PATH-PLANNING FOR A MOBILE ROBOT SWEEPER

Chandrajit L. Bajaj
Fausto Bernardini
Steve Cutchin
Kokichi Sugihara

Purdue University
Department of Computer Sciences
West Lafayette, IN 47908-1398

CSD-TR-94-060
September 1994

Path-Planning for a Mobile Robot Sweeper*

Chandrajit Bajaj Fausto Bernardini Steve Cutchin Kokichi Sugihara†

Department of Computer Science,
Purdue University,
West Lafayette, IN 47907

Telephone: 317-494-6531
FAX: 317-494-0739
Email: bajaj@cs.purdue.edu

Abstract

We present details of the design and implementation of a software simulation testbed for the evaluation of mobile robot collision-free sweeping trajectories. One is allowed to interactively create or load in a description of the workspace ("the floor") and the mobile robot ("the sweeper") as a collection of simple two dimensional (2D) primitives, including Bezier curves. The program internally computes a sliced representation of the three dimensional (3D) free configuration space using generalized 2D Voronoi diagram and offset computations. Interactive visualization and animation of different bounded velocity and acceleration trajectories can be generated in both 2D and 3D.

Category: Robotics, Collision-free Path Planning, Voronoi Diagrams, Animation

*This work was supported in part by NSF grants CCR 92-22467, AFOSR grants F49620-93-10138, ONR grant N00014-94-1-0370, CEE Project 6546 Promotion and a grant from Shinko Electric Company

†While at Purdue on his sabbatical. Permanent address: Department of Mathematical Engineering and Information Physics, University of Tokyo, Tokyo, Japan 113

1 Interface with Gati

I recompiled the FORTRAN programs written by Sugihara. These programs read a description of the workspace and the sweeper (a collection of simple 2D primitives) and produce a representation of the Voronoi diagram of points on the boundary of the obstacles and postscript files with pictures of the obstacles, the robot and the path.

The first program, `vorendo1`, reads the description of the workspace (the "floor"), approximates its boundary with a collection of points (suitably spaced), and computes the Voronoi diagram of this set of points. Its output is a file with a description of the Voronoi diagram (both its geometry and topology) and 3 postscript files with pictures of the workspace, the Voronoi diagram and the obstacle-avoiding path. The latter is obtained by simply considering the subtree of edges in the Voronoi diagram that do not cross obstacle boundaries.

The second program `vorendo2`, reads in the path computed in step 1, a description and the orientation of the sweeper, and "trims" the path so that all points in the resulting path are collision free for that particular orientation. The output of `vorendo2` is a file with a representation of the trimmed path and a postscript file with a picture of the path in the workspace.

I wrote a shell script that allows to run the programs for various orientations of the robot, given an initial angle, an increment angle and the number of steps.

I also wrote a program that reads in the path computed by `vorendo2`, computes an Euler tour on it (each undirected edge on the graph is counted twice, one per direction, so to enforce conditions of existence for such a tour), and produces a Gati script as output. A script header written by Steve Cutchin provides the correct initial position and orientation for all objects and for the view.

Examples of the output produced in the various steps are shown in Figures 1, 2, 3 and 4.

2 Computation of the free-space

I did some preliminary experiments with triangulations of free space. I wrote a program to compute the convolution of two convex polygons. I use this program to compute configuration space obstacles for a given orientation of the sweeper (this must be approximated by a convex shape, of course). Then

