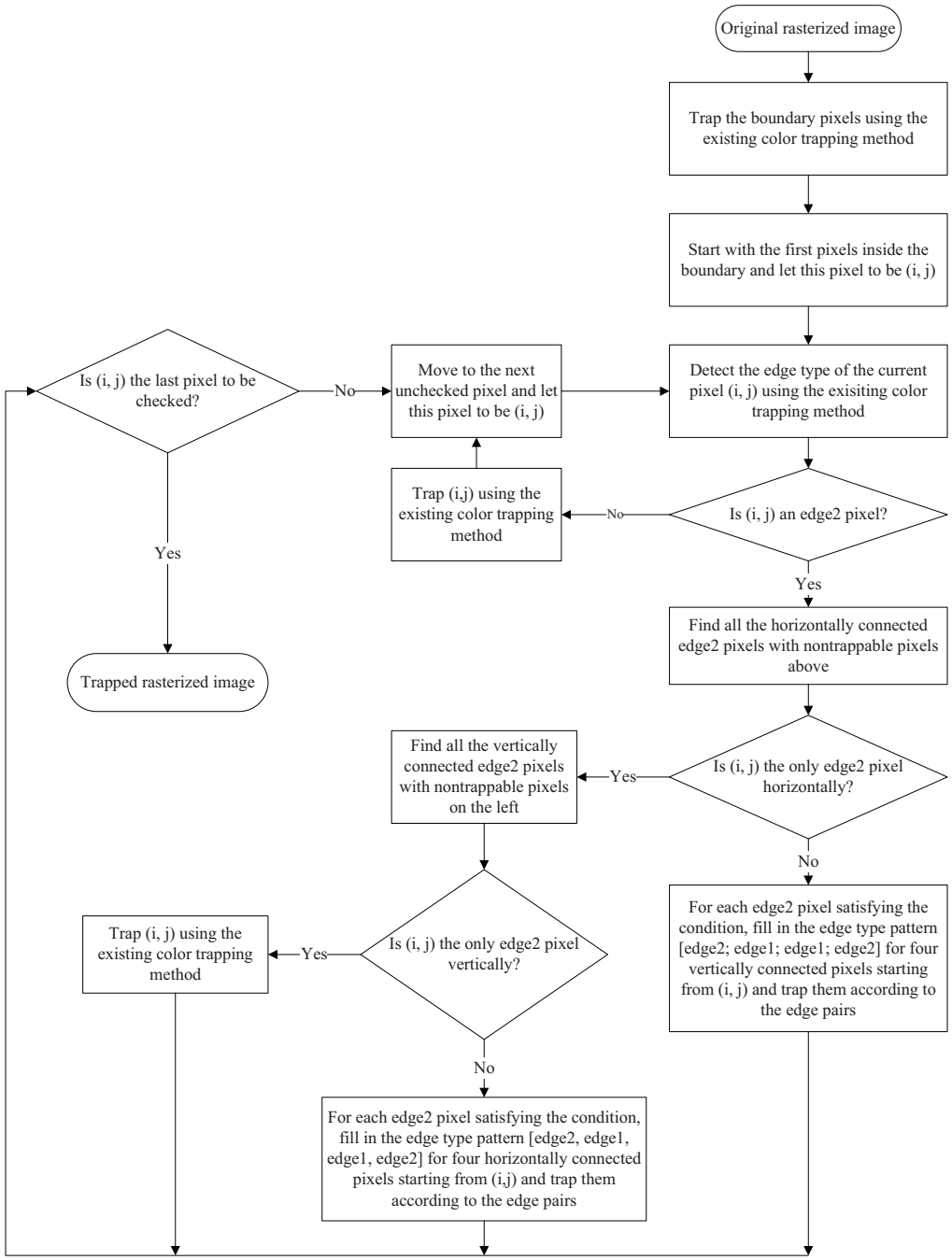

Figure 4. Flow diagram of the proposed Joint Color Trapping Strategy.

(b) Trapping of pixels surrounding a detected vertical/horizontal edge The pixels along detected vertical/horizontal edges form the aforementioned *E2112* pattern. We now describe our strategy to trap these pixels more efficiently. Let c_1 be the color of the selected edge pixel: this color is assumed to represent the color of one of the regions next to the edge. In order to determine the trapping parameters, we need to determine the color c_2 of the other region. This is done according to the following rules (Fig. 6):

* For the first pixel (*edge2*), c_2 is the color of the third pixel (*edge1*).

