

Crystal Voronoi Diagram and Its Applications to Collision-Free Paths

Kei Kobayashi¹ and Kokichi Sugihara²

¹ University of Tokyo, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan,
kkoba@stat.t.u-tokyo.ac.jp

² University of Tokyo, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan,
sugihara@simplex.t.u-tokyo.ac.jp

Abstract. This paper studies the multiplicatively weighted crystal-growth Voronoi diagram, which describes the partition of the plane into crystals with different growth speeds. This type of the Voronoi diagram is defined, and its basic properties are investigated. An approximation algorithm is proposed. This algorithm is based on a finite difference method, called a fast marching method, for solving a special type of a partial differential equation. The proposed algorithm is applied to the planning of a collision-free path for a robot avoiding enemy attacks.

1 Introduction

Suppose that various types of crystals grow from different start points in the plane with different speeds. When two crystal regions meet, they stop growing in that direction. Then, the plane is partitioned into individual crystal regions; this partition is called the *multiplicatively weighted crystal-growth Voronoi diagram*, which is the topic of this paper.

A number of types of generalized Voronoi diagrams have been proposed on the basis of different types of weighted distances, including the additively weighted Voronoi diagrams, the multiplicatively weighted Voronoi diagrams, and the compoundly weighted Voronoi diagrams [2,3]. However, the multiplicatively weighted crystal-growth Voronoi diagram is quite different from the others, because a crystal cannot enter into the area which is already occupied by another crystal. A crystal with a high speed should grow around avoiding slowly growing crystals. Hence, the “distance” between two points at a given time should be measured by the length of the shortest path that avoids crystal regions generated by that time. In this sense, the computation of this Voronoi diagram is very hard.

The concept of the multiplicatively weighted crystal-growth Voronoi diagram was first proposed by Schaudt and Drysdale [1]. They presented an $O(n^3)$ approximation algorithm for n crystals.

This paper studies this Voronoi diagram from various points of view. First, we present a new approximation algorithm for constructing this Voronoi diagram. Secondly, we apply this Voronoi diagram to the search of the shortest path for a robot that moves among enemy robots.

The structure of the paper is the following. In Section 2, we review definitions and fundamental properties of Voronoi diagrams. In Section 3, we construct a new algorithm for approximately computing the multiplicatively weighted crystal-growth Voronoi diagram, and in Section 4, it is applied to the collision-free path planning for robots. In Section 5, we give the conclusion.

2 Multiplicatively Weighted Crystal-Growth Voronoi Diagram

2.1 Ordinary Voronoi Diagram

Let $S = \{P_1, P_2, \dots, P_n\}$ be a set of n points in the plane. For each P_i , let $R(S; P_i)$ be the set of points that are nearer to P_i than to other P_j 's ($j \neq i$), that is,

$$R(S; P_i) = \{P \mid \|P - P_i\| < \|P - P_j\|, j \neq i\}, \tag{1}$$

where $\|P - Q\|$ denotes the Euclidean distance between the two points P and Q . The plane is partitioned into $R(S; P_1), R(S; P_2), \dots, R(S; P_n)$ and their boundaries. This partition is called the *Voronoi diagram* for S , and the elements of S are called the *generators* of the Voronoi diagram. The region $R(S; P_i)$ is called the *Voronoi region* of P_i , and the boundary lines of the Voronoi diagram are called *Voronoi edges*.

In the following subsections we generalize the concept of the Voronoi diagram. In order to avoid confusion, the above-defined Voronoi diagram is sometimes called the *ordinary* Voronoi diagram.