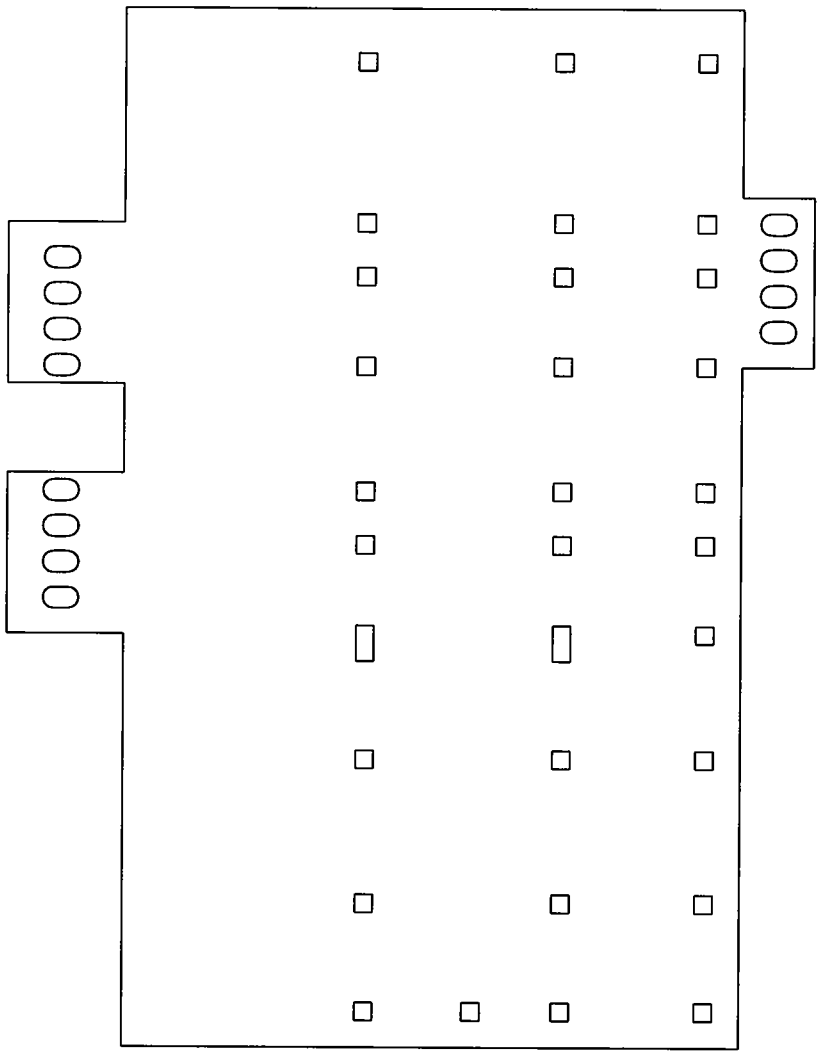

Figure 1: The workspace and the sweeper.