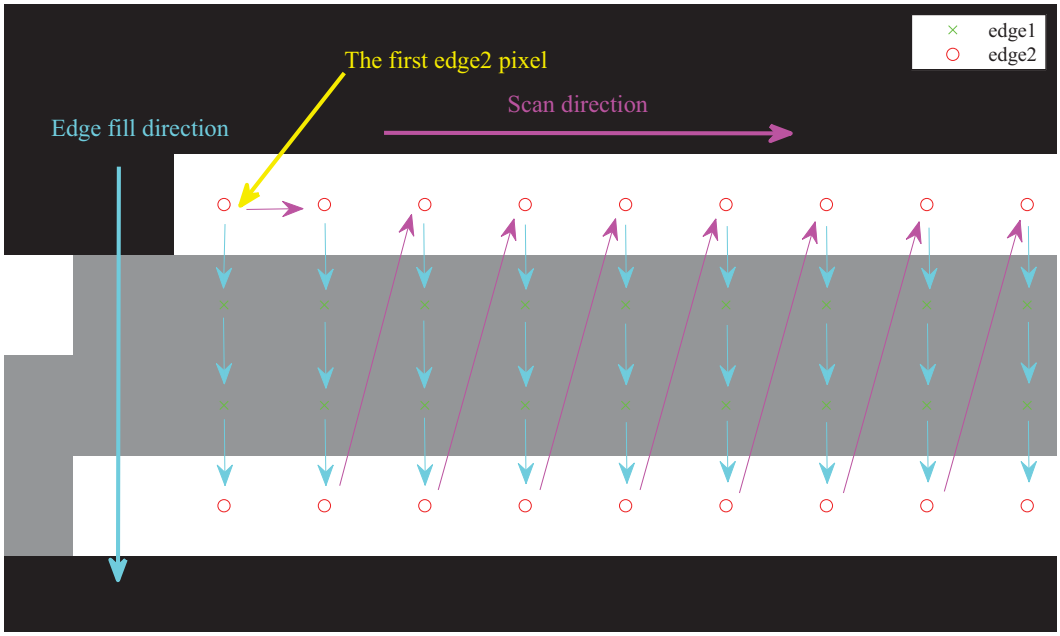

Figure 5. Illustration of edge type filling process around a horizontal edge.

* For the second pixel (*edge1*), c_2 is the color of the third pixel (*edge1*), and vice versa.

* For the last pixel (*edge2*), c_2 is the color of the second pixel (*edge1*).

By applying the trapping rules described above, we deduce directly the edge types and the colors of pixels c_1 and c_2 for these pixels, which are needed in order to obtain the trapping parameters. This enables us to skip the steps of bit truncation, color categorization and edge feature extraction which, in our previous three algorithms,

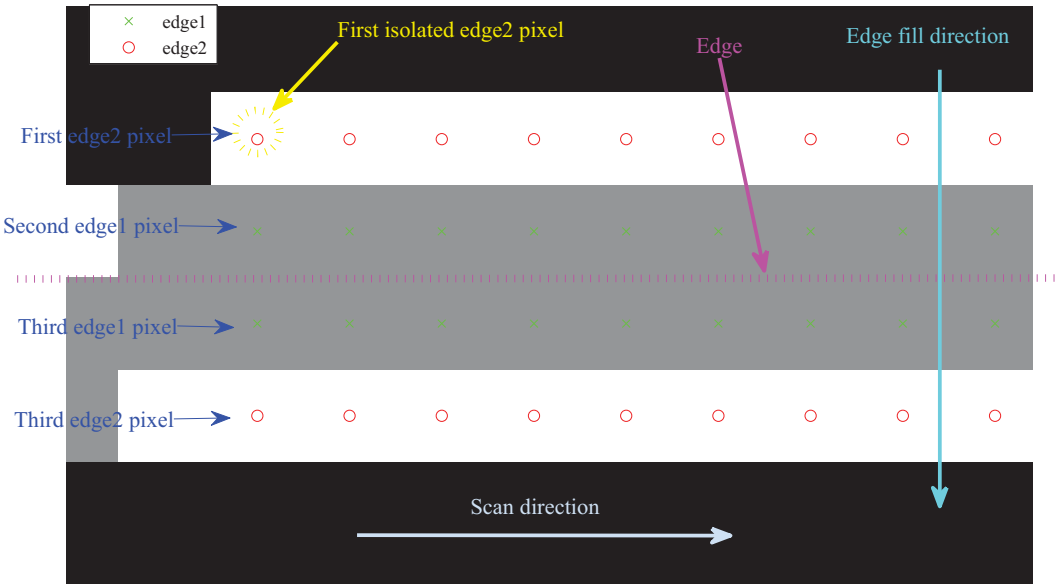

Figure 6. Horizontal edge trapping strategy.