2.2 Multiplicatively Weighted Voronoi Diagram

Let $S = \{P_1, P_2, \dots, P_n\}$ be the set of points in the plane, and v_i be a positive real assigned to P_i for $i = 1, 2, \dots, n$. For any point P , we call $\|P - P_i\|/v_i$ the *multiplicatively weighted distance*, and call v_i the *weight* assigned to P_i . We define region $R_m(S; P_i)$ by

$$R_m(S; P_i) = \{P \mid \|P - P_i\|/v_i < \|P - P_j\|/v_j, j \neq i\}, \tag{2}$$

that is, $R_m(S; P_i)$ denotes the set of points that is closer to P_i than to any other P_j in terms of the multiplicatively weighted distance. The plane is partitioned into $R_m(S; P_1), R_m(S; P_2), \dots, R_m(S; P_n)$. This partition is called the *multiplicatively weighted Voronoi diagram* [2,5].

A boundary of two Voronoi regions is a part of a circle, which is known as the Apollonius circle [6].

Fig. 1 shows an example of a multiplicatively weighted Voronoi diagram; the numbers in the parentheses represent the weights of the generators.

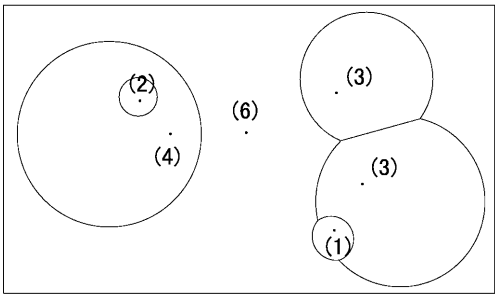


Fig. 1. Multiplicatively weighted Voronoi diagram

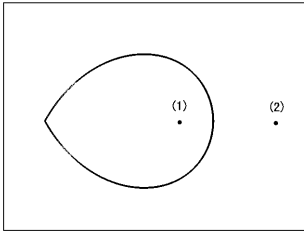


Fig. 2. Multiplicatively weighted crystal Voronoi diagram

2.3 Multiplicatively Weighted Crystal-Growth Voronoi Diagram

As in previous subsections, let $S = \{P_1, P_2, \dots, P_n\}$ be the set of generators in the plane and v_i be the weight assigned to P_i . Suppose that for each i , the i -th crystal grows from P_i by its own speed v_i . The crystals can grow only in empty areas; they cannot intrude into those areas that are already occupied by other crystals. Hence, a faster crystal must go around slower crystals. Thus, unlike the multiplicatively weighted distance, the time required for the i -th crystal to reach P is not determined by P and P_i only; it depends also on the locations and speeds of other crystals.

In this sense, the resulting crystal pattern is different from the multiplicatively weighted Voronoi diagram. This crystal pattern is called the *multiplicatively weighted crystal-growth Voronoi diagram*, or the *crystal Voronoi diagram* for short.

In the crystal Voronoi diagram, each crystal behaves as an obstacle against other crystals. Hence, for a point P in the i -th crystal region the distance from P_i to P should be measured along the shortest path completely included in the crystal.

Fig. 2 shows the crystal Voronoi diagram for two generators, P_1, P_2 , with weights 1 and 2.

If all the growth speed v_i are the same, the crystal Voronoi diagram coincides with the ordinary Voronoi diagram.

Note that, unlike the multiplicatively weighted Voronoi diagram, the Voronoi region of a crystal Voronoi diagram is always connected. This is because a crystal cannot go through other crystals in the process of growing.

3 Simulation of the Crystal Growth

We can obtain the boundary for two crystals in the analytic form. But for three or more crystals, the calculation becomes difficult and complicated. In this section

we consider a method for computing the boundary curves approximately. For this purpose we employ the fast marching method for solving a certain type of a partial differential equation.

3.1 Fast Marching Method

Eikonal Equation. Let $\Omega \subset \mathbf{R}^2$ be a bounded region in the plane, and Γ be its boundary. Let $F(\mathbf{x})$ be a real-valued functions satisfying $F(\mathbf{x}) > 0$ for any $\mathbf{x} \in \Omega$. Furthermore, let $g(\mathbf{x})$ be a function on Γ . We consider a nonlinear partial differential equation

$$|\nabla u(\mathbf{x})| = F(\mathbf{x}) \quad \text{in } \Omega \tag{3}$$

with a boundary condition

$$u(\mathbf{x}) = g(\mathbf{x}) \quad \text{on } \Gamma, \tag{4}$$

where $F(\mathbf{x})$ and $g(\mathbf{x})$ are known and $u(\mathbf{x})$ is unknown. The equations (3) is called the *Eikonal equation*.