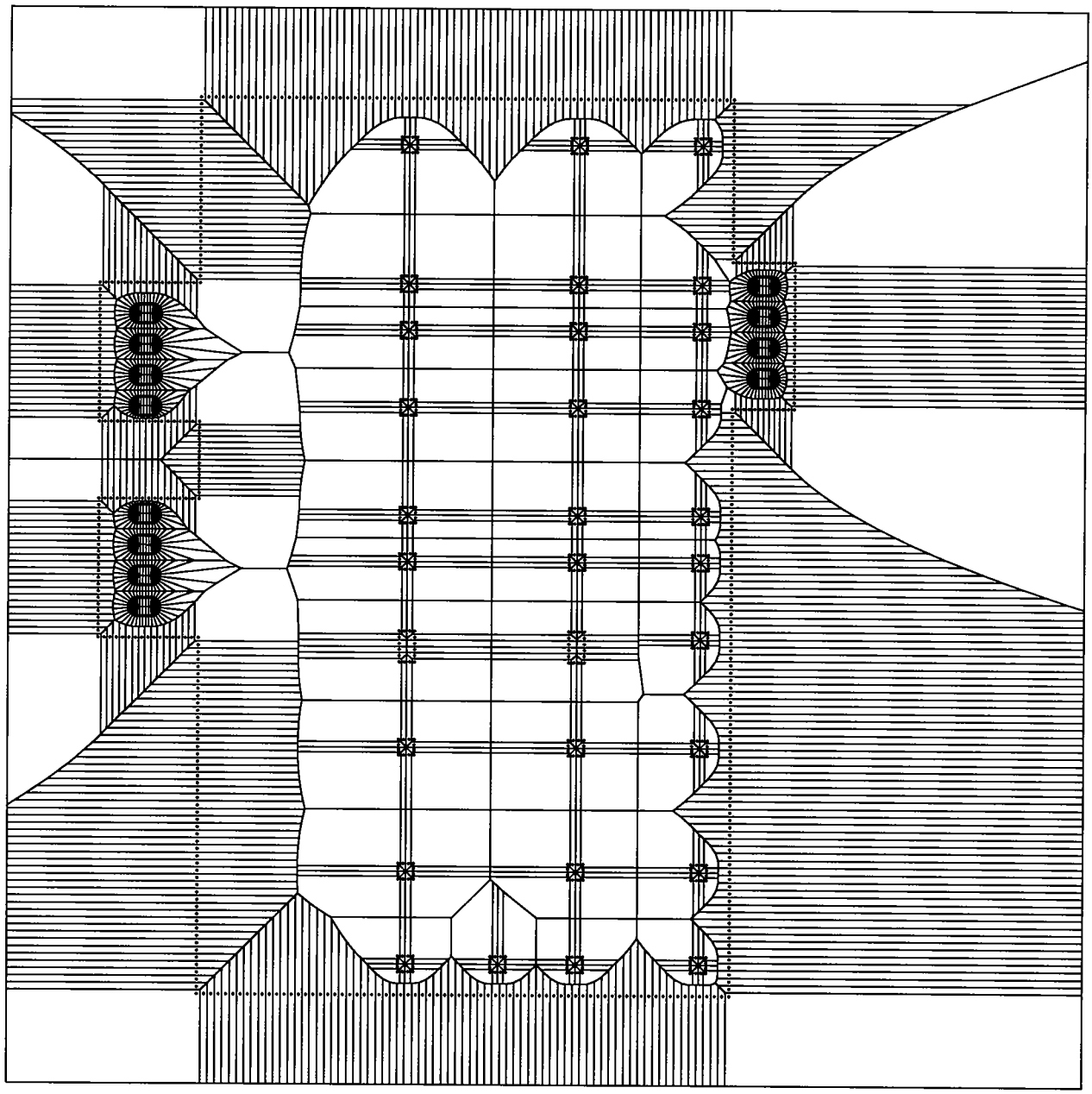

Figure 2: The Voronoi diagram computed by `vorendo1`.