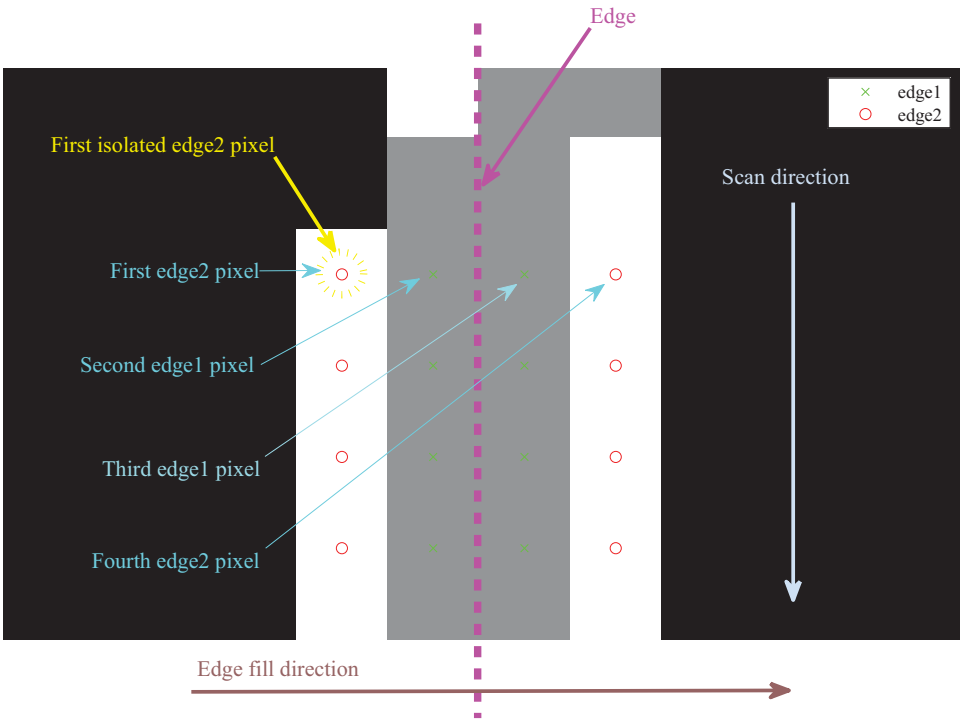

Figure 7. An example of a vertical edge.

are needed to determine the edge type of the selected pixel and to figure out the color of the first color B pixel (previously used in place of c_2).

(i) Trapping of *edge*1 pixels within an *E2112* pattern We trap the two *edge1* pixels in an *E2112* pattern as a group in order to save computations. There is a total of 16 different trapping cases, which are summarized in Table 1. We now describe how each case is trapped.

The first case occurs when neither of the two pixels to be trapped is white. Since the two pixels considered have the same edge type, and since their reference colors c_1 and c_2 are the reverse of each other, we are in a symmetric situation that can be exploited. More precisely, the two pixels can be trapped as a pair according to the equations

$$X_{c_1}^{trapped} = X_{c_1} + \delta_{X,c_1} \times (X_{c_1} - X_{c_2}), \qquad \text{for } X = C, M, Y, K, \tag{2}$$

$$X_{c_2}^{trapped} = X_{c_2} - \delta_{X,c_2} \times (X_{c_2} - X_{c_1}), \qquad \text{for } X = C, M, Y, K. \tag{3}$$

For $i = 1, 2$, the parameters δ_{X,c_i} are the same for $X = C, M, Y$ but different for $X = K$. This is due to the fact that K appears visually darker than the other three types of color when they have the same toner levels.

Thus one can write

$$(\delta_{X,c_1}, \delta_{X,c_2}) = f(X_{c_1}, X_{c_2}), \qquad \text{for } X = C, M, Y, \tag{4}$$

$$(\delta_{X,c_1}, \delta_{X,c_2}) = g(X_{c_1}, X_{c_2}), \qquad \text{for } X = K. \tag{5}$$

In other words, we only need two function calls (represented by a set of LUTs for each function call) and four δ_X's to trap two non-white *edge1* pixels together.