Assume that $1/F(\mathbf{x})$ represents the speed of a moving object at point \mathbf{x} in Ω , and that $g(\mathbf{x}) = 0$ on Γ . Then, the solution $u(\mathbf{x})$ of the above Eikonal equation can be interpreted as the shortest time required for the object initially on the boundary Γ to reach the point \mathbf{x} . Therefore, we can use this equation to represent the behavior of the growth of a crystal. In particular, if $F(\mathbf{x}) = \infty$ in some area, this area behaves as an obstacle because the speed (i.e., $1/F(\mathbf{x})$) in this area is considered 0. This property is suitable to our purpose, because the areas occupied by crystals behave as obstacles to other crystals. In what follows, we assume that $g(\mathbf{x}) = 0$ on Γ .

To solve the equation (3) together with the boundary condition (4), Sethian [4] proposed a finite-difference method, called the *fast marching method*. In the finite-difference method, the unknown continuous function $u(\mathbf{x}) = u(x, y)$ is replaced by a finite set of values at discretized points

$$u_{i,j} = u(i\Delta x, j\Delta y), \tag{5}$$

where Δx and Δy are small values representing the interval for discretization in the x and y directions. We set the values of $u_{i,j}$'s on Γ being 0, and starting with these boundary points, we compute the values of the other $u_{i,j}$'s in the increasing order.

Apparently similar techniques have already been used in digital picture processing; they are called *distance-transformation* methods [7]. But usually the obtained distance is either L_1 -distance or L_∞ -distance, which is different from what we want to obtain, i.e., the Euclidean distance. Algorithms for obtaining the Euclidean distance are also proposed in digital image processing [8,9], but they cannot treat the obstacles, and hence cannot be applied to our purpose.

Finite-Difference Equation in the First Marching Method. Using the discretized value $u_{i,j}$, Sethian proposed finite-difference approximations of the equation (3). The most basic approximation is the first-order finite-difference equation defined by

$$[\max(D_{i,j}^{-x}u, -D_{i,j}^{+x}u, 0)^2 + \max(D_{i,j}^{-y}u, -D_{i,j}^{+y}u, 0)^2]^{1/2} = F_{i,j}, \quad (6)$$

where

$$\begin{aligned} D_{i,j}^{-x}u &= \frac{u_{i,j} - u_{i-1,j}}{\Delta x}, \quad D_{i,j}^{+x}u = \frac{u_{i+1,j} - u_{i,j}}{\Delta x}, \\ D_{i,j}^{-y}u &= \frac{u_{i,j} - u_{i,j-1}}{\Delta y}, \quad D_{i,j}^{+y}u = \frac{u_{i,j+1} - u_{i,j}}{\Delta y}, \\ F_{i,j} &= F(i\Delta x, j\Delta y). \end{aligned} \quad (7)$$

Eq. (6) is used to compute the unknown value $u_{i,j}$ from given u values at the upwind neighbor points and given $F_{i,j}$ [4].

Sethian also proposed the second-order approximation of eq. (3) by

$$\left(\begin{aligned} &\max\left[\left[D_{i,j}^{-x}u + \text{switch}_{i,j}^{-x} \frac{\Delta x}{2} (D_{i,j}^{-x})^2 u\right], \right. \\ &\quad \left. -\left[D_{i,j}^{+x}u + \text{switch}_{i,j}^{+x} \frac{\Delta x}{2} (D_{i,j}^{+x})^2 u\right], 0\right]^2 \\ &\quad + \\ &\max\left[\left[D_{i,j}^{-y}u + \text{switch}_{i,j}^{-y} \frac{\Delta y}{2} (D_{i,j}^{-y})^2 u\right], \right. \\ &\quad \left. -\left[D_{i,j}^{+y}u + \text{switch}_{i,j}^{+y} \frac{\Delta y}{2} (D_{i,j}^{+y})^2 u\right], 0\right]^2 \end{aligned} \right)^{\frac{1}{2}} = F_{i,j}, \quad (8)$$