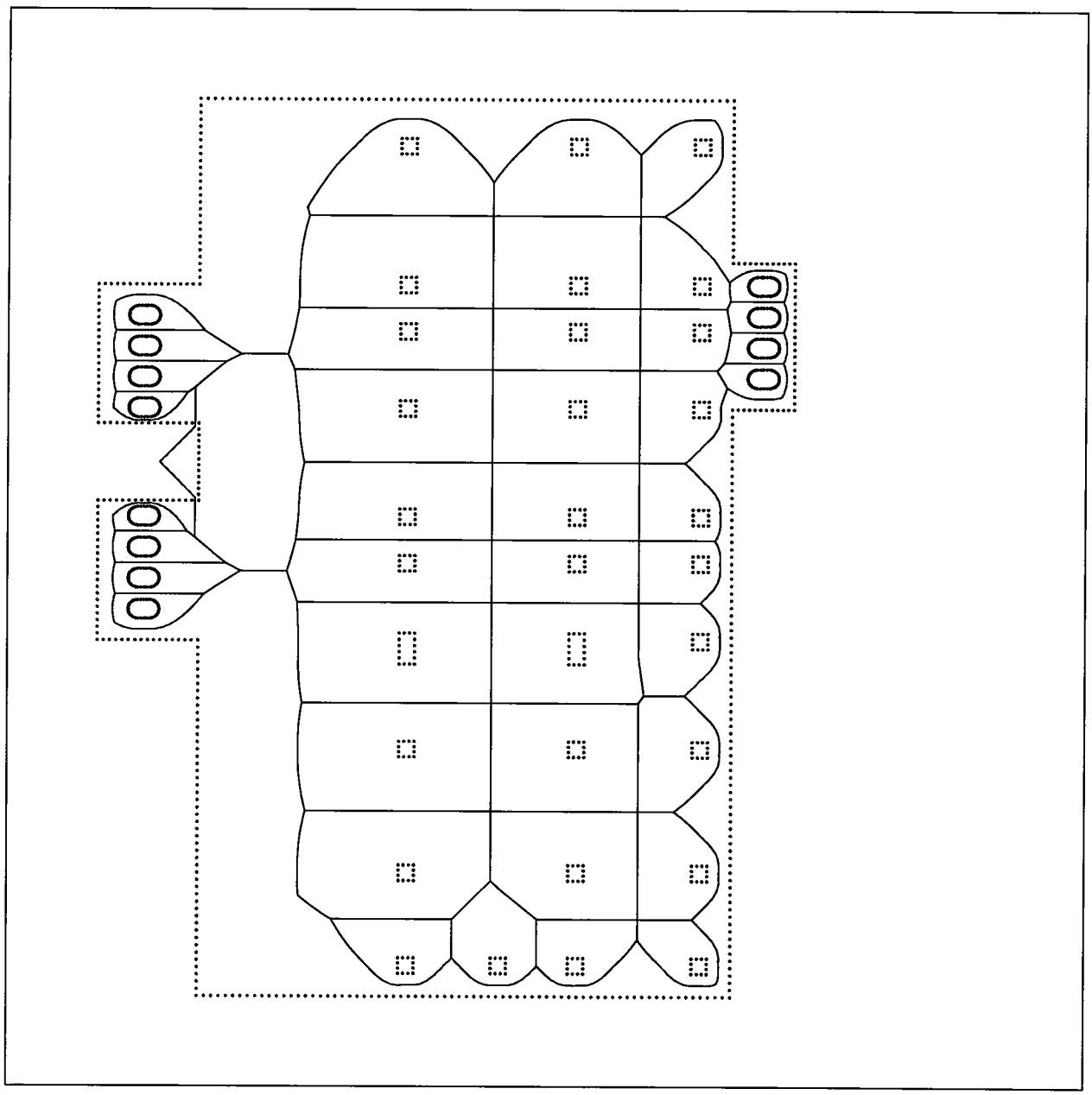

Figure 3: The path computed by `vorendo1`.