Table 1. Trapping decisions for color pixels c_1 and c_2 next to the edge

			Pixel c_1			
			Trapped Before		Not Yet Trapped	
			White	Non-White	White	Non-White
Pixel c_2	Trapped Before	White	-	-	-	Trap c_1/W[a]
		Non-White	-	-	-	Trap c_1/c_2[b]
	Not Yet Trapped	White	-	-	-	Trap c_1/W
		Non-White	Trap c_2/W[c]	Trap c_2/c_1[d]	Trap c_2/W	Trap c_1,c_2[e]

[a]Trap color pixel c_1 according to white
[b]Trap color pixel c_1 according to color pixel c_2
[c]Trap color pixel c_2 according to white
[d]Trap color pixel c_2 according to color pixel c_1
[e]Trap color pixels c_1 and c_2 together

Now if one of the two pixel colors, say c_2, is white, then that pixel does not need to be trapped. In that case, the amount of computation (i.e., the number of LUT accesses and "if" condition checks) can be further reduced, as (5) then becomes

$$\delta_{X,c_1} = \begin{cases} f(X_{c_1}, 0) & \text{if } X = C, M, Y, \\ 0 & \text{if } X = K, \end{cases} \tag{6}$$

Notice that, because of our proposed page scanning strategy, the *edge1* pixels considered may have already been trapped previously. More precisely, the *E2112* pattern considered may appear as part of a detected horizontal edge, but it may also lie at the junction of a previously detected vertical edge. In order to save computation, we thus check whether the pixel has already been trapped before attempting to trap it. This check is performed in conjunction with the white pixel check, and the trapping decision indicated in Table 1 is then made.

(ii) Trapping of *edge*2 pixels within an *E2112* pattern We now consider the trapping of the two *edge2* pixels of an *E2112* pattern. Unlike the *edge1* pixels discussed previously, these *edge2* pixels cannot be trapped in pairs, as their second reference color c_2 is not symmetric to them (i.e., c_1 is not a reference color of c_2 and they have different edge types.). Thus the two *edge2* pixels are to be trapped one by one.

Again, we note that white pixels, as well as previously trapped pixels, should not be trapped. Therefore, we start by checking if the given pixel is white or previously trapped before the calculating the trapping parameters.

Since in this case, the selected pixel c_1 and its reference color pixel c_2 are of different edge types, the set of intermediate trapping parameters is different from that for trapping two *edge1* pixels together. With this taken in consideration, for each *edge2* pixel to be trapped, we then proceed to obtain its trapping parameters δ_{X,c_1}, which can be written as

$$\delta_{X,c_1} = f'(X_{c_1}, X_{c_2}), \qquad \text{for } X = C, M, Y, \tag{7}$$
$$\delta_{X,c_1} = g'(X_{c_1}, X_{c_2}), \qquad \text{for } X = K. \tag{8}$$

However, if color c_2 is white, the equations can be further simplified to

$$\delta_{X,c_1} = \begin{cases} f'(X_{c_1}, 0) & \text{if } X = C, M, Y, \\ 0 & \text{if } X = K, \end{cases} \tag{9}$$

4. NUMERICAL EXPERIMENTS AND DISCUSSION

The proposed trapping strategy can be combined with any color trapping method that processes subsequent pixels independently in order to make it more efficient. To illustrate this, we implemented this strategy with an existing algorithm, namely WBTA09. The computational complexity and overall quality of the trapping results are discussed below.

