

Revised Aggregation-tree Used in Metadata Extraction from SVG Images

Shuju Bai
 Dept. of Computer Science
 Southern University
 Baton Rouge, LA, U.S.A.
 1-225-771-2060
bais@cmps.subr.edu

Abdus Salam
 Dept. of Computer Science
 Southern University
 Baton Rouge, LA, U.S.A.
 1-225-771-2060
salam@cmps.subr.edu

Ebrahim Khosravi
 Dept. of Computer Science
 Southern University
 Baton Rouge, LA, U.S.A.
 1-225-771-2060
khosravi@cmps.subr.edu

ABSTRACT

Many of the multimedia researchers have focused on the issue of the retrieval of images using indexed image collections. A number of spatial data structures based on Minimum Bounding Rectangles (MBRs) have been developed. Previously, we presented aggregation-tree structure to extract metadata from Scalable Vector Graphics (SVG) documents [8]. In this paper, we revised the aggregation-tree to better model the data in SVG and reduce the size of the aggregation-tree. Aggregation-tree is constructed based on aggregation rules for different topological relations between objects.

Keywords

SVG image, Aggregation-tree, Object aggregation, Relation.

1. INTRODUCTION

Scalable Vector Graphics (SVG) is an XML-based language for drawing two-dimensional graphics. It obtains invariant resolution, compactness, faster downloads, high compatibility, integration, and dynamic interactivity. However, information retrieval from the SVG documents is weak, which restricts the use of SVG. Thus, it is vital to develop a mechanism that extracts metadata from SVG documents. By the nature of graphical data, it is not easy to use tag matching technique to retrieve the desired data. This shortcoming motivates us to create a method for describing the contents of SVG images.

Many of the multimedia researchers have focused on the issue of the retrieval of images using indexed image collections. One of the most important methods for differentiating among the objects is the perception of the objects and spatial relationship that exists between them. "Directional Relations" are not sufficient for characterizing spatial similarity because they consider only the spatial orientation of an object while ignoring its spatial extent [1]. Gudivada's framework for retrieving images by spatial similarity defines an algorithm that utilizes both the directional and topological relations for quantifying similarity between images, retrieves similar

images and recognizes images even after they undergo translation, scaling and rotation [2].

A number of spatial data structures have been developed based on minimum bounding rectangle (MBR). The most promising group includes R-trees [3] and their variations [4, 5, 6]. The R-trees are direct extensions of B-trees in k-dimensions. The MBRs of the actual data objects are assumed to be stored in the leaf nodes of the tree. Intermediate nodes are built by grouping all rectangles at the lower level. An intermediate node is associated with some rectangle which encloses all rectangles that correspond to lower level nodes. Each pair of nodes may satisfy any of the topological relations [7].

Previously, we presented an aggregation-tree structure to extract metadata from SVG [8]. In this paper, we will continue to focus on the retrieval of topological relations. The aggregation functions are revised to better model the data in SVG. We will use the same terms as in [8]. Standard shapes are predefined in SVG. The topological relationships between two region objects used include the following eight: disjoint, meet, overlap, covered_by, covers, inside, contains, and equal (Figure 1) [7].

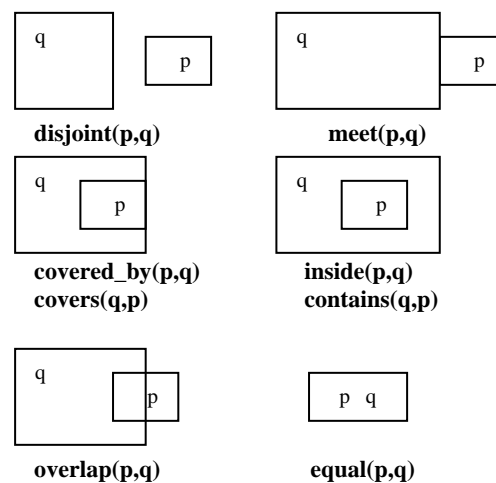
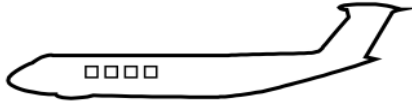


Figure 1. Topological relations [7]