where

$$\text{switch}_{i,j}^{\pm x} = \begin{cases} 1, & \text{if } u_{i\pm 2,j} \text{ and } u_{i\pm 1,j} \text{ are known and } u_{i\pm 2,j} \leq u_{i\pm 1,j}, \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

and $\text{switch}_{i,j}^{\pm y}$ is defined similarly.

The coefficient switch in eq. (8) is necessary, because $F(\mathbf{x})$ depends on \mathbf{x} so that the shortest path might be curved, and consequently $u_{i-2,j}$, for example, might not be known even if the upwind-neighbor value $u_{i-1,j}$ is known.

For our purpose of computing the crystal Voronoi diagram, we use the first-order approximations to choose the upwind neighbors, and use the second-order approximation to compute the value of $u_{i,j}$.

Original Fast Marching Algorithm. The original fast marching algorithm proposed by Sethian is as follows.

Algorithm 1 (Fast marching method)

Step 1 (Initialization). Cover the region Ω with grid points $(i\Delta x, j\Delta y)$. Initialize KNOWN to be the set of all grid points on the boundary Γ , and TRIAL to be the set of all points that are one-grid far from KNOWN, and FAR to be the set of all the other points. Initialize the value $u_{i,j}$ as $u_{i,j} = 0$ for points in KNOWN, $u_{i,j} = \inf$ for points in FAR, and determine the value of $u_{i,j}$ according to eq. (8) for points in TRIAL.

Step 2 (Main loop). Repeat Steps 2.1 to 2.5.

2.1. From TRIAL choose and delete the point, say Q , with the smallest u value, and add it to KNOWN.

2.2. For each of the four neighbors of Q that is in FAR, move it from FAR to TRIAL.

2.3. For each of the four neighbors of Q that are in TRIAL, compute the u value using eq. (8). (If the point already has the u value, update it only if the new u value is smaller than the old one.)

2.4. If TRIAL is empty, stop. Otherwise go to 2.1. ■

If we use a heap for representing and manipulating the set TRIAL, this algorithm runs in $O(N \log N)$ time for N grid points. Refer to [4] for the details of this algorithm.

3.2 Computation of the Crystal Voronoi Diagram

We apply the fast marching method to the simulation of the growth of crystals. We discretize the region in which we want to compute the crystal structure into grid points, and assign the generators to the nearest grid points, say P_1, P_2, \dots, P_n . Let N be the total number of the grid points. We assign sequential numbers to all the grid points, and name them as Q_1, Q_2, \dots, Q_N . Basically we follow Algorithm 1, but in several points we change it in the following way.

First, for each grid point Q_j , we assign the “crystal name” $CNAME[Q_j]$, which represents the ordinal number of the crystal to which Q_j belongs. The value of $CNAME[Q_j]$ is either an integer from 1 to n or “NONE”. At the initial stage, we set $CNAME[P_k] = k$ for all the generators P_k , $k = 1, 2, \dots, n$, set $CNAME[Q_j] = k$ for grid point Q_j that is one-grid far from P_k , and set $CNAME[Q_j] = \text{NONE}$ for the other grid points. Whenever the k -th crystal reaches Q_j , $CNAME[Q_j]$ is changed to k .

Secondly, at the initial stage, we set KNOWN to be the set $\{P_1, P_2, \dots, P_n\}$ of the generators.

Thirdly, for the computation of the u value of a four-neighbor point, say Q_j , in TRIAL of the point Q in Step 1 or in Step 2.3 in Algorithm 1, we slightly modify the procedure in the following way.

