ADAPTIVE PHOTO COLLECTION PAGE LAYOUT

C. Brian Atkins

Imaging Technology Department
HP Labs
Palo Alto, CA 94304
cbatkins@hpl.hp.com

ABSTRACT

This paper presents a new photo collection page layout that attempts to maximize page coverage without having photos overlap. Layout is based on a hierarchical partition of the page, which provides explicit control over the aspect ratios and relative areas of the photos. We present an efficient method for finding a hierarchical partition that produces a photo arrangement suitable for the shape of the page. Rather than relying on a stochastic search we employ a deterministic procedure that mimics the natural process of adding photos to the layout one by one.

1. INTRODUCTION AND PRIOR WORK

This paper addresses the scenario where a user would like to print or display a photo collection, without the benefit of any time (or without the inclination) to arrange the photos for viewing in the situation at hand. For example, supposing the user had a photo collection accessible from a server, they might at different times like to show it displayed on a small monitor; projected onto a large wall; printed on small booklet pages; and printed on a poster. In each different situation, any number of factors influences how the collection should be arranged, such as the size and shape of the viewable area; the viewing distance; the time available to show the photos; and the cost to produce each page of output. These facts point to a need for methods that (1) make it easy to customize how a photo collection is presented, and (2) generate layouts dynamically to make the best use of available space. In this report, we present an adaptive photo collection page layout that attempts to meet both criteria.

We accomplish page layout using a "slicing structure," or a hierarchical partition of the page as illustrated in Fig. 1, where the terminal nodes correspond to photos and the interior nodes correspond to horizontal ("H") and vertical ("V") divisions. This basic form has been used in other fields including page layout for mixed documents [1], data visualization [2] and VLSI floorplan design [3]. One focus of this work was to use the slicing structure in a way that allows photos to be large without overlapping, while providing explicit control over photo aspect ratios and relative areas. Another focus was to determine the slicing structure itself, so that the layout takes into account not only the photos, but also the shape of the page.

In our experience with album and slideshow software, the most common page layout technique is to use templates, where photos are inserted into designated regions or "holes." Templates have the drawback that if the aspect ratios of photo and hole are not equal, then the photo must be cropped or shrunk to be visible through the hole. Moreover, new templates need to be added for every additional page shape, and for each new number of photos per page.

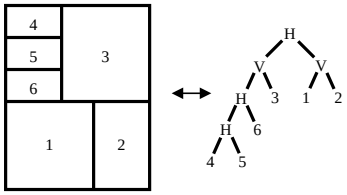

Fig. 1. Page layout is guided by a hierarchical page partition.

One prior approach, due to Geigel and Loui [4], accomplishes album page layout using genetic algorithms. This prior approach shares similarity with the present approach in that both search to find a layout having a highest score. However, [4] and the present approach are different in how page layouts are represented internally. In [4], page layout is specified with independent size and position parameters for each photo. In the present approach, page layout is specified using a tree structure that unifies how photos on the same page are placed and sized, which makes the search and scoring processes more tractable and intuitive.

This paper focuses strictly on page layout: given a collection of photos assigned to a single page, we present a method to determine their precise areas and positions. We have also investigated pagination, as well as optimization of layouts having the form described here, and we hope to report the results separately. The remainder of this paper is organized as follows. Section 2 provides a detailed description of our page layout algorithm. Section 3 shows some experimental results, and in Sec. 4 we wrap up with some discussion and conclusions.

2. PHOTO COLLECTION PAGE LAYOUT

Prior approaches to layout [3, 4, 1] have employed stochastic optimization to find a layout having a highest fitness or score. Our approach, by contrast, was to see whether something "good enough" could be obtained using a simpler, deterministic tactic.

The layout score we use is to calculate coverage, or the fraction of the page occupied by photo. More sophisticated layout scores would be based on user preferences and visual factors like symmetry, unused space and object alignment. However, we feel the results obtained using this simple metric are very encouraging.