2. PRELIMINARY

SVG files define images. The <path>, <circle>, <rect>, and <ellipse> elements draw the outline of any arbitrary shape by specifying a series of connected lines, arcs, and curves. This outline can be filled and drawn with a stroke. An example is shown in Figure 2 where the SVG document describes an airplane. The element 'path' specifies the outline of the airplane; the four 'rect' elements specify the four windows on the airplane.



```
<svg width="2.7720in" height="2.4807in" viewBox="0 0 1099 271">
<path style="fill:none;stroke:#000;stroke-width:10" d="M850,140 10,0 -
1, ..., 12 -7,8 -2,4z" />
<rect style="fill:none;stroke:#000;stroke-width:5" x="249" y="190"
width="30" height="30" />
<rect style="fill:none;stroke:#000;stroke-width:5" x="299" y="190"
width="30" height="30" />
<rect style="fill:none;stroke:#000;stroke-
width:5"x="349" y="190" width="30" height="30" />
<rect style="fill:none;stroke:#000;stroke-width:5"
x="399" y="190" width="30" height="30" />
</svg>
```

Figure 2. An example of SVG images and documents

3. AGGREGATION-TREE

3.1 Definition

Node: A node in an Aggregation-tree represents a region or a set of regions of an image. It is also called object.

Minimum Bounding Rectangle (MBR): An MBR $[(x_b, y_b), (x_u, y_u)]$ is the smallest rectangle which represents an object on a two dimensional plane.

3.2 Object identification

The primitive objects can be identified based on the predefined standard shapes in SVG image files. 'rect', 'circle', 'ellipse', 'line', 'polyline', and 'polygon' are tags used in SVG to represent graphic shapes rectangle, circle, ellipse, line, polyline, and polygon. Object identifier will use the tag matching technique to identify each primitive object in SVG images. Consider the image in Figure 2, there are five primitive objects identified. They are the outline of the airplane and the four windows on the airplane.

3.3 Object Aggregation

Every primitive object identified from the SVG file is distinguishable from other primitive objects. Higher level

objects can be obtained by aggregating the lower level objects according to the topological relationships between the lower level objects. The eight topological relationships used in this paper are shown in Figure 1. MBR is used to represent the two-dimensional region that an object occupies.

A primitive object p is described by MBR and shape description. This information is stored in SVG documents. Therefore, $p = (MBR, shape_description)$. If the relationship between two objects p and q are disjoint, the aggregation of p and q forms a higher level object n which can be concisely described using a new MBR derived from the MBRs of p and q [8].

$$MBR_n = [(min(x_{pb}, x_{qb}), min(y_{pb}, y_{qb})), (max(x_{pu}, x_{qu}), max(y_{pu}, y_{qu}))]$$

So object n has one attribute, MBR, i.e., $n = (MBR)$. We call this aggregation *Disjoint Aggregation*. For the case where the relation between p and q is inside/contains, covers/covered_by or equal, the higher level object n which is aggregated from p and q can be described using a new MBR and the shape description of the bigger object of p and q [8].

$$MBR_n = [(min(x_{pb}, x_{qb}), min(y_{pb}, y_{qb})), (max(x_{pu}, x_{qu}), max(y_{pu}, y_{qu}))]$$

Then we have, $n = (MBR, shape_discription)$. Since the outline of p is inside q , or the outline of q is inside p (equal, covered_by/covers are special cases of inside), this aggregation is called *Total Inside Aggregation*. In the overlap(p, q) and meet(p, q) relations, p and q are inside of each other partially (meet(p, q) is a special case of overlap(p, q)). We have an aggregation called *Partial Inside Aggregation*, which puts p and q together and forms a higher level object n . n has MBR as attribute, $n = (MBR)$ (Figure 3).