(i) We read the crystal name $k = CNAME[Q]$, and use the growth speed of the k -th crystal, that is, we substitute $F_{i,j} = 1/v_k$ to eq. (8).

(ii) We use the u values of only those points Q_l that are included in the k -th crystal, i.e., $CNAME[Q_l] = k$, in solving eq. (8).

(iii) Because of the above modifications (i) and (ii), the resulting u value is not necessary smaller than the previous value. Hence, only when the recomputed u value is smaller than the present value, we update the u value, and change $CNAME[Q_j]$ to k .

The output of the fast marching method modified as described above can be interpreted as the crystal Voronoi diagram in the sense that each grid point Q_j belongs to the crystal $CNAME[Q_j]$.

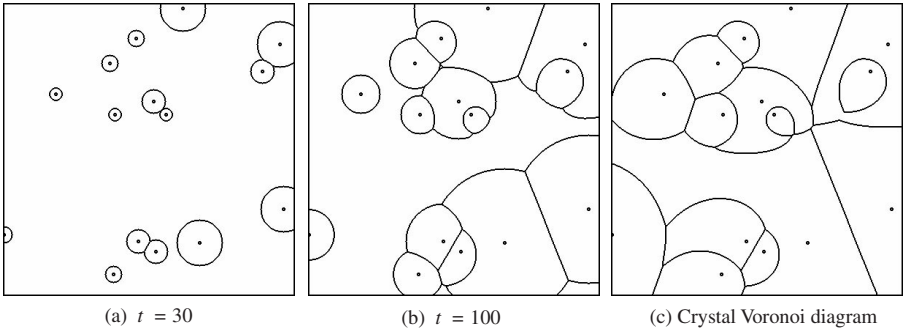


Fig. 3. Simulation of crystal Voronoi diagram by the fast marching method (t means the radius of the fastest growing crystal when the width between grids is one)

Fig. 3 shows the behavior of the algorithm. Here, the square region was replaced by 400×400 grid points and 15 generators were placed. Fig. 3 (a) and (b) show the frontiers of the crystals at the stage where the fastest crystal grows 30 times the grid distance and 100 times the grid distance, respectively. Fig. 3 (c) shows the final result.

4 Application to Path Planning

4.1 Fast Marching Method for Collision-Free Path

Sethian applied the fast marching method to the collision-free path among static obstacles [4]. Here, we extend his idea, and propose a method for finding a collision-free path among moving competitive robots. First, let us review Sethian’s idea [4].

The Eikonal equation (3) can be written in the integral form as

$$u(\mathbf{x}) = \min_{\gamma} \int_A^{\mathbf{x}} F(\gamma(\tau)) d\tau, \tag{10}$$

where A is a start point, γ is a path from A to \mathbf{x} in Ω . Thus, $u(\mathbf{x})$ represents the shortest time in which a robot can move from A to \mathbf{x} . Suppose that we get $u(\mathbf{x})$ for every point \mathbf{x} in Ω using the fast marching method. Next, for any point B in Ω , the solution $X(t)$ of equation

$$X(t) = -\nabla u, \quad X(0) = B \tag{11}$$

gives the shortest path from A to B .

This idea can be extended to the case where the robot has its own shape instead of just a point. Suppose, for example, that a moving robot is a rectangle. Let (x, y) be the location of the center of the robot and θ be the angle of the longer edge of the rectangle with respect to the positive x direction; we measure

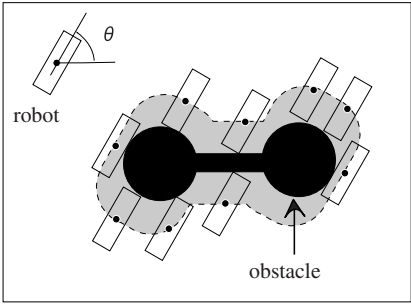


Fig. 4. The area where the robot’s center cannot enter when it rotates at an angle of θ

