Blocked Recursive Image Composition

C. Brian Atkins
Hewlett-Packard Laboratories
Palo Alto, CA USA
brian.atkins2@hp.com

ABSTRACT

Presentations that feature arrangements of photos (e.g., collages, photobooks and slideshows) are a popular means of sharing and communication. In the prevalent framework for creating such presentations, layouts are supplied to the user in the form of designed templates. However, a template library may not support a desired number of photos, combination of aspect ratios or range of areas. To alleviate this issue, we present BRIC, a method for arranging practically any number of photos on a rectangular canvas. Primary constraints ensure respect of photo aspect ratios and specified gutter thickness; while secondary constraints encourage layouts in which photo areas correspond to desired relative area values. Photos are arranged based on a recursive partition of the page, with photo areas computed as the solution to a linear system implied by the partition. We present a detailed description of BRIC along with examples that demonstrate its versatility.

Categories and Subject Descriptors

I.7.2 [**Document and Text Processing**]: Document Preparation – *Photocomposition/typesetting*

General Terms: Algorithms, Design

Keywords: album, automatic layout, collage, composition, photo layout

1. INTRODUCTION

With the proliferation of digital photography, presentations that feature arrangements of photos (e.g., collages, photobooks and slideshows) are emerging as a popular means of sharing and communication. At the same time, solutions for creating such presentations have become widespread. In the prevalent solution framework, layouts are supplied to the user in the form of designed templates with regions where photos can be placed. Such a framework can be straightforward for the developer to implement; flexible for the producer to maintain with new and different designs; and easy for the user to master.

However, the number of possible designed templates grows rapidly with the number of photos. For example, a complete set of templates that would accommodate 5 or more photos could be prohibitively expensive. Most solutions deal with this issue by providing only a limited sample. As a result, a given template library may not support a desired number of photos, combination

MM'08, October 26–31, 2008, Vancouver, British Columbia, Canada.

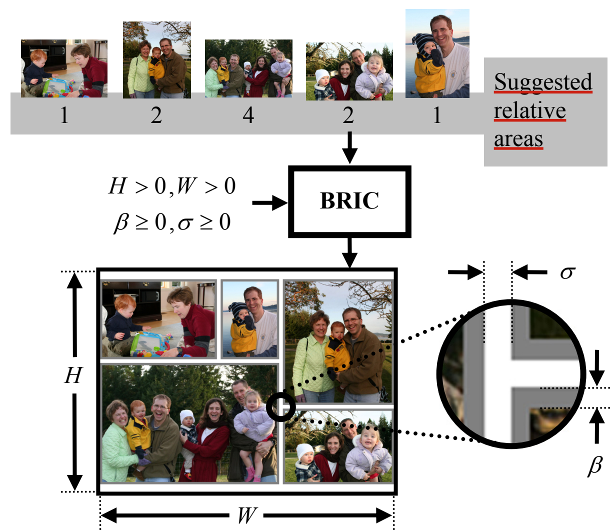

Figure 1. BRIC photo layout creation process. In the example, (H, W, β, σ) = (24, 32, 0.25, 0.40 inches). The canvas is outlined in black and borders are shown gray.

of aspect ratios or range of areas. In such situations the user is forced to compromise by excluding photos, or by placing photos in regions of inappropriate aspect ratio or area.

To alleviate this issue we present Blocked Recursive Image Composition, or BRIC, an algorithm for generating layouts. Here, *layout* refers only to the arrangement of photos. The overall idea is for the BRIC layout to be one layer in a composite with other elements such as background artwork and photo border treatments. To illustrate this, the results presented later show BRIC layouts as foreground, with drop shadows on plain white backgrounds.

BRIC arranges practically any number of photos on a canvas with arbitrary dimensions, subject to the following primary criteria:

(a) Photo aspect ratios are respected

(b) Photo borders and spacing between adjacent photo borders are precisely specified

Criterion (a) reflects the assumption that the layout process should not impose overlap or cropping of photographic content unless directed to do so. Criterion (b) reserves space for photo borders while protecting visibility of background and conferring visual uniformity [10]. We also include secondary criteria:

(c) Photo areas should be proportional to positive relative area values supplied with the photos

(d) Photos should occupy a maximum of the canvas area

We regard the primary criteria as "musts," and the secondary criteria as "high wants."