4.1 Computational Assessment

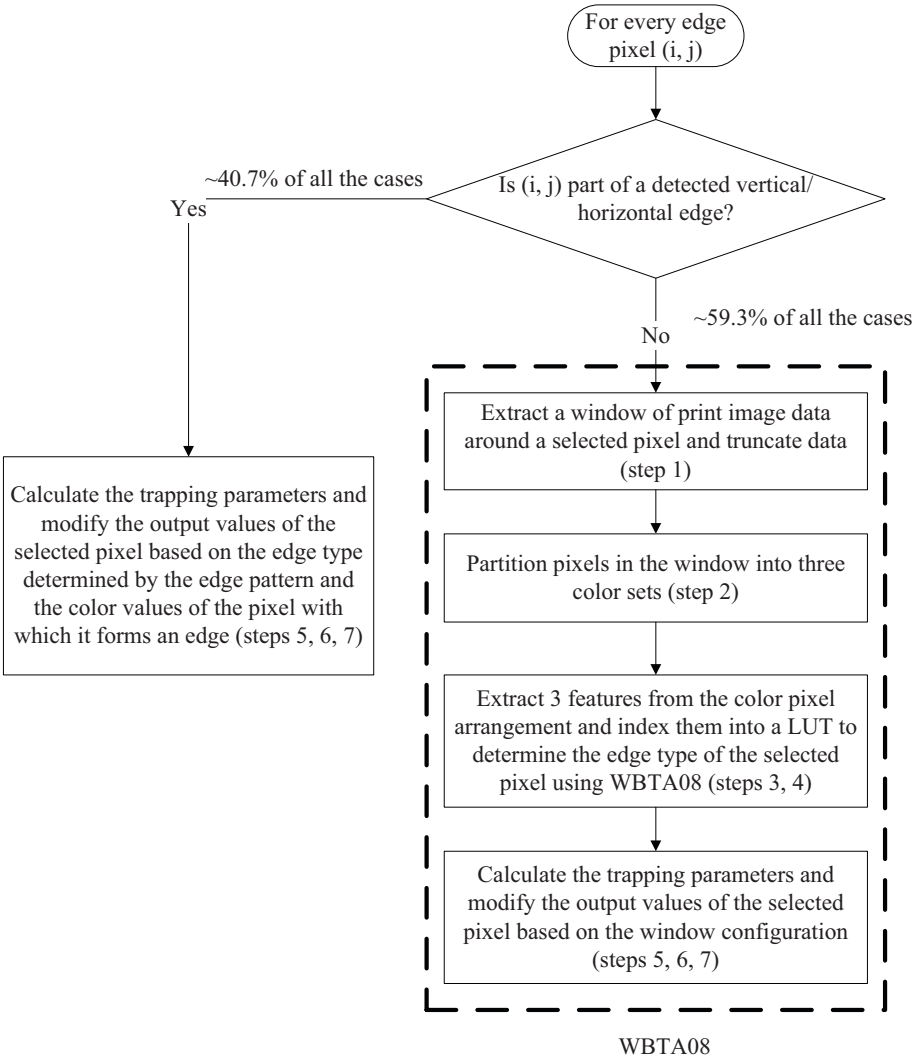

Figure 8. Computational savings analysis of the proposed Joint Color Trapping Strategy.

As illustrated in Fig. 8, our proposed joint-trapping algorithm incurs extra computation through the control logic used to detect the presence of vertical and horizontal edges, as well as the pixel map updates and comparisons. But this extra burden is small in comparison to the cost of running WBTA08 for trapping a pixel. Indeed, the added computational cost is offset by the fact that about 40.70% of the edge pixels (according to our test set of 73 images) are not processed using the method in WBTA08. For most of these edge pixels, the bit truncation, color categorization, feature extraction and edge detection steps are thus bypassed. Meanwhile, for all of these, the trapping parameter computation and the trapped value computations are greatly simplified. The complexity of trapping a pixel of color c_1 according to a pixel of color c_2 is analyzed in Table 2. Since there is no multiplication involved in this method, only number of 'if' statements and number of additions are tabulated.

Table 2. Computations needed to trap a color c_1 *edge1* pixel according to the color values c_2 of the other *edge1* pixel in an *E2112* pattern

			Pixel c_1			
			T[a]		NT[b]	
			W[c]	NW[d]	W	NW
Pixel c_2	T	W	1 'if'		2 'if'	8 'if', 3 '+'
		NW	1 'if'		2 'if'	10 'if', $\leq$ 4 '+'
	NT	W	1 'if'		2 'if'	9 'if', 3 '+'
		NW	1 'if'		2 'if'	9 'if', $\leq$ 4 '+'

[a]Trapped before
[b]Not yet trapped
[c]White
[d]Non-White

Table 3. Computations needed to jointly trap a color c_1 *edge1* pixel and the color values c_2 of the other *edge1* pixel together in an *E2112* pattern

			Pixel A			
			T[a]		NT[b]	
			W[c]	NW[d]	W	NW
Pixel B	T	W	6 'if'			11 'if', $\leq$ 3 '+'
		NW	6 'if'			17 'if', $\leq$ 3 '+'
	NT	W	6 'if'			11 'if', $\leq$ 3 '+'
		NW	9 'if', $\leq$ 3 '+'	16 'if', $\leq$ 7 '+' 'if'	9 'if', $\leq$ 3 '+'	19 'if', $\leq$ 7 '+'

[a]Trapped
[b]Not Trapped
[c]White
[d]Non-White

When two *edge1* pixels across the edge from each other are trapped, their trapping parameters are obtained in a single process and they are trapped together. The complexity analysis for this joint trapping process is tabulated in Table 3. Although it seems that it is more complicated to trap c_1 and c_2 together, the computation time is saved by bypassing the steps required to determine their edge types and the corresponding reference color pixels (i.e., data extraction, bit truncation, color categorization, feature extraction and edge detection steps), which are more time-consuming.

