数理解析研究所講究録
1185巻 2001年 109-119

Crystal Voronoi Diagram and Its Applications

Kei KOBAYASHI and Kokichi SUGIHARA

Graduate School of Engineering, University of Tokyo
Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
{keikoba,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. The analytic equation describing the boundary curve is given for a simple case. For the general case, 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.

Keywords: crystal Voronoi diagram, fast marching method, collision-free path

1 Introduction

Suppose that various types of crystals grow from different start points in the plane with different speeds. 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 aditively 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. Indeed, it is difficult to compute this diagram strictly, and their approximation algorithm is the only algorithm proposed so far.

This paper studies this Voronoi diagram from various points of view. First, for the case of only two crystals, we derive an analytic expression of the boundary curve of the diagram. Secondly, we present a new approximation algorithm for constructing the Voronoi diagram for a general case. This algorithm is based on a finite difference method for solving a partial differential equation [4], and generates an approximation of the Voronoi diagram in the form of a digital picture. The time complexity does not depend on the number of crystals; it depends on the size of the pixels in the digital picture. Thirdly, 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 the definitions and the fundamental properties of the ordinary Voronoi diagram, the multiplicatively weighted Voronoi diagram, and the multiplicatively weighted crystal-growth Voronoi diagram. In Section 3, we present an analytic expression of the boundary of the Voronoi diagram with two crystals. In Section 4, we construct a new algorithm for approximately computing the multiplicatively weighted crystal-growth Voronoi diagram. In Section 5, our algorithm is applied to another variant of the generalized Voronoi diagram, and in Section 6, it

is applied to the collision-free path planning for robots. In Section 7, we give the conclusion.

2 Multiplicatively Weighted Crystal-Growth Voronoi Diagram

2.1 Ordinary Voronoi Diagram

Let $S = \{P_1, P_2, \cdots, P_n\}$ 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 $R(S; P_1), R(S; P_2), \cdots, R(S; P_n)$ 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*. Fig. 1 shows an example of the Voronoi diagram.

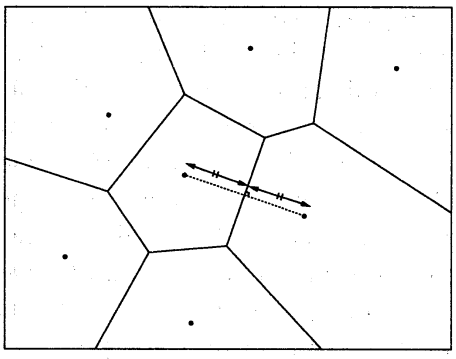


Figure 1: Voronoi diagram.

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 $S = \{P_1, P_2, \cdots, P_n\}$ be the set of points in the plane, and v_i be a positive real assigned to P_i for $i = 1, 2, \cdots, n$. 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 $R_m(S; P_1), R_m(S; P_2), \cdots, R_m(S; P_n)$. This partition is called the *multiplicatively weighted Voronoi diagram* [2, 6].

A boundary of two Voronoi regions is a part of a circle, which is known as the Apolonius circle [7].

Fig. 2 shows an example of a multiplicatively weighted Voronoi diagram; the numbers in the parentheses represent the weights of the generators.

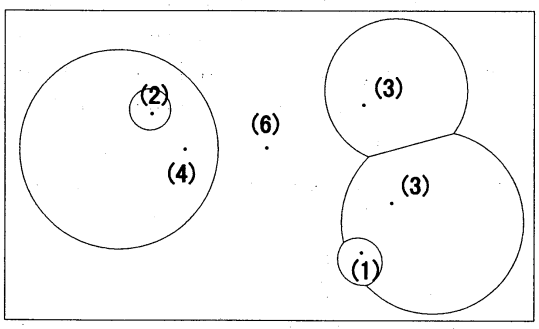


Figure 2: Multiplicatively weighted Voronoi diagram.