An overview of this method, including an example layout, is shown in Fig. 1. Note that because of the primary criteria, pairs of adjacent photos and blocks of photos have equal heights or widths. As long as the borders and spacing are not so thick that they take up the entire canvas in either dimension (a weak restriction), any set of photos will fit together in this way. However, we cannot guarantee the secondary criteria will be satisfied. Considering criterion (c), for example, notice that the photo with suggested relative area 4 does not have area exactly 4 times that of either photo with suggested relative area 1. Considering criterion (d), although all the photos are large enough to be easily visible, the layout does not completely fill the canvas in the vertical direction.

Prior work in automated layout dates at least as far back as early graphical computer interfaces. Many of those methods developed the notion that a layout can be conveniently characterized in terms of geometric forms, coupled with constraints on their spatial relationships [8]. A recent example is [6] which enables fast and space-efficient browsing of photo collections.

Other approaches have emphasized aesthetic presentation. For example, [5] includes a page layout module that uses a genetic algorithm to optimize aspects such as balance and symmetry. In [1] we presented a photo layout method where criteria (a) and (c) were primary, but criterion (b) was ignored (so layouts from [1] would not satisfy spacing and border specifications).

In related research, automatic layout is guided by content analysis. For example, in collages from [4], content is distributed spatially according to continua such as chronological order (e.g., movement along a contour over the layout might pass by photos in the order of a vacation time line). In [2], [3] and [9], photos are arranged so as to minimize occlusion of faces or other salient content. Since [4] and [2] report using previously designed templates to accomplish layout, it is possible they could use automatic methods such as that proposed here as a source of variation and flexibility.

In the remainder of this paper, Sec. 2 gives a detailed description of the proposed layout method. Sec. 3 presents some results, and Sec. 4 wraps up with discussion and conclusions. We do not address pagination or layout across page spreads in this paper.

2. LAYOUT METHOD

In this section, we address the problem of arranging $N \geq 1$ photos on a canvas having height $H > 0$ and width $W > 0$ subject to the primary and secondary criteria, denoting the border thickness as $\beta \geq 0$ and the spacing between adjacent borders as $\sigma \geq 0$.

The layout is encoded as a binary tree which leads to a recursive partition of the canvas as illustrated in Figure 2. In the tree, each terminal node (p_i for $1 \leq i \leq 5$) corresponds to a photo. Each interior node corresponds to a bounding box on the canvas, and its designation as a horizontal (H) or vertical (V) cut corresponds to dividing the box into two smaller boxes. This structural form has been used in other applications as discussed in [1].

In Sec. 2.1 we describe how an arbitrary tree having this form is mapped to a layout (that is, a precise position and area for each photo) that conforms to the primary criteria. Next, in Sec. 2.2 we describe how candidate trees are generated and scored. This tree construction process is responsible for finding a layout that conforms to the secondary criteria.

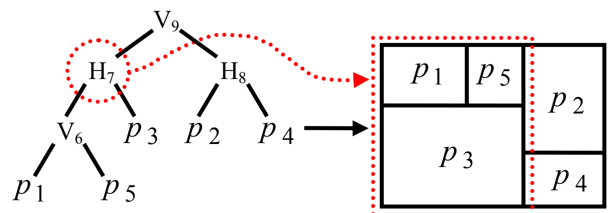

Figure 2. Layout is encoded as a binary tree that induces a recursive partition. Interior tree nodes correspond to bounding boxes on the canvas.

2.1 Mapping from tree structure to layout

To determine layout from tree structure, the essential problem is to compute photo areas subject to the primary criteria. Since criterion (a) fixes photo aspect ratios, it is sufficient to compute photo widths. To compute photo widths, we solve a system of N linear equations in N unknowns, where the variables are the photo widths.