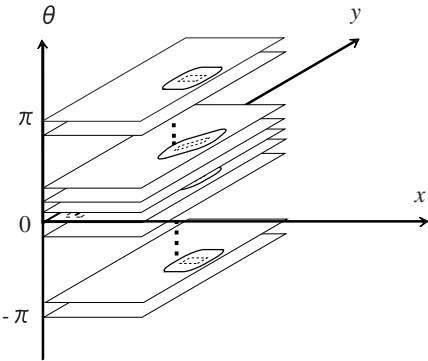


Fig. 5. 3-dimensional space of fast marching method for robot navigation

the angle counterclockwise. Thus the position and the posture of the robot can be represented by a point (x, y, θ) in a three-dimensional parameter space.

Next for each θ , we find the region in which the robot cannot enter without colliding the obstacle, as shown by the shaded area in Fig. 4. The boundary of this region can be obtained as the trajectory of the center of the robot that moves around keeping in contact with the obstacle. For this fixed θ , to consider the rectangular robot moving around the original obstacle is equivalent to consider a point robot moving around the extended region. Thus, we can reduce the problem of the moving robot among the obstacles to the problem of a moving point among the enlarged obstacles.

However, this reduction should be done for each value of θ . Hence, we discretize θ as well as x and y , and construct the three-dimensional grid structure as shown in Fig. 5. A fixed value of θ corresponds to a horizontal plane, in which we extend the obstacles.

Sethian used the fast marching method to solve the Eikonal equation

$$\left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \alpha \left(\frac{\partial u}{\partial \theta} \right)^2 \right]^{1/2} = 1 \tag{12}$$

in the three-dimensional (x, y, θ) space. The partial derivatives $\partial u / \partial x$ and $\partial u / \partial y$ represent the inverses of x and y components of the velocity while $\partial u / \partial \theta$ represents the inverse of the angular velocity. The coefficient α represents the ratio of the time to translate the robot by unit length over the time to rotate the robot by unit angle.

4.2 Extension to Competitive Robots

Here we consider the situation where our robot moves among enemy robots. Suppose that our robot has an arbitrary shape while the enemy robots are circles,

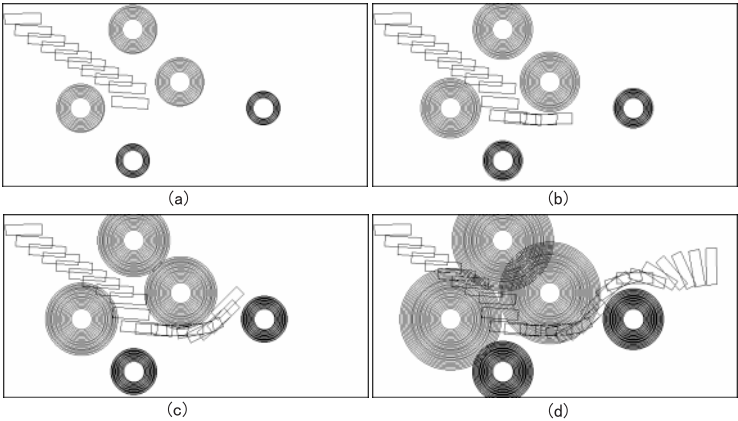


Fig. 6. Optimal answers of the robot navigation problems.

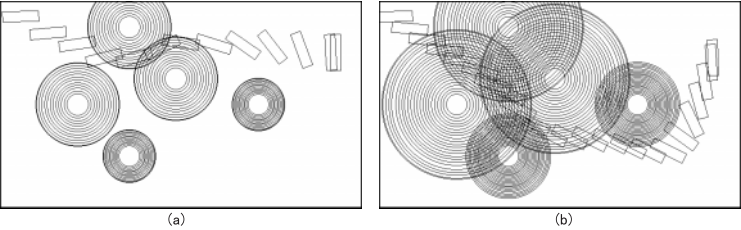


Fig. 7. Optimal answers of the robot navigation problems for other robot velocities

and each robot has its own velocity. Our robot wants to move avoiding enemies from the start point to the goal as fast as possible, while the enemy robots try to attack it. In this situation we want to find the worst-case optimal path from the start point to the goal.