Note that the multiplicatively weighted distance is defined as the Euclidean distance multiplied by $1/v_i$, but not by v_i. This definition is intuitively natural because a larger weight implies a larger Voronoi region $R_m(S; P_i)$. In other words, we can interpret v_i as the velocity of a vehicle assigned to P_i, and the weighted distance $\|P - P_i\|/v_i$ as the time required by the vehicle to travel from P_i to P. Hence the multiplicatively weighted Voronoi diagram can be understood as the partition of the plane according to which vehicle can reach in the shortest time.

In the multiplicatively weighted Voronoi diagram, a region $R_{\mathrm{m}}(S;\mathrm{P}_i)$ may be disconnected. Indeed, a generator with a large weight gets an area that is beyond the regions of generators with smaller weights.

2.3 Multiplicatively Weighted Crystal-Growth Voronoi Diagram

As in previous subsections, let $S = \{\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_n\}$ 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 area; 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. 3 shows the crystal Voronoi diagram for two generators 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 jump in the process of growing.

3 Analytic Solution for the Simplest Case

In this section we consider the simplest case, that is, the crystal Voronoi diagram for two generators. This case was already studied by Schaudt and Drysdale [1]; they stated that the boundary curve is a logarithmic spiral, but they did not give an explicit expression of this curve. Here, we derive the analytic expression of this curve explicitly.

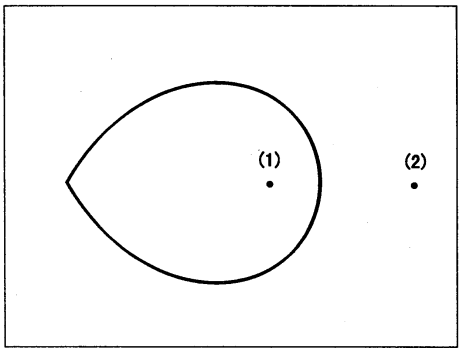


Figure 3: Multiplicatively weighted crystal Voronoi diagram.

Let P_1 and P_2 be two generators with the weights (i.e., the growth speeds) v_1 and v_2, respectively. Without loss of generality we assume that $v_1 < v_2$. Let us define $k \equiv v_2/v_1$. Moreover, let the coordinates of the generators be $\mathrm{P}_1 = (0,0)$ and $\mathrm{P}_2 = (a,0)$, as shown in Fig. 4.

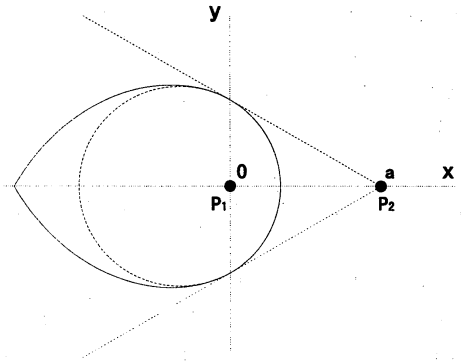


Figure 4: Analysis for the two-side case.

First, we consider the portion of the boundary to which both of the crystals reach without obsecured by the other. Point P on this portion of the boundary satisfies

$$\|\mathrm{P} - \mathrm{P}_i\|/v_i = \|\mathrm{P} - \mathrm{P}_j\|/v_j. \qquad (3)$$

Let the coordinates of P be (x,y). Then, this

equation can be rewritten by

$$\left(x - \frac{a}{k^2-1}\right)^2 + y^2 = \frac{k^2}{(k^2-1)^2}a^2. \quad (4)$$

This is the circle formed by the points from which the distance to P_i and that to P_j have the consstant ratio k. This circle is called the Appolonius circle [7].

Let C_1 be the Appolonius circle represented by eq. (4). Form P_2, let us draw the two tangent lines to C_1. Then, they touch C_1 at two points, say Q_1 and Q_2. Let Q_1 be the tangent point in the $y > 0$ area, and Q_2 be the tangent point in the $y < 0$ area. Both Q_1 and Q_2 are on the y axis. Hence, any point P on C_1 that satisfies $x \geq 0$ is visible from both P_1 and P_2, that is, the line segment connecting P to P_1 and that connecting P to P_2 do not intersect C_1 except at P. This means that the portion of C_1 that satisfies $x \geq 0$ belongs to the boundary of the crystal Voronoi diagram.