Since one goal of this work is to allow the output form factor to be arbitrary, it would be useless for the user to specify absolute photo sizes. To address this, we replace the notion of *absolute*

photo area with *relative* photo area. Put more precisely, we assume each photo has assigned to it a positive scalar-valued *relative area proportion*. Denoting the relative area of a photo as e and the area of the rendered output photo as A, for any two photos i and j on the same page, the ratio of relative areas equals the ratio of rendered areas:

$$\frac{A_i}{e_i} = \frac{A_j}{e_j} \ .$$

Of course, if the user is not inclined to set a relative area for each photo, then numerical values can be assigned.

In this paper, we regard photo aspect ratios as fixed. This reflects the assumption that the user has already framed or cropped photos to their satisfaction, any any further cropping would not be desirable.

As mentioned above, our method amounts to the construction of a certain kind of tree structure, and this description is divided into two parts. Section 2.1 describes how to obtain the tree; and Sec. 2.2 describes how to render an output page from a tree (*i.e.*, compute a location and an absolute area for each photo).

2.1. Generating the binary tree

We generate the binary tree by adding photos one at a time, as illustrated in Fig. 2. We will fill in more details about the tree structure as we go along; but for now, it is sufficient to know that each node corresponds to a box in the layout: each interior node represents a bounding box around the boxes of its children; and each terminal node represents a cell where a photo is placed.

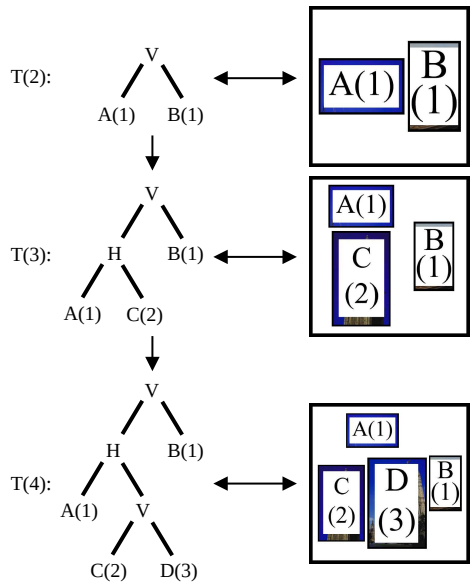

Fig. 2. Page layout generation process. Numbers in parentheses are relative areas.

We start with a page layout having only one photo, and we add photos one at a time until all photos are included. Formally, denoting the total number of photos as N, the page layout is determined as the last in an increasing sequence of binary trees

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

where $T(k)$ for $k \geq 1$ denotes a tree with k terminal nodes. Notice that each of the intermediate trees $\{T(k) : 1 \leq k \leq N-1\}$ generates a viable layout.

The way we add the next photo to the page is to introduce a new cell in the previous layout. Formally, this is accomplished by augmenting the previous tree. As illustrated in the figure, we select a subtree, and then displace the subtree with a new internal node. The new internal node becomes the parent of (a) a new terminal node corresponding to the new cell, and (b) the subtree that was displaced. (In the figure, the "selected subtrees" happen to be terminal nodes; but generally, any subtree could be selected, including subtrees rooted at internal nodes.)

To choose which cell which we introduce in the previous layout, we evaluate a collection of candidate layouts, where each candidate layout is the previous layout with the next photo inserted into a different new cell, and we select the cell for which the resulting layout has a highest score. To put this formally, assume that the k–th intermediate layout, represented with the tree $T(k)$, has a set $\mathcal{L}$ of possible new cells. We will write the tree structure $T(k)$ augmented by adding new cell $l \in \mathcal{L}$ as $T(k; l)$. Then the next intermediate layout is determined as

$$T(k+1) = T(k; L)$$

where

$$L = \arg\max_{l \in \mathcal{L}} \left[\text{score}\left(T(k; l)\right)\right]$$

and where $\text{score}(\cdot)$ is the scoring function mentioned above.

Notice that the cost of adding each next photo to the layout increases linearly with the number of photos. To be more precise, for intermediate tree $T(k)$, $\|\mathcal{L}\| = 2(2k-1)$, since there are $(2k-1)$ nodes in $T(k)$ and the new cell can be positioned either vertically or horizontally relative to the box of the displaced subtree. However, pages tend to have sufficiently few photos that the computational cost of evaluating all possible candidate layouts is not excessive.