Comparing to WBTA08, the Joint Color Trapping algorithm further simplifies the trapping process by applying the edge-filling rules to deduce the edge type of the pixels belonging to the *E2112* pattern and jointly trapping the two *edge1* pixels together in the pattern. As a result, the complexity is lowered down by effectively reducing the number of pixels entering all the steps required for WBTA08. The overall complexity analysis based on average number of operations executed per pixel is presented in Table 4. As can be seen, with only a little increase in accessing small LUTs (1.4% increase for accessing one-dimensional LUTs and 9.6% increase for accessing two-dimensional LUTs), the number of "if" statements and the number of additions are greatly reduced (29% drop in number of "if" statements and 43% drop in number of additions) as compared to WBTA08.

4.2 Overall accuracy rate

We tested the accuracy of our proposed Joint Trapping Algorithm as compared with WBTA08 on our previously described 73 image test set. Note that the two algorithms can differ in two ways: 1) the edge type assigned to the pixels may be the different, and 2) the color values of the trapped pixels may be different. Overall, we found that

Table 4. Overall complexity comparison between WBTA08 alone and with Joint Color Trapping Strategy

	WBTA08	WBTA08 with Joint Color Trapping Strategy
One-dimensional LUTs[a]	190.37	193.04
Two-dimensional LUTs[a]	46.39	50.85
Three-dimensional LUTs[a]	1.00	0.79
No. of "if" statements[a]	35.48	25.17
No. of additions[a]	55.02	31.20
No. of multiplications[a]	0	0

[a]The number of times accessed is calculated as an average per pixel over our 73 test image set. The maximum size used is 48×1 for one-dimensional LUTs, 9×256 for two-dimensional LUTs and $32 \times 32 \times 32$ for three-dimensional LUTs.

about 25.33% of the edge pixels are assigned a different edge type. However, only 0.078% of the total number of pixels (15.56% of total trapped pixels) end up being trapped differently (in terms of trapped color values) from WBTA08. Therefore, the visual appearance of the trapped images is highly similar for both methods.

5. CONCLUSION

Color trapping of rasterized images is currently performed sequentially in a way that treats each pixel independently. We proposed a different strategy that takes into account the previously processed neighboring pixels in order to save computation. Our strategy is based on the observation of two important edge properties: 1) edges are continuous, and 2) edges have two sides. We used these properties to replace the existing sequential processing strategy for the edge type determination and trapping parameter computation. More precisely, we determine the edge type/trapping parameters of groups of pixels in the neighborhood of edges following a two-step procedure: 1) Determine the edge type/trapping parameters of some pixels using an existing method, 2) Deduce the edge type/trapping parameters of the other pixels using some simple rules. Computation is reduced for each pixel for which the edge type and/or trapping parameter are deduced.

We implement the proposed strategy in combination with an existing color trapping algorithm (WBTA08). Our numerical tests indicate an average reduction of close to 38% in the combined number of multiplications, additions, and "if" statements required to trap a page, as compared with WBTA08 by itself. The computational advantage of our strategy is more pronounced when the image contains many horizontal and/or vertical edges. In general, the resulting trapped image is very similar to the same image trapped by WBTA08.

ACKNOWLEDGMENTS

This work was supported by the Hewlett-Packard Company. We thank Peter Majewicz for providing some of the test images.

REFERENCES

[1] Bloomberg, S. J., "Electronic trapping system for digitized text and images," (1994). US 5,581,667.
[2] Weinholz, P. and Funke, V., "Method for generating trapping contours in a print page," (2001). US 6,795,214.
[3] Klassen, R. V., "Methods for automatic trap selection for correcting for separation misregistration in color printing," (1999). US 6,345,117 B2.
[4] Morgana, S. C., "Methods and systems for detecting the edges of objects in raster images using diagonal edge detection," (1999). US 6,377,711 B1.
[5] Trask, J. L., "Trapping methods and arrangements for use in printing color images," (April 2003).
[6] Wang, H., Boutin, M., Trask, J., and Allebach, J., "An efficient method for color trapping," in [*Color Imaging XIII: Processing, Hardcopy, and Applications, IS&T/SPIE Electronic Imaging Symposium*], (Jan-Feb 2008).