The key to obtaining the equations is to characterize the bounding box dimensions using expressions that are linear in the photo widths. This can be done quickly by submitting the root node of the tree to the following recursion.

- If the current node is a terminal node, the bounding box height and width are $h_n + 2\beta$ and $w_n + 2\beta$ respectively
- If the current node is an interior node:
  - Submit left-hand child of current node to recursion
  - Submit right-hand child of current node to recursion
  - If current node is associated with a horizontal cut:
    - bounding box height is the sum of the bounding box heights of the two children plus σ
    - bounding box width is the bounding box width of either of the two children
  - If current node is associated with a vertical cut:
    - bounding box height is the bounding box height of either of the two children
    - bounding box width is the sum of the bounding box widths of the two children plus σ
- Return

Table 1 shows the bounding box dimensions for the partition in Fig. 2. Note that for $n \leq 5$, h_n and w_n denote respectively the height and width of a photo; but for $n > 5$, h_n and w_n denote the dimensions of a bounding box, including all borders and spacings contained therein. We define the aspect ratio as height divided by width, so the expressions in Table 1 are linear in the photo widths.

Table 1. Bounding box dimensions for partition in Fig. 2.

Node label	Bounding box height	Bounding box width
$n = 1, \ldots, 5$	$h_n + 2\beta$	$w_n + 2\beta$
6	$h_1 + 2\beta$	$w_1 + w_5 + \sigma + 4\beta$
7	$h_6 + h_3 + \sigma + 2\beta$	w_6
8	$h_2 + h_4 + \sigma + 4\beta$	$w_2 + 2\beta$
9	h_8	$w_7 + w_8 + \sigma$

We obtain one equation from each of the $(N-1)$ interior nodes. Specifically for each vertical (horizontal) interior node, we constrain the bounding boxes of its two children to have equal heights (widths). For example, in Fig. 2, the constraint furnished by node H_7 is $w_6 = w_3 + 2\beta$.

There are two possible N-th constraints: one equating the height of the root bounding box to the height of the canvas, and another for the width. Using one of these constraints will yield photo areas for which the root bounding box fits on the page, while using the other will yield a root bounding box that exceeds the canvas in the unconstrained dimension. To resolve this issue we solve two systems, one for each of the two possible constraints, and we pick the solution for which the root bounding box fits inside the canvas. As an example, a complete system for Fig. 2 is

$$\begin{bmatrix} a_1 & 0 & 0 & 0 & -a_5 \\ 1 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 \\ a_1 & -a_2 & a_3 & -a_4 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \begin{bmatrix} 0 \\ -(2\beta+\sigma) \\ 0 \\ 0 \\ W-(6\beta+2\sigma) \end{bmatrix}$$

where the bottom row constrains the width of the root bounding box to equal the width of the canvas.

It is possible for a tree to yield a solution vector with values that are not positive, suggesting that photos should have "negative areas". This happens when the fixed constraints due to borders and spacing leave no room for photos. In our implementation, trees with such solutions are thrown out.

2.2 Tree construction process

At its heart, this process is trial and error: candidate trees are generated and scored, and the tree with highest score is selected. Any of a number of stochastic processes could be used to generate candidates; however, they can be complex and computationally expensive, especially when the number of photos is large.

A simple deterministic search as used in [1] is to construct the tree directly by adding terminal nodes (or photos) one at a time. This yields a sequence of trees

$$T(1), T(2), \cdots, T(N)$$

where $T(k)$ is a tree with $k \geq 1$ terminal nodes, and the last tree is the result. In this simple search, each tree $T(k)$ is used to generate a set of candidate next trees, and the candidate next tree having the highest score is designated $T(k+1)$.

The approach used here is only slightly broader: at each stage, instead of carrying forward the best tree, we carry forward $L > 1$ trees having the L highest scores. By carrying forward trees that have *different* scores, we avoid loading the list with layouts that are equivalent (reflected versions of each other). A greater value of L increases the search at greater computational cost. We used $L = 4$ for the results shown here.

