Sweep Methods for Parallel Computational Geometry

M. T. Goodrich, M. R. Ghouse, and J. Bright

Abstract. In this paper we give efficient parallel algorithms for a number of problems from computational geometry by using versions of parallel plane sweeping. We illustrate our approach with a number of applications, which include:

- General hidden-surface elimination (even if the overlap relation contains cycles).
- CSG boundary evaluation.
- Computing the contour of a collection of rectangles.
- Hidden-surface elimination for rectangles.

There are interesting subproblems that we solve as a part of each parallelization. For example, we give an optimal parallel method for building a data structure for line-stabbing queries (which, incidentally, improves the sequential complexity of this problem). Our algorithms are for the CREW PRAM, unless otherwise noted.

Key Words. Parallel algorithms, Computational geometry, Constructive solid geometry, Hidden-line elimination, Plane sweeping.

1. Introduction. There are a number of algorithms in computational geometry that rely on the “sweeping” paradigm (e.g., see [20], [34], and [42]). The generic framework in this paradigm is for one to traverse a collection of geometric objects in some uniform way while maintaining a number of data structures for the objects that belong to a “current” set. For example, the current set of objects could be defined by all those that intersect a given vertical line as it sweeps across the plane, those that intersect a line through a point p as the line rotates around p, or those that intersect a point p as it moves through the plane. The problem is solved by updating and querying the data structures at certain stopping points, which are usually called “events.” We are interested in the problem of parallelizing sweeping algorithms.

Most previous approaches to parallelizing sweeping algorithms have been to abandon the sweeping approach altogether and to solve the problem using a completely different paradigm. Examples include the line-segment intersection methods of Rüb [46] and Goodrich [23], the trapezoidal decomposition algorithm of Atallah et al. [4], the method of Aggarwal et al. [2] for constructing Voronoi diagrams, and the method of Chow [16] for computing rectangle intersections. A notable exception, which kept with the plane sweeping approach, were the methods of Atallah et al. [4] for two-set dominance counting, visibility from a point, and computing three-dimensional maxima points. In each of these algorithms, Atallah et al. adapted the cascading technique used in Cole’s merge
sort [17], to achieve a full parallelization of the sequential fractional cascading method of Chazelle and Guibas [15]. This allowed them to parallelize sweeping methods that sweep the objects with a vertical line, maintaining the set of objects cut by the line, and computing an associative function (such as "plus" or "min") on the current set of objects for each event.

In this paper we give methods for parallelizing other types of sweeping algorithms. Specifically, we address problems where the sweep can either be described as a single sequence of data operations or a related collection of operation sequences. The techniques do not depend on the sweep being defined by moving a vertical line across the plane, nor any other specific geometric object for that matter. We study cases where the sweep involves moving a point around a planar subdivision and cases where the sweep can be viewed as involving a number of coordinated line sweeps. We motivate our approach by giving efficient parallel algorithms for a number of computational geometry problems. In these cases the approach of Atallah et al. is not in itself sufficient. In particular, we derive the following results:

- **Hidden-surface elimination.** One is given a collection of opaque polygons in \( \mathbb{R}^3 \) and asked to determine the portion of each polygon that is visible from \((0, 0, +\infty)\) [22], [47], [50]. We show that this problem can be solved in \( O(\log n) \) time using \( O(n \log n + I) \) processors in the CREW PRAM model, where \( n \) is the number of edges and \( I \) is the number of edge intersections in the projections of the polygons to the \( xy \)-plane.

- **CSG evaluation.** One is given a collection of primitive objects, which are either polygons (in the two-dimensional case) or polytopes (in the three-dimensional case), and a tree \( T \) such that each leaf of \( T \) has an object associated with it and each internal node of \( T \) is labeled with a boolean operation (such as union, intersection, exclusive-union, or subtraction) [45], [52], [53]. The problem is to construct a boundary representation for the object described by the root of \( T \). We show that the two-dimensional version of this problem can be solved in \( O(\log n) \) time using \( O(n \log n + I) \) processors, and we also show how to extend this method to three-dimensional CSG evaluation.

- **Constructing rectangle contours.** One is given a collection of iso-oriented rectangles in the plane and asked to determine the edges of the contour of their union [12], [35], [57], [58]. We show that this problem can be solved in \( O(\log n) \) time using \( O(n \log n + k) \) work (which is optimal), where \( k \) is the size of the output.

- **Rectilinear hidden-surface elimination.** One is given a collection of opaque iso-oriented rectangles in \( \mathbb{R}^3 \) and asked to determine the portion of each rectangle that is visible from \((0, 0, +\infty)\) [9], [22], [25], [28], [37], [43]. We show that this problem can be solved in \( O(\log^2 n) \) time using \( O((n + k) \log n) \) work, where \( k \) is the size of the output.

One of the main ingredients in each of our solutions is the use of a parallel data structure of Atallah et al. [6] called the *array-of-trees*. We apply this data structure in a variety of ways in order to solve each of the above problems. Interestingly, for each problem, there is some additional difficulty to be overcome in order to apply our general framework, which was not an issue in the original sequential algorithm. In the case of hidden-surface elimination the difficulty is the definition of a comparison rule for polygons that is consistent even if the overlap relation contains cycles. For CSG evaluation the difficulty