On the other hand, the portion of C_1 that satisfies $x < 0$ is not on the boundary of the crystal Voronoi diagram, because this portion is not visible from P_2. In the area $x < 0$, the crystal starting at P_2 should go around avoiding the other crystal. Let this portion of the boundary of the crystal Voronoi diagram be $r(\theta)$ represented by the polar coordinate system, that is, $r(\theta)$ be the distance from the origin to the boundary point in the direction that forms angle θ, $\pi/2 < \theta < \pi$, with respect to the positive x axis.

The curve $r(\theta)$ satisfies

$$\int_{\pi/2}^{\theta} \sqrt{r(\phi)^2 + r'(\phi)^2}\mathrm{d}\phi + \frac{ak}{\sqrt{k^2-1}} = k \cdot r(\theta). \quad (5)$$

Suppose that we devide the both sides of this equation by v_2. Then, the first term in the left-hand side represents the weighted length from Q_1 to $r(\theta)$ along the boundary of the crystal Voronoi diagram, and the second term in the left-hand side represents the weighted distance from P_2 to Q_1. Hence, the left-hand side represents the weighted distance from the generator P_2 to the point $r(\theta)$ along the shortest path avoiding the other crystal area. The right-hand side, on the other hand, represents the weighted distance from P_1 to $r(\theta)$, that is, the weighted length of the line segment from P_1 to $r(\theta)$. Since the two crystals meet at $r(\theta)$, their weighted distance from P_1 and P_2, respectively, should be the same, and hence eq. (5) should be satisfied.

Since the curve $r(\theta)$ should pass through the point Q_1, we get

$$r(\pi/2) = \frac{a}{\sqrt{k^2-1}}. \quad (6)$$

From eqs. (5) and (6), we obtain

$$r(\theta) = \frac{a}{\sqrt{k^2-1}} \exp \frac{\theta - \pi/2}{\sqrt{k^2-1}}. \quad (7)$$

This equation represents a logarithmic spiral centered at P_1 [8].

Since the boundary of the crystal Voronoi diagram is symmetric with respect to the x axis, we obtain the portion of the boundary for $\pi < \theta < 3\pi/2$ in a similar manner.

Summing up all the above discussions, we get the boundary of the crystal Voronoi diagram as

$$r(\theta) = \frac{a}{\sqrt{k^2-1}} \exp \frac{\theta - \pi/2}{\sqrt{k^2-1}} \quad (\pi/2 < \theta < \pi), \quad (8)$$

$$r(\theta) = \frac{a}{\sqrt{k^2-1}} \exp \frac{3\pi/2 - \theta}{\sqrt{k^2-1}} \quad (\pi < \theta < 3\pi/2), \quad (9)$$

$$\left(x + \frac{a}{k^2-1}\right)^2 + y^2 = \frac{k^2}{(k^2-1)^2}a^2 \quad (0 \leq \theta \leq \pi/2, 3\pi/2 \leq \theta < 2\pi). \quad (10)$$

This is the analytic expression of the boundary between the two crystals.

4 Simulation of the Crystal Growth

It is difficult to obtain the boundary for three or more crystals in the analytic form. 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.

4.1 Fast Marching Method

4.1.1 Eikonal Equation

Let $\Omega \subset \mathbf{R}^2$ be a bounded region in the plane, and Γ be its boundary. Let $F(\boldsymbol{x})$ be a real-valued functions satisfying $F(\boldsymbol{x}) > 0$ for any $\boldsymbol{x} \in \Omega$.