For this purpose, we can apply the first marching method. The only difference from Sethian’s path planning is that the obstacles are not static; they move with the intention to attack our robot. Hence, as we extended Sethian’s fast marching method to the crystals, we treat the enemy robots as if they are crystals growing isotropically in every direction; these crystal regions represent the maximum area that the enemy robot can reach.

Fig. 6 shows an example of the collision-free path found by our method. The five enemy robots, starting with the initial circles representing the sizes of the robots, grow their regions by their own speed. Our robot, on the other hand, is a rectangle that can translate and rotate. In Fig. 6, (a), (b) and (c) show the status at some instants, while (d) shows the whole path of the robot to reach the goal. Fig. 7 (a) shows the generated path for the case that our robot can move faster than in Fig. 6, while Fig. 7 (b) shows the case that our robot moves more slowly than in Fig. 6.

5 Concluding Remarks

This paper studied the crystal Voronoi diagram from the computational point of view. First, we presented a method for computing the approximated diagram, where we modified the fast marching method to solve the Eikonal equation.

The approximation method proposed by Schaudt and Drysdale [1] requires $O(n^3)$ time for n crystals, whereas our new method runs in $O(N \log N)$ time for N grid points. This time complexity does not depend on the number of crystals.

Furthermore, we applied the crystal Voronoi diagram to the collision-free path planning among enemy robots, and evaluated our method by computational experiments.

One of the main problems for future is to raise the efficiency of the method. We might decrease the computational cost by using a coarse grid together with interpolation techniques. We might also decrease the memory cost by discarding the u values except around the frontiers of the crystals.

In our application to the path planning among competitive robots, we assumed that the enemy robots are circles. To generalize our method for arbitrary enemy shapes is another important problem for future.

Acknowledgements. The authors express their thanks to Prof. K. Hayami, Mr. T. Nishida and Mr. S. Horiuchi of the University of Tokyo for valuable comments. This work is supported by Toray Science Foundation, and the Grant-in-Aid for Scientific Research of the Japanese Ministry of Education, Science, Sports and Culture.

References

1. B.F. Schaudt and R.L. Drysdale: Multiplicatively weighted crystal growth Voronoi diagram. *Proceedings of the Seventh Annual Symposium on Computational Geometry* (North Conway, June 1991), pp. 214–223.
2. F. Aurenhammer: Voronoi diagrams—A survey of a fundamental Geometric data structure. *ACM Computing Surveys*, vol. 23, no. 3 (1991), pp. 345–405.
3. A. Okabe, B. Boots, and K. Sugihara: *Spatial Tessellations—Concepts and Applications of Voronoi Diagrams*. John Wiley, Chichester, 1992.
4. J.A. Sethian: Fast marching methods. *SIAM Review*, vol. 41, no. 2 (1999), pp. 199–235.
5. C.A. Wang and P.Y. Tsin: Finding constrained and weighted Voronoi diagrams in the plane. *Proceedings of the Second Canadian Conference in Computational Geometry* (Ottawa, August 1990), pp.200–203.
6. D. Pedoe: *Geometry—A Comprehensive Course*. Cambridge University Press, London, 1970.
7. A. Rosenfeld and J. Pfalts: Sequential operations in digital picture processing. *Journal of ACM*, vol. 13 (1966), pp. 471–494.
8. L. Chen and H.Y.H. Chuang. A fast algorithm for Euclidean distance maps of a 2-d binary image. *Infor. Process. Lett.* vol. 51 (1994), pp. 25–29.
9. T. Hirata: A unified linear-time algorithm for computing distance maps. *Infor. Process. Lett.*, vol. 58 (1996), pp. 129–133.