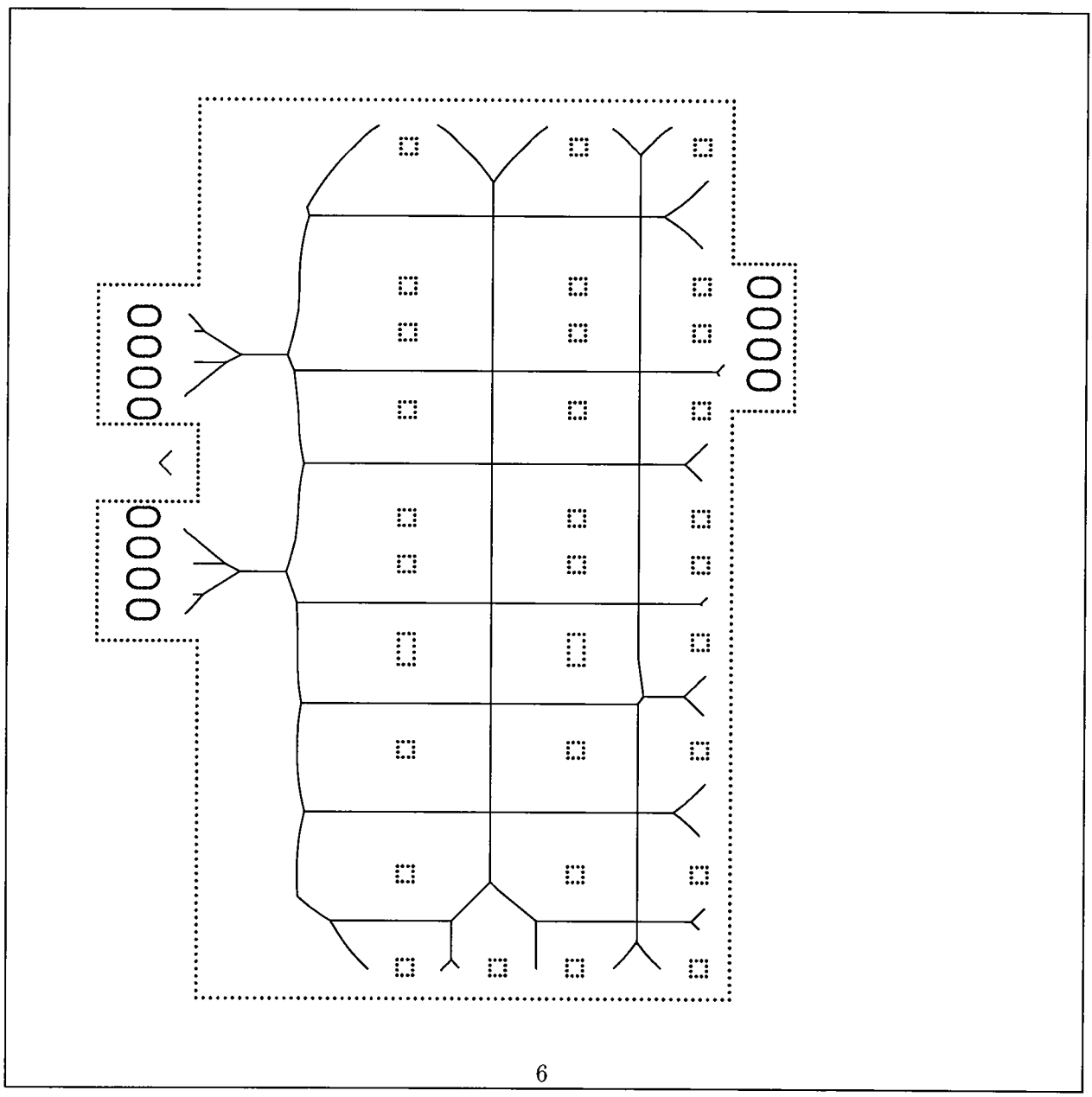

Figure 4: The trimmed path computed by `vorendo2`.

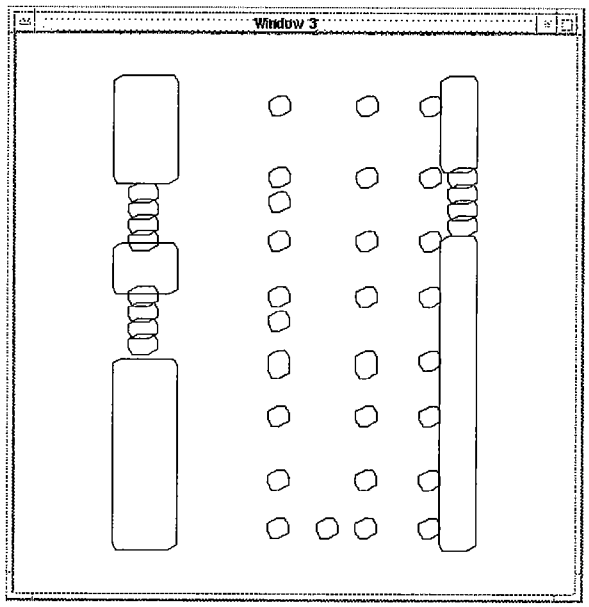

Figure 5: The configuration space obstacles, obtained as convex convolution of the obstacles and sweeper. The sweeper is rotated by 30 degrees.

I can compute the Delaunay triangulation of the vertices of these objects (I insert points to enforce all edges on the boundary of objects) and select only those triangles that are not part of some obstacle. The result is the triangulation of the free space, for that given orientation of the sweeper. I can easily compute the correspondent Voronoi diagram by duality.

Examples of the output of various steps of the algorithm are given in Figures 5, 6 and 7.