Furthermore, let $g(\boldsymbol{x})$ be a function on Γ. We consider a nonlinear partial differential equation

$$|\nabla u(\boldsymbol{x})| = F(\boldsymbol{x}) \quad \text{in } \Omega \tag{11}$$

with a boundary condition

$$u(\boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{on } \Gamma, \tag{12}$$

where $F(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are known and $u(\boldsymbol{x})$ is unknown. The equations (11) is called the *Eikonal equation.*

Assume that $1/F(\boldsymbol{x})$ represents the speed of a moving object at point $\boldsymbol{x}$ in Ω, and that $g(\boldsymbol{x}) = 0$ on Γ. Then, the solution $u(\boldsymbol{x})$ 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 $\boldsymbol{x}$. Therefore, we can use this equation to represent the behavior of the growth of a crystal. In particular, if $F(\boldsymbol{x}) = \infty$ in some area, this area behaves as an obstacle because the speed (i.e., $1/F(\boldsymbol{x})$) 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 $g(\boldsymbol{x}) = 0$ on Γ.

To solve the equation (11) together with the boundary condition (12), Sethian [4, 5] proposed a finite-difference method, called the *fast marching method.* In the finite-difference method, the unknown continuous function $u(\boldsymbol{x}) = u(x, y)$ is replaced by a finite set of values at discretized points

$$u_{i,j} = u(i\Delta x, j\Delta y), \tag{13}$$

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 of the reach time.

Apparently similar techniques have already been used in digital picture processing; they are called *distance-transformation* methods [9]. In these methods, the distance values at discretized points are computed one by one by checking the four-neighbor points (i.e., the immediately left, right, upper, and lower points) or the eight-neighbor points (i.e., the four-neighbor points plus the left-upper, right-upper, left-lower, and right-lower points). 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 [10, 11, 12, 13], but they cannot treat the obstacles, and hence cannot be applied to our purpose.

4.1.2 Finite-Difference Equation in the First Marching Method

Using the discretized value $u_{i,j}$, Sethian proposed finite-difference approximations of the equation (11). The most basic approximation is the first-order finite-difference equation defined by

$$\begin{aligned} & [\max(D_{i,j}^{-x}u, -D_{i,j}^{+x}u, 0)^2 \\ & + \max(D_{i,j}^{-y}u, -D_{i,j}^{+y}u, 0)^2]^{1/2} \\ = \; & F_{i,j}, \end{aligned} \tag{14}$$

where

$$D_{i,j}^{-x}u = \frac{u_{i,j} - u_{i-1,j}}{\Delta x}, \tag{15}$$

$$D_{i,j}^{+x}u = \frac{u_{i+1,j} - u_{i,j}}{\Delta x}, \tag{16}$$

$$D_{i,j}^{-y}u = \frac{u_{i,j} - u_{i,j-1}}{\Delta y}, \tag{17}$$

$$D_{i,j}^{+y}u = \frac{u_{i,j+1} - u_{i,j}}{\Delta y}, \tag{18}$$

$$F_{i,j} = F(i\Delta x, j\Delta y). \tag{19}$$

The reason why the maximum is taken in eq. (14) is the following. Recall that the solution $u_{i,j}$ can be interpreted as the shortest time at which an object starting from the boundary Γ reaches the point $(i\Delta x, j\Delta y)$. Suppose that this object passes through the point $(i\Delta x, j\Delta y)$ in the positive x direction. Then, $D_{i,j}^{-x}u > 0$ and $-D_{i,j}^{+x} < 0$, and hence $D_{i,j}^{-x}u$ is taken by the first maximum in the left-hand side of eq. (14). On the other hand, if the object passes through $(i\Delta x, j\Delta y)$ in the negative x direction, then $D_{i,j}^{-x} < 0$ and $-D_{i,j}^{+x}u > 0$ and consequently $-D_{i,j}^{+x}$ is taken. If the object motion has no x component at $(i\Delta x, j\Delta y)$, then $D_{i,j}^{-x}u = D_{i,j}^{+x}u = 0$. The second maximum in eq. (14) behaves similarly if we replace the x direction with the y direction. In this way, the maximums in eq. (14) are intended to choose the upwind finite difference to approximate $|\nabla u(\boldsymbol{x})|$ in eq. (11). This is reasonable because we want to compute the

time along the shortest path from the boundary to each point $(i\Delta x, j\Delta y)$.