The purpose of the score is to guide the search, by selecting trees that match the secondary criteria. Our approach is to compute "ideal estimates" of the photo areas based on the suggested relative areas and canvas dimensions; then to compute a score that reflects how well the photo areas in the candidate layout match their respective estimated values.

Accordingly, a simple score for a tree with $k \geq 1$ terminal nodes is as follows. Denoting suggested relative areas as $e_1, \ldots, e_k$, ideal area estimates are computed

$$\hat{e}_i = HWe_i / \sum_{j=1}^{k} e_j \qquad (1 \leq i \leq k). \qquad (1)$$

Then denoting actual photo areas (from the method of Sec. 2.1) as $A_1, \ldots, A_k$, individual photo scores are computed

$$r_i = \min(A_i / \hat{e}_i, 1.0) \qquad (1 \leq i \leq k). \qquad (2)$$

Essentially, the individual photo score is penalized if the photo is smaller than its estimated ideal area, but not if the photo is larger. The simple score is the sum of the individual photo scores:

$$\sum_{j=1}^{k} r_j \quad . \qquad (3)$$

For the results presented here, we used a slightly more complicated score. First, we modified (1) to take into account the area that would be occupied by borders and spacing. Second, we modified (2) to penalize the individual photo score further if the actual area was less than half the estimated ideal area. Finally, we modified (3) to apply a global penalty whose magnitude increased with the number of photos having individual photo scores less than 0.25. These penalties were useful in helping to avoid layouts with excessively small photos.

3. RESULTS AND EXTENSIONS

Figure 3 presents BRIC layouts using three different sets of canvas dimensions, with all other variables (including the photos, their relative areas and the order in which they were added to layouts) held constant. Note how the layout changes according to canvas dimensions.

Figure 4 shows BRIC layouts using different sets of relative areas, with all other variables (including the photos and the canvas dimensions) held constant. These results are indicative of our broader experience: relative area values do exert some influence over actual photo areas, but the spatial constraints imposed by criteria (a) and (b) and the canvas dimensions can limit the level of influence. Of course, the number of photos on the canvas also affects control over relative areas.

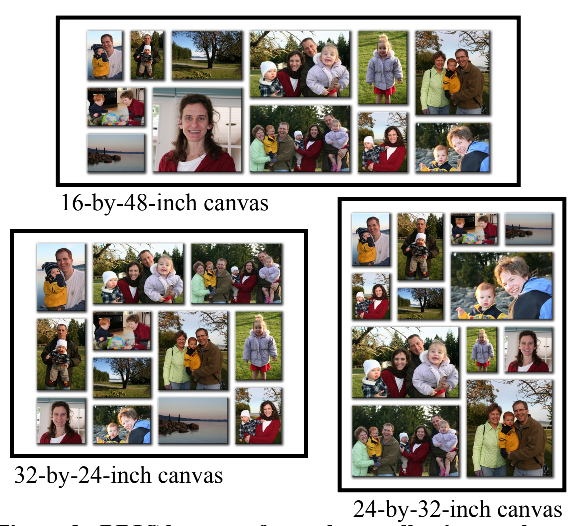

Figure 3. BRIC layouts of one photo collection on three different canvases, with (β,σ) = (0.25, 0.40) inches.

Figure 4. BRIC layouts from different relative area assignments: default 1.0; red 6.0; blue 3.0; green 0.125.

Figure 5. A photo-video layout including 7 photos and one set of 6 keyframes extracted from a video.

A generalization of BRIC is to replace the notion of a photo with a "graphic assembly" or group of photos that can appear in any of multiple alternate arrangements. Each arrangement is characterized as a tree structure, and the purpose of the search is to find not only a position for this graphic assembly, but also to select an arrangement. The scoring function is as described above, with each keyframe having its own suggested relative area. Although this increases the computational cost, it also enhances spatial flexibility because the arrangements have different shapes. Figure 5 includes 7 photos and one graphic assembly with 6 keyframes extracted from a video using [12]. The graphic assembly had 4 possible arrangements: 1x6, 2x3, 3x2 and 6x1. Each arrangement was a filled set of rows and columns with lexicographical order following chronological order. In addition, the keyframes were separated by a smaller spacing distance than elsewhere in the layout, to visually suggest that the keyframes are a unified group.