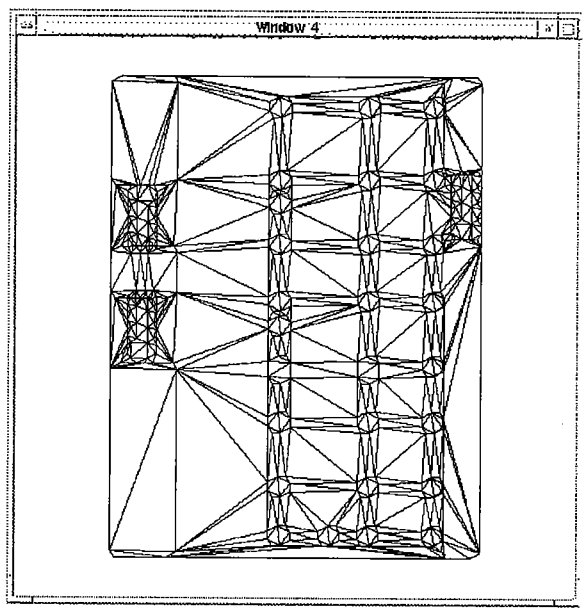

Figure 6: The constrained Delaunay triangulation of the free space.

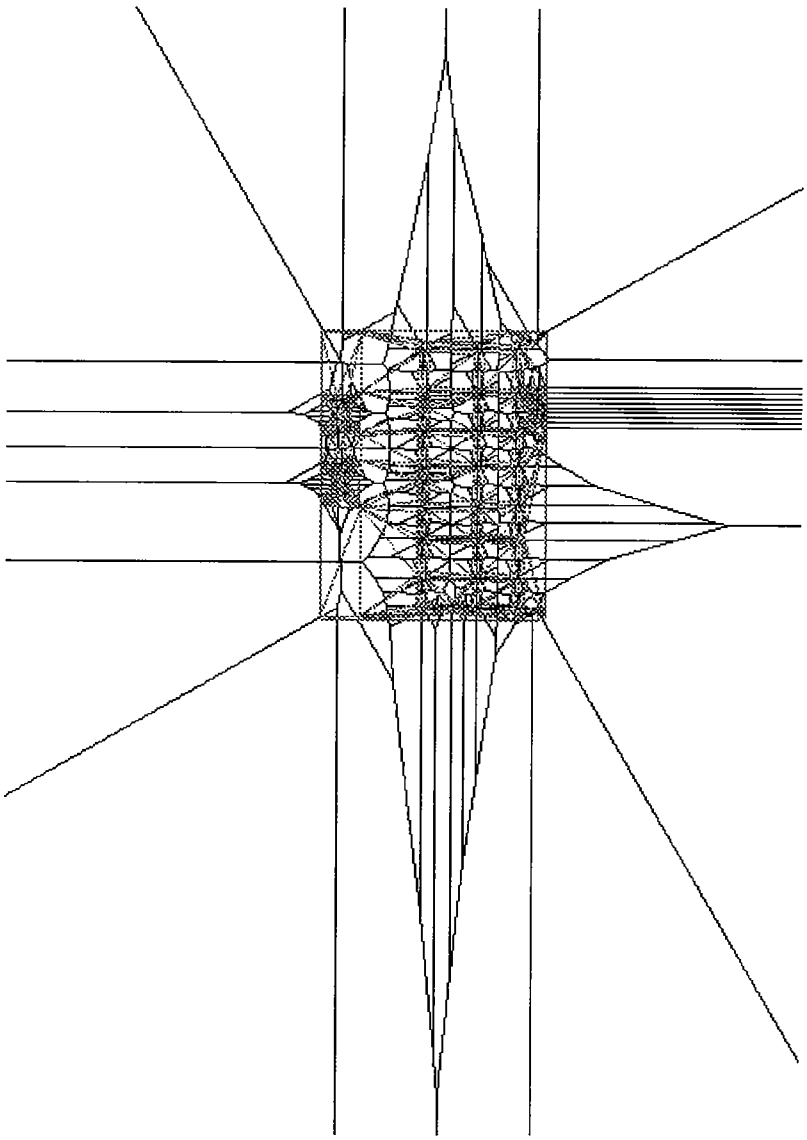

Figure 7: The Voronoi diagram of configuration space obstacles.