2.2. Rendering the output page from the binary tree

The input to this process, illustrated in Fig. 3, is a tree structure where each terminal node has an aspect ratio and a relative area proportion, and each interior node indicates either a horizontal or vertical division. The first step is to characterize bounding boxes for the nodes, and the second step is to allocate a precise region of page space to each node. These regions are nested like the tree structure. We will refer to the regions allocated to terminal nodes as *cells*. Once a cell is known, the position of the photo assigned to it is determined by centering the photo in the cell. The area of the photo is figured using a formula given below.

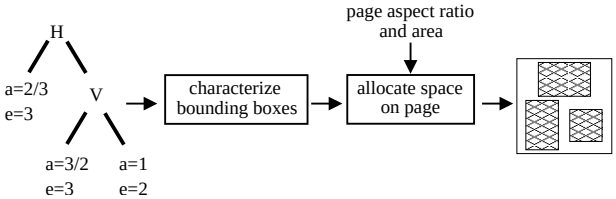

Fig. 3. How to render an output page from a tree structure.

2.2.1. Characterizing bounding boxes

The objective of this step is to compute an aspect ratio and a relative area for each node in the tree. In this paper, we regard relative areas and aspect ratios of photos as fixed, so the bounding boxes of terminal nodes remain unchanged. Each bounding box is a function of the boxes it encloses, so this process starts at the terminal nodes and works toward the root, following a depth-first search as illustrated in Fig. 4.

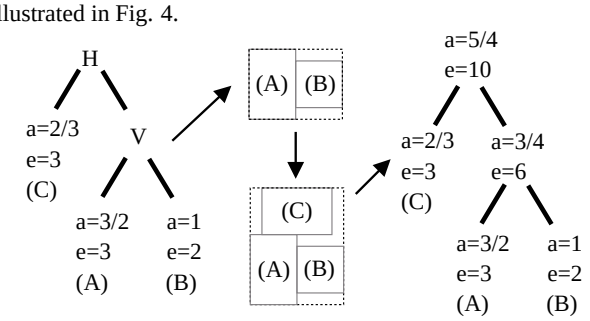

Fig. 4. Computing bounding boxes.

The aspect ratio a and relative area proportion e for any interior node are determined from the aspects and relative areas of its two children. Denote as (a_r, e_r) the aspect and relative area of the right-hand child, and (a_l, e_l) for the left. Then if the two children represent boxes arranged side by side,

$$a = \frac{\sqrt{a'e'}}{\sqrt{\frac{e_l}{a_l}} + \sqrt{\frac{e_r}{a_r}}} \quad (1)$$

$$e = \sqrt{a'e'}\left(\sqrt{\frac{e_l}{a_l}} + \sqrt{\frac{e_r}{a_r}}\right) \quad (2)$$

where

$$\sqrt{a'e'} = \max_{i\in\{r,l\}} [\sqrt{a_i e_i}] \ . \quad (3)$$

On the other hand, if the two children children represent boxes arranged one on top of the other,

$$a = \frac{\sqrt{a_l e_l} + \sqrt{a_r e_r}}{\sqrt{\frac{e'}{a'}}}$$

$$e = \sqrt{\frac{e'}{a'}}\,(\sqrt{a_l e_l} + \sqrt{a_r e_r})$$

where

$$\sqrt{\frac{e'}{a'}} = \max_{i\in\{r,l\}} \left[\sqrt{\frac{e_i}{a_i}}\right] \ . \quad (4)$$

To interpret these formulas, it is useful to think of the quantities $\sqrt{ae}$ and $\sqrt{\frac{e}{a}}$ as "relative height" and "relative width," respectively, in the same way that e is a relative area. For example, the aspect ratio in (1) is the ratio of a relative height divided by the sum of two relative widths, and the relative area in (2) is the product of the same. Finding the maximum in (3) determines which box is taller, and therefore governs the height of the box. An analogous discussion holds for the case of vertically arranged boxes, where (4) determines which box is wider, and therefore governs the width of the box.

The bounding box of the root is particularly important because it conveys the shape and relative area of the entire layout. We refer to it as the *principal bounding box*.