To summarize above, the aggregation function is

$$n(attribute\ list) = aggr(p, q, relation(p, q))$$

Specifically,

$$n(MBR) = disjoint_aggr(p, q)$$

where the relationship of p and q is disjoint

$$n(MBR, shape_description) = total_inside_aggr(p, q)$$

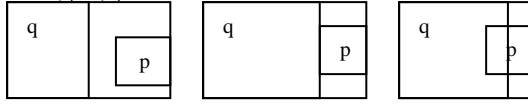
where the relationship of p and q is equal, covered_by/covers, or inside

$$n(MBR) = partial_inside_aggr(p, p)$$

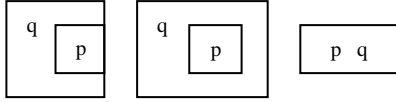
where the relationship of p and q is meet or overlaps

In all the three cases,

$$MBR_n = [(min(x_{pb}, x_{ql}), min(y_{pb}, y_{ql})), (max(x_{pw}, x_{qu}), max(y_{pw}, y_{qu}))]$$



Disjoint aggregation Partial inside aggregation



Total inside aggregation

Figure 3. Three types of aggregations. Disjoint aggregation aggregate the objects which are disjoint to each other. Partial inside aggregation aggregate objects which have meet or overlaps relation. Total inside aggregation aggregate objects which have cover/covered_by, inside/contain, or equal relation.

3.4 Aggregation-tree

Aggregation-tree can be generated based on the topological relations of the objects. Each node is a primitive object or aggregated object. The attributes of a node will be the same as generated from the aggregation function. In addition, we add an attribute *node_id* to each node in the tree to distinguish a node from others.

The construction of the aggregation rule is the following. The primitive objects involved in *disjoint aggregation* and *partial inside aggregation* will be the leaf nodes in the Aggregation-tree. Leaf node is defined as

$$leaf_node = (node_id, MBR, shape_description)$$

Intermediate nodes are those objects aggregated from lower level objects. An intermediate node can be *disjoint_aggr* node generated using disjoint aggregation function, *total_inside_aggr* node generated using total inside aggregation function, or *partial_inside_aggr* node generated using partial inside aggregation function. A *disjoint_aggr* node is defined as

$$disjoint_aggr_node = (node_id, MBR)$$

A *total_inside_aggr* node is defined as

$$total_inside_aggr_node = (node_id, MBR, shape_description)$$

A *partial_inside_aggr* node is defined as

$$partial_inside_aggr_node = (node_id, MBR)$$

3.5 Example

Let's reconsider the airplane in Figure 2. Five primitive objects are identified, one outline of the airplane and four windows. Since the four windows are disjoint to each other, the disjoint aggregation function is used to generate the higher level object, a *disjoint_aggr_node*, say windows. The four windows are leaves of the tree. The aggregated object, windows, is inside the outline of the airplane. By total inside aggregation, a new object, *total_inside_aggr_node*, the airplane, is generated. At this moment, aggregation is completed (Figure 4). Each node has a unique *node_id* and an MBR, or a unique *node_id*, an MBR, and shape description depending on the type of the node.

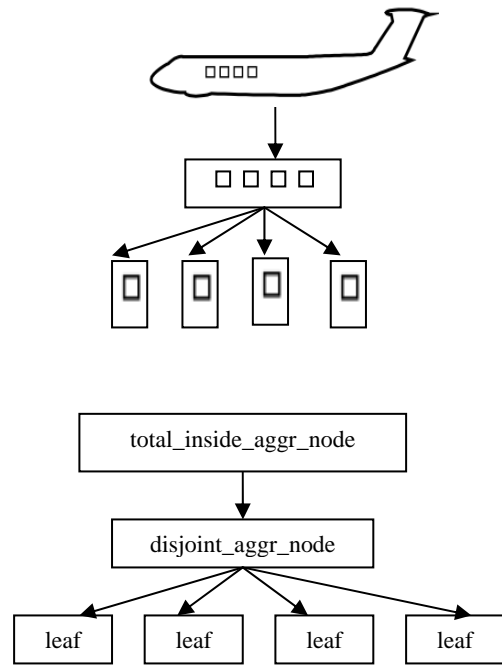


Figure 4. An example of Aggregation-tree. Upper: Aggregation-tree expressed in object images. Lower: the same Aggregation-tree showing how each node is obtained.

3.6 Advantages of the Revised Aggregation-tree

We proposed the new version of aggregation-tree in the hope of extracting metadata from SVG documents efficiently. Since only the primitive objects involved in *disjoint aggregation* and *partial inside aggregation* will be the leaves, the total number of nodes in an aggregation-tree is less than that of the previous one. Let's take the plane again as an example. Figure 5 is the

Aggregation-tree we got previously [8]. It is obvious that the new aggregation-tree in Figure 4 has fewer nodes than the previous one shown in Figure 5 without losing any information.