Eq. (14) is used to compute the unknown value $v_{i,j}$ from given u values at the upwind neighbor points and given $F_{i,j}$.

Sethian also proposed the second-order approximation of eq. (11) by

$$\left(\begin{array}{c} \max[[D_{i,j}^{-x}u + switch_{i,j}^{-x}\frac{\Delta x}{2}(D_{i,j}^{-x})^2 u], \\ -[D_{i,j}^{+x}u + switch_{i,j}^{+x}\frac{\Delta x}{2}(D_{i,j}^{+x})^2 u], 0]^2 \\ + \\ \max[[D_{i,j}^{-y}u + switch_{i,j}^{-y}\frac{\Delta y}{2}(D_{i,j}^{-y})^2 u], \\ -[D_{i,j}^{+y}u + switch_{i,j}^{+y}\frac{\Delta y}{2}(D_{i,j}^{+y})^2 u], 0]^2 \end{array} \right)^{\frac{1}{2}} = F_{i,j} \tag{20}$$

where

$$switch_{i,j}^{\pm x} = \begin{cases} 1, & \text{if } u_{i\pm 2,j} \text{ and } u_{i\pm 1,j} \text{ are} \\ & \text{known and } u_{i\pm 2,j} \le u_{i\pm 1,j}, \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

and $switch_{i,j}^{\pm y}$ is defined similarly.

The coefficient *switch* in eq. (20) is necessary, because $F(\boldsymbol{x})$ depends on $\boldsymbol{x}$ 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 $u_{i,j}$.

4.1.3 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. (20) 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. (20). (If the point already has the u value, recompute and update it.)

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, 5] for the details of this algorithm.

4.2 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 $\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_n$. Let N be the total number of the grid points. We assign sequential numbers to all the grid points, and name them as $\mathrm{Q}_1, \mathrm{Q}_2, \cdots, \mathrm{Q}_N$. 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, $k = 1, 2, \cdots, n$, set CNAME[Q_j] $= k$ for grid point Q_j that is one-grid far from P_k, and set CNAME[Q_j] =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 $\{\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_n\}$ 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. (20).

(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. (20).

(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. 5 shows the behavior of the algorithm. Here, the square region was replaced by 400×400 grid points and 15 generators were placed. Fig. 5 (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. 5 (c) shows the final result.

5 A Generalization of the Crystal Voronoi Diagram

The distance in the multiplicatively weighted Voronoi diagram is defined along a straight line segment no matter whether it crosses other regions, while in the crystal Voronoi diagram the distance is measured along the path completely included in one crystal region. We can consider a mixture of these two distances and thus can define a new type of a generalization of the Voronoi diagram.

Suppose that, instead of crystals, different species of plants start growing at $P_1, P_2, \cdots, P_n$ in a field, and that they can grow at their own speeds in an empty space whereas at slower speeds when they enter regions with other plants. We partition the field into regions according to which plant reaches first. We call this partition a *generalized crystal Voronoi diagram.*

The speed of growth can be formulated in the following way. Suppose that the plants starting from $P_{m_1}, P_{m_2}, \cdots, P_{m_n}$ reach point P in this order. Then, we define the speed at P of the plant starting from P_{m_k} as

$$f_{m_k}(P) = f(P_{m_1}, P_{m_2}, \cdots, P_{m_k}). \quad (22)$$

This means that the speed at the point P of the plant growing from P_{m_k} depends also on the plants that have already reached P.