Since relative areas are consistent throughout the tree, we can score a layout as soon as the bounding boxes are known. For a tree T, the fraction of the page covered by photo is

$$\text{score(T)} = \mathcal{G}\frac{\sum_{i=1}^{N} e_i}{e_{\text{pbb}}}\frac{\min\{a_{\text{pbb}}, a_{\text{page}}\}}{\max\{a_{\text{pbb}}, a_{\text{page}}\}}$$

where $\mathcal{G}$ is a positive constant described below; N is the number of photos; e_i is the relative area of photo i; e_{pbb} and a_{pbb} are the relative area and aspect of the principal bounding box; and a_{page} is the aspect of the page.

2.2.2. Allocating regions of space to nodes

This process, which is illustrated in Fig. 5, has two steps. The first step is to divide the page into cells; and the second is to position each photo in its cell.

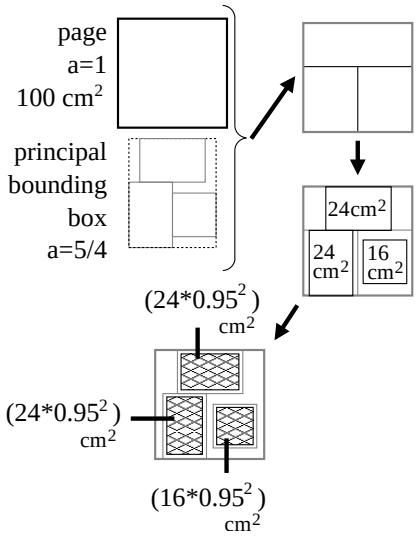

Fig. 5. Allocating space on the page.

To divide the page into cells, we split rectangles as illustrated in Fig. 1, starting with the entire usable area of the page as the first rectangle. Each split is accomplished by drawing a line segment for a corresponding interior node. Starting at the root, we work down, following the order of a breadth-first search.

In the case of an interior node corresponding to a vertical division, the task is to select a horizontal position along the width of the available region. Denoting the position with a scalar $x \in (0, 1)$, where $x = 0$ would represent the leftmost position and $x = 1$ the rightmost, we compute

$$x = \frac{\sqrt{\frac{e_l}{a_l}}}{\sqrt{\frac{e_l}{a_l}} + \sqrt{\frac{e_r}{a_r}}}$$

where a_l, e_l and a_r, e_r are the aspects and relative areas of the bounding boxes for the left and right children of the interior node. Notice this formula uses the relative widths of the bounding boxes of the children of the interior node as proportions. The analogous

formula for a horizontal division uses the relative heights. Denoting the position with a scalar $y \in (0,1)$, where $y = 0$ would represent the bottom position and $y = 1$ the top, we compute

$$y = \frac{\sqrt{e_b a_b}}{\sqrt{e_b a_b} + \sqrt{e_t a_t}}$$

where a_b, e_b and a_t, e_t are respectively the aspect and relative area of the bounding boxes for the bottom and top children of the interior node.

The area A_i for photo i is computed from the relative area of the photo assigned to it, as follows:

$$A_i = \mathcal{G} \frac{e_i}{e_{\text{pbb}}} A_{\text{pbb}} \ ,$$

where A_{pbb} is an area for the principal bounding box, computed

$$A_{\text{pbb}} = A_{\text{page}} \frac{\min\{a_{\text{pbb}}, a_{\text{page}}\}}{\max\{a_{\text{pbb}}, a_{\text{page}}\}}$$

with A_{page} the area of the usable page space. $\mathcal{G}$ is a scalar used to shrink each photo from the maximum area it could have as dictated by parameters of the tree, Setting $\mathcal{G}$ to a value between 0 and 1 has the effect of creating a buffer of space around the photo. We used $\mathcal{G} = 0.95^2$.