4. CONCLUSION

This paper has revised a previously proposed Aggregation-tree to extract metadata for SVG documents. Metadata stored in an SVG image file can be reorganized into an Aggregation-tree. The leaf nodes are the primitive objects involved in *disjoint aggregation* and *partial inside aggregation*. An intermediate node in Aggregation-tree is constructed using appropriate aggregation rule based on the topological relation between the objects. This improved Aggregation-tree has less number of nodes and models the data in SVG documents efficiently.

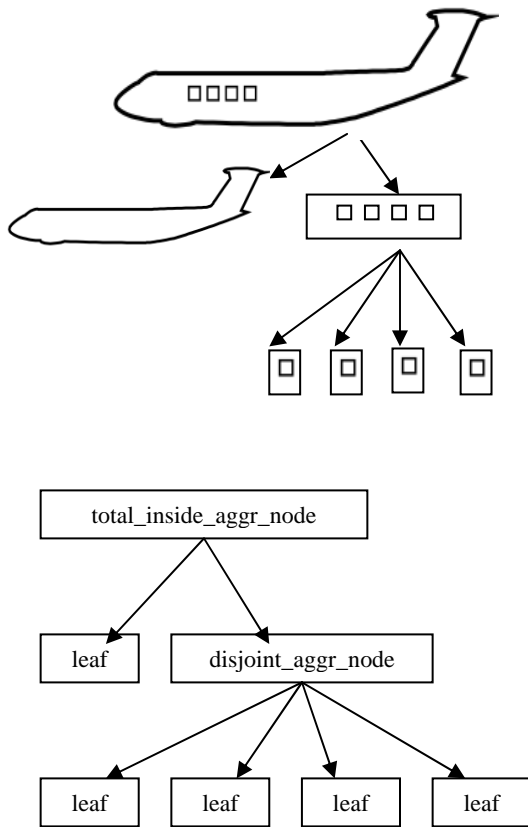


Figure 5. The previous Aggregation-tree. Upper: Aggregation-tree expressed in object images. Lower: the same Aggregation-tree showing how each node is obtained [8].

6. REFERENCES

- [1] El-Kwae, E. A., and Kabuka, M. R. A robust framework for content-based retrieval by spatial similarity in image databases. *ACM Transactions on Information Systems*, 17, 2 (April 1999), 174-198.
- [2] Gudivada, V. N., and Raghavan, V. V. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Transactions on Information Systems*, 13, 2 (April 1988), 630-638.
- [3] Guttman, A. R-trees: A dynamic index structure for spatial searching. *Proc. of ACM SIGMOD Intl. Conf. on Management of Data* (1984), 47-57.
- [4] Beckmann, N., Kriegel, H., Schneider, R., and Seeger, B. The $\$R\$$ -tree: An efficient and robust access method for points and rectangles. *Proc. ACM SIGMOD int. Conf. on Management of DATA, Atlantic City, NJ* (1990), 322-331.
- [5] Kuan, J. K. P. and Lewis, P. H. Fast k nearest neighbour search for R-tree family. *Proc. On First International Conf. on Information, Communications, and Signal Processing, Singapore*, (September 1997), 924-928.
- [6] Berg, M. de, Gudmundsson, J., Hammar, M., and Haverkort, H. J. Box-trees and R-trees with near-optimal query time. *Discrete and Computational Geometry, Vol. 28*, (2002), 291-312.
- [7] Papadias, D., and Theodoridis, Y. Topological relations in the world of minimum bounding rectangles: a study with R-trees. *ACM SIGMOD Record*, 24, 2, (May 1995), 92-103.
- [8] Bai, S., Salam, A., and Khosravi, E. Metadata Extraction from SVG Images Using A-tree. *Proceedings of the 2005 International Conference on Data Mining (DMIN'05). Las Vegas, NV*, (June 20-23), 2005, 159-165.

5. ACKNOWLEDGMENTS

This research is funded by NIMA's HBCU/MI Program.