4. CONCLUSIONS AND DISCUSSION

In this paper, we have presented BRIC, an automatic photo layout method that arranges virtually any number of photos on a canvas having any dimensions. The layout is encoded as a binary tree in which interior nodes correspond to nested bounding boxes (hence the word *recursive* in the title); and the mapping from binary tree to page layout explicitly respects photo aspect ratios, as well as borders around photos and spacing between adjacent photos (hence the word *blocked*). The tree itself is constructed using a deterministic search in which photos are added to the layout one at a time.

We presented results that demonstrate how BRIC layouts depend upon the shape of the canvas, and how BRIC provides influence over photo areas. We also demonstrated how BRIC reserves space for rendering photo border artwork, and for uniform separation. Finally, we presented a novel photo-video layout in which videos are represented by automatically extracted keyframes.

In contrast to some prior automatic layout algorithms, BRIC determines a layout without any occlusion or cropping of photographic content. On the other hand, if it is acceptable to crop photos, then we can think of aspect ratios as variables that provide additional control over the distribution of photo areas, and the overall shape of the layout. For example, as described in [11] any photo can be grown or shrunk by a precisely defined amount by manipulating the aspects of all the other photos.

BRIC is quite computationally tractable: all photo composites presented here were generated in well under a second. In fact, our implementation is fast enough to act as "layout engine" in a mixed-initiative collage authoring experience [11]. For larger numbers of photos BRIC can be computationally expensive. In our implementation, generating a page with more than 20 or 25 photos can result in a runtime of multiple seconds. One way to alleviate this is to avoid evaluating equivalent candidate trees (that is, candidate trees related through a sequence of reflections about cuts) in the search.

We have found that the order in which photos are added to the layout affects the result. It would be interesting to investigate whether an optimal ordering can be found in reasonable time for an arbitrary set of (aspect ratio, relative area) pairs. We have run studies to evaluate whether users find BRIC layouts to be aesthetically pleasing, with the result that users were generally satisfied [11]. Further incorporation of metrics based on visual design principles [7][10] could be a promising path forward.

5. ACKNOWLEDGMENT

The author would like to thank Daniel R. Tretter of HP Labs for many useful discussions.

6. REFERENCES

[1] C. B. Atkins. Adaptive photo collection page layout. *ICIP'04*, 2004.

[2] J. C. Chen, W. T. Chu, J. H. Kuo, C. Y. Weng and J. L. Wu. Tiling slideshow. *ACMMM'06,* 2006.

[3] P. Chiu and A. Girgensohn. Stained Glass Photo Collages. *UIST'04*, 2004.

[4] N. Diakopoulos and I. Essa. Mediating photo collage authoring. *UIST'05*, 2005.

[5] J. Geigel and A. Loui. Using Genetic Algorithms for Album Page Layouts. *IEEE Multimedia,* 10(4), 2003.

[6] J. Kustanowitz and B. Shneiderman. Hierarchical Layouts for Photo Libraries. *IEEE Multimedia*, 13(4), 2006.

[7] S. Lok, S. Feiner, and G. Ngai. Evaluation of Visual Balance for Automated Layout. *IUI'04*, 2004.

[8] S. Lok and S. Feiner. A Survey of Automated Layout Techniques for Information Presentations. *Proc. of the SmartGraphics Symposium*, 2004.

[9] C. Rother, L. Bordeaux, Y. Hamadi and A. Blake. Autocollage. *SIGGRAPH'06,* 2006.

[10] R. Williams. *The Non-Designer's Design Book,* Second Edition, Peachpit Press, Berkeley, CA, 2004.

[11] J. Xiao, X. Zhang, Y. Gao, P. Cheatle and C. B. Atkins. Mixed-Initiative Photo Collage Authoring. *ACMMM'08,* 2008.

[12] T. Zhang. Intelligent keyframe extraction for video printing. *Proc. SPIE Conference on Internet Multimedia Management Systems*, 2004.