3. RESULTS

Figure 6 presents three different layouts of the same collection of eight photos. For the first layout, we divided the photos into two groups of four, and we used a page aspect ratio of $8\frac{1}{2} \times 11$. For the second layout, we used a square page to simulate printing on an album page or a CD case insert; and for the third, we used an aspect of 2×5 to simulate printing a large banner. Notice that because of the hierarchical scheme described above, the photos are arranged efficiently; for example, the short, wide photo at bottom of the first $8\frac{1}{2} \times 11$ page is situated low, affording greater space for the other three photos to be rendered larger. Also notice that on any given page in this figure, all photos have the same area, regardless of their aspect ratios. This is a deliberate result of the fact that we fixed the photo relative areas to be equal.

See Fig. 7 for three layouts that have the same photos, but different assignments of relative areas. For example, in the layout at left, the area of photo A is twice that of photo B, and $\frac{2}{3}$ the area of photo D.

4. DISCUSSION AND CONCLUSIONS

A number of factors that could contribute to a more complex scoring function (*e.g.*, aspect ratios and relative areas) are not allowed to vary. Photos are not allowed to overlap, and they are not rotated. Allowing any of these attributes to change while developing improved score functions may improve our results. In particular, notice that we could allow the a's and e's of terminal nodes to vary within tolerances. Allowing e's to change would affect the area ratios, and allowing a's to vary would require cropping photos. As long as the tolerances are not excessive, this may afford far greater layout scores without adversely affecting perceived layout quality.

We have assumed photos may be scaled to arbitrary rendered area. This means we assume there is sufficient pixel resolution to guarantee satisfactory image quality, and we ignore restrictions around rendered pixels per inch. Empirically, this has not been a problem in our experiments with 3-, 4- and 5-megapixel photos rendered on typical displays and $8\frac{1}{2} \times 11$ printed pages.

This work has presented a fast method for arranging a group of photos on a page. We have found this is a reliable method to generate a satisfactory layout quickly. However, many layouts appear to leave some room for improvement, and one of our next steps for this work is to explore ways these layouts can be optimized.

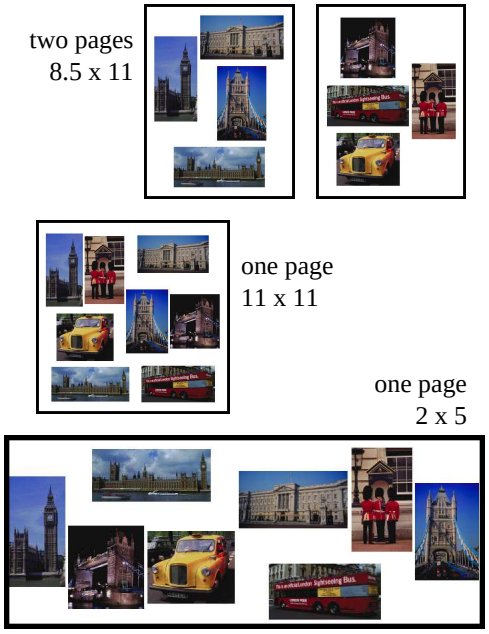

Fig. 6. Three different layouts of the same eight-photo collection. For each page, the relative areas of the photos were all equal.

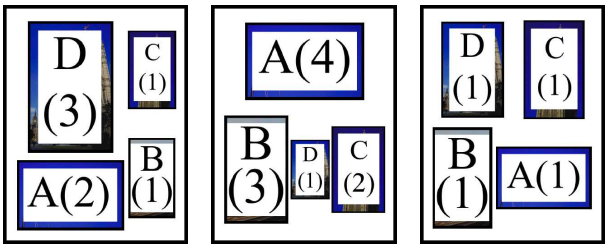

Fig. 7. Layouts of the same four photos, with different assignments of relative areas.

5. REFERENCES

[1] E. Goldenberg, "Automatic Layout of Variable-Content Print Data," M.S. thesis, University of Sussex, Brighton, UK, 2002.

[2] B. Shneiderman, "Tree Visualization with Tree-Maps: 2-d Space-Filling Approach," *ACM Transactions on Graphics*, vol. 11, no. 1, pp. 92–99, January 1992.

[3] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design," in *23rd Design Automation Conference*, Las Vegas, NV, 1986, IEEE, pp. 101–107.

[4] J. Geigel and A. Loui, "Using Genetic Algorithms for Album Page Layouts," *IEEE Multimedia*, vol. 10, no. 4, October–December 2003.