Note that, if

$$f_{m_k}(P) = \begin{cases} f(P_{m_k}) & (k = 1), \\ 0 & (k \neq 1), \end{cases} \quad (23)$$

then the resulting partition is the crystal Voronoi diagram, whereas if

$$f_{m_k}(P) = f(P_{m_k}) \quad \text{for any } k, \quad (24)$$

then the resulting partition is the multiplicatively weighted Voronoi diagram. Thus, our new diagram is a generalization of both the crystal Voronoi diagram and the multiplicatively weighted Voronoi diagram.

The new diagram can also be computed approximately by our modified fast marching method; the only change is to substitute our new grow speed to $1/F_{i,j}$ in eq. (20).

Examples of the generalized crystal Voronoi diagrams are shown in Fig. 6. In this example, we use the speed defined by

$$f_{m_k}(P) = \begin{cases} f(P_{m_1}) & (k = 1), \\ a \cdot f(P_{m_1}) & (k \neq 1). \end{cases} \quad (25)$$

The diagrams (a), (b), $\cdots$, (k) in this figure correspond to the cases $a = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4$, respectively. The diagram with $a = 0.0$ coincides with the crystal Voronoi diagram, and the diagram with $a = 1.0$ coincides with the multiplicatively weighted Voronoi diagram. It might be interesting to note that the change of the diagram is not gradual, but is drastical around some value of a (around $a = 0.8$ in the example in Fig. 6).

6 Application to Path Planning

6.1 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 the Sethian's idea [4].

The Eikonal equation (11) can be written in the integral form as

$$u(\boldsymbol{x}) = \min_{\gamma} \int_A^{\boldsymbol{x}} F(\gamma(\tau))\mathrm{d}\tau, \quad (26)$$

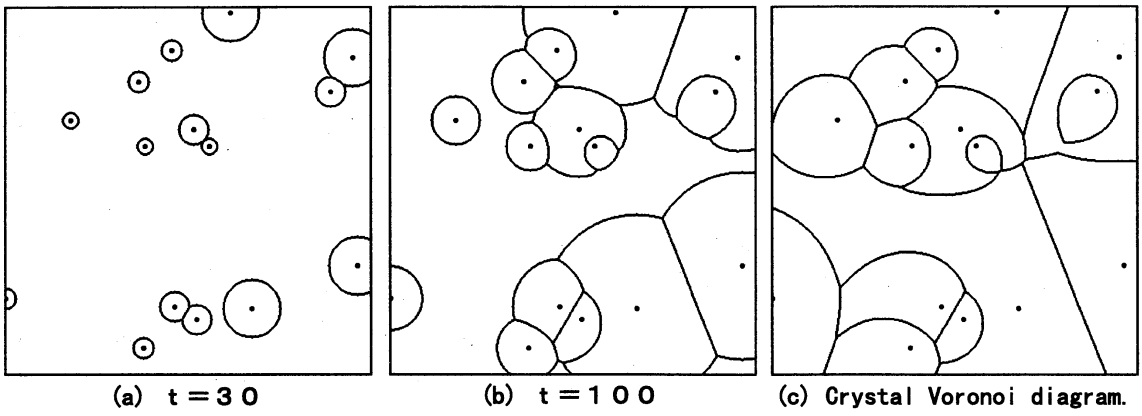


Figure 5: Simulation of crystal Voronoi diagram by the fast marching method (t means the radius of the fastest growing crystal when the width betwen grids is one).

