Title:
Removing Rectangle Containment from Area NURBS Generated Shapes

Authors:
Les A. Piegl, lpiegl@gmail.com, University of South Florida, USA
Parikshit Kulkarni, Parikshit.Kulkarni@synopsys.com, Synopsys, Inc., USA
Khairan D. Rajab, khairanr@gmail.com, Najran University, Saudi Arabia

Keywords:
Rectangle containment, area NURBS, shape modeling

DOI: 10.14733/cadconfP.2014.19-21

Introduction:
Given a set of axis-aligned rectangles $R_0,\ldots,R_n$ in the plane, for each query rectangle R_q we are looking for a set of rectangles $R_k,\ldots,R_l$ so that $R_i \subset R_q, i=k,\ldots,l$. That is, we are looking for all rectangles that are *contained* inside R_q. In order to serve certain applications such as VLSI design and graphical queries, we are also looking for a set of rectangles so that $R_q \subset R_i, i=k,\ldots,l$. That is, all rectangles are sought so that they *enclose* the query rectangle R_q. The algorithm presented herein will remove both contained and enclosed rectangles simultaneously.

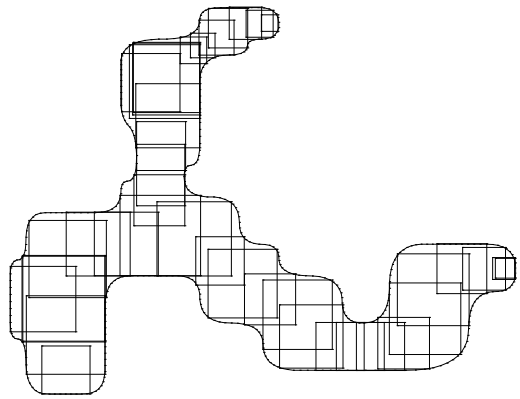

Fig. 1: Typical geometrical shape represented by a boundary and a set of rectangles.

This problem arises in applications where rectangles are used to describe shapes. Fig. 1 shows such a shape with a small set of rectangles covering a percentage of the free-form area. One way to generate such a set of rectangles is via region growing: select a guiding curve such as the medial axis, then for each discretized point on the medial axis grow a rectangle until it hits the boundary, i.e. until it cannot be grown any further without leaving the boundary or creating a skewed rectangle. Fig. 2 shows the process for one branch of the media axis. On the left a large set of 40 rectangles are produced via region growing creating lots of containments. On the right all containment and enclosures have been removed to generate 15 non-containing rectangles; more than half of the rectangles were unnecessary!

Since our main application is to generate area NURBS from these rectangles, containment is not allowed to produce meaningful shapes.

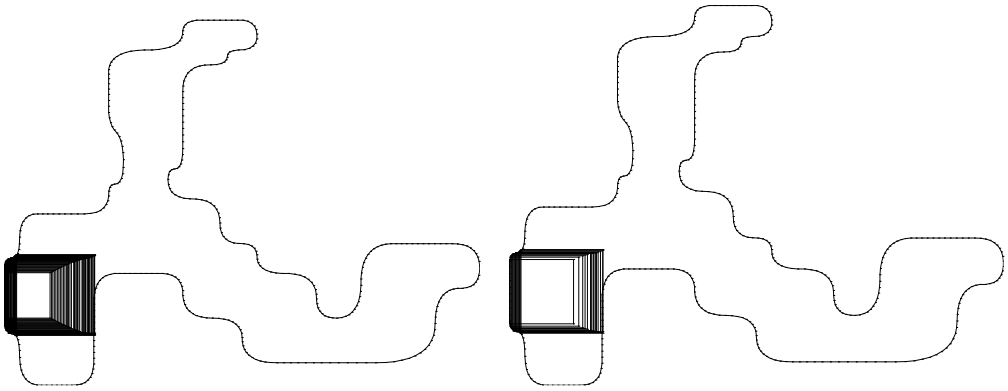

Fig. 2: Rectangles obtained via region growing: original rectangles (left), containment removed (right).

Methodology:
The algorithm presented herein has the following major components:

1. Data preparation: a grid data structure is built to assist the algorithm with fast range searching.
2. Pre-processing: each rectangle is processed into a set of cells in the grid structure that cover the given rectangle. On average about 5-15% of the total number of rectangles are associated with a cell.
3. Remove containment: for each query rectangle find all candidates that may contain this rectangle using the built grid structure. Once containment is found, update the global rectangle structure as well as the local cell indexes.
4. Get a parallel implementation on GPUs: assign a processor to a group of cells and perform the elimination in parallel for a faster performance.

Analysis:

The performance of the method depends greatly on the number of rectangles as well on the density. Nearly linear-time performance is possible on random sets as well as on sets describing shapes up to about 10,000 rectangles. Then as the density increases, so does the time complexity. The performance on GPUs is very good: using a 128-core processor, the problem is tackled in linear time up to 8,000 points within a very small fraction of a second.

Summary:
A robust algorithm for the rectangle enclosure and containment problem is presented. It is used extensively in VLSI design and proved to be stable, reliable with very little maintenance. It operates on a dynamic data structure and performs quite well even on a single processor up to some 5,000-10,000 rectangles, depending on density. The method admits easy parallel implementation and performs very well even on random data with a high level of density. Future applications will include shape similarity, image registration or layered manufacturing using rectangular shots.

References:
[1] Abel, D. J.; Smith, J. L.: A data structure and algorithm based on a linear key for a rectangle retrieval problem, Computer Vision, Graphics and Image Processing, 24, 1983, 1-13. http://dx.doi.org/10.1016/0734-189X(83)90017-8

[2] Bistiolas, V.; Sofotassios, D.; Tsakalidis, A.: Computing rectangle enclosures, Computational Geometry: Theory and Applications, 2, 1993, 303-308. http://dx.doi.org/10.1016/0925-7721(93)90012-U

[3] Bozanis, P.; Kitsios, N.; Makris, C.; Tsakalidis, A.: The space-optimal version of a known rectangle enclosure reporting algorithm, Information Processing Letters, 61, 1997, 37-41. http://dx.doi.org/10.1016/S0020-0190(96)00186-X

[4] Gupta, P.; Janardan, R.; Smid M.; Dasgupta, B.: The rectangle enclosure and point-dominance problems, International Journal of Computational Geometry and Applications, 7(5), 1997, 437-457. http://dx.doi.org/10.1142/S0218195997000260

[5] Lagogiannis, G.; Makris, C.; Tsakalidis, A.: A new algorithm for rectangle enclosure reporting, Information Processing Letters, 72, 1999, 177-182. http://dx.doi.org/10.1016/S0020-0190(99)00145-3

[6] Lee, D. T.; Preparata, F. P.: An improved algorithm for the rectangle enclosure problem, Journal of Algorithms, 3, 1982, 218-224. http://dx.doi.org/10.1016/0196-6774(82)90021-9

[7] Vaishnavi V.; Wood, D.: Data structures for the rectangle containment and enclosure problems, Computer Graphics and Image Processing, 13, 1980, 372-384. http://dx.doi.org/10.1016/0146-664X(80)90034-9