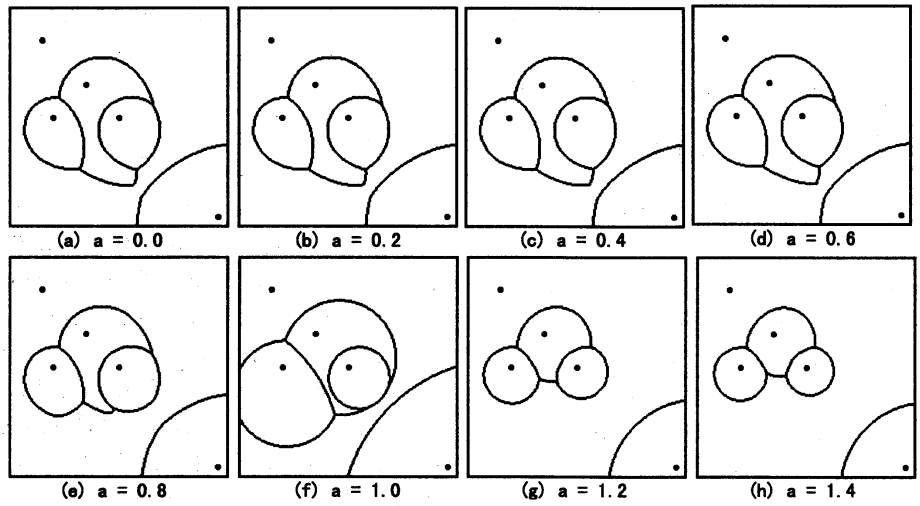


Figure 6: Generalized crystal Voronoi diagrams growing at the a times faster velocity in other crystals.

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

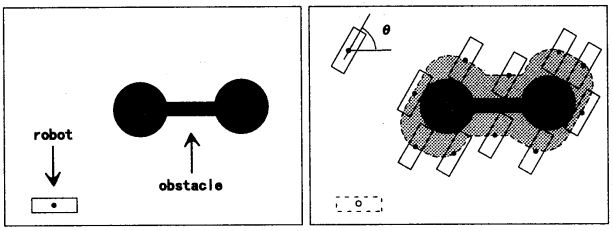
$$X(t) = -\nabla u, \quad X(0) = B \tag{27}$$

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 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. 7. The boundary of this region can be obtained as the tragectory 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.



(a) First state of a robot and an obstacle. (b) The area where the robot's center cannot enter when it rotates at an angle of θ.

Figure 7: The area where where the robot's center cannot enter.

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. 8. A fixed value of θ corresponds to a horizontal plane, in which we extend the obstacles.

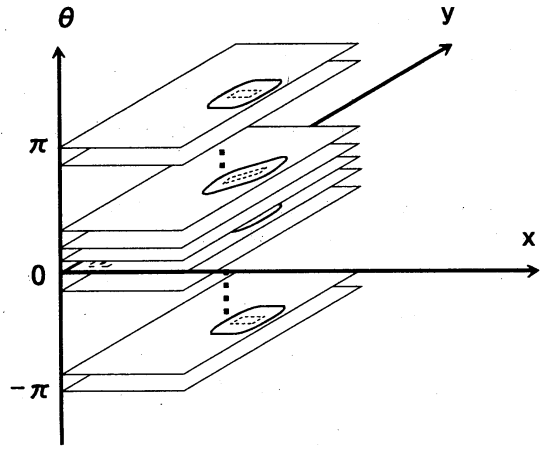


Figure 8: 3-dimensional space of fast marching method for robot navigation.

Sethian used the fast marching method to solve the Eikonal equation

$$\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \alpha\left(\frac{\partial u}{\partial \theta}\right)^2\right]^{1/2} = 1 \quad (28)$$

in the three-dimensional (x, y, θ) space. The partial derivatives $\partial u/\partial x$ and $\partial u/\partial y$ represent the x and y components of the velocity while $\partial u/\partial \theta$ represents 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.

6.2 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, 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. 9 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. 9, (a), (b) and (c) show the status at some instants, while (d) shows the whole path of the robot to reach the goal.

Fig. 10 (a) shows the generated path for the case that our robot can move faster than in Fig. 9, while Fig. 10 (b) shows the case that our robot moves more slowly than in Fig. 9.

7 Concluding Remarks

This paper studied the crystal Voronoi diagram from the computational point of view. First, we gave an explicit expression of the boundary of two crystals. Next, we present a method for computing the approximated diagram, where we modify the fast marching method to solve the Eikonal equation.

The approximation method proposed by Schaudt and Drysdale [1] requires $O(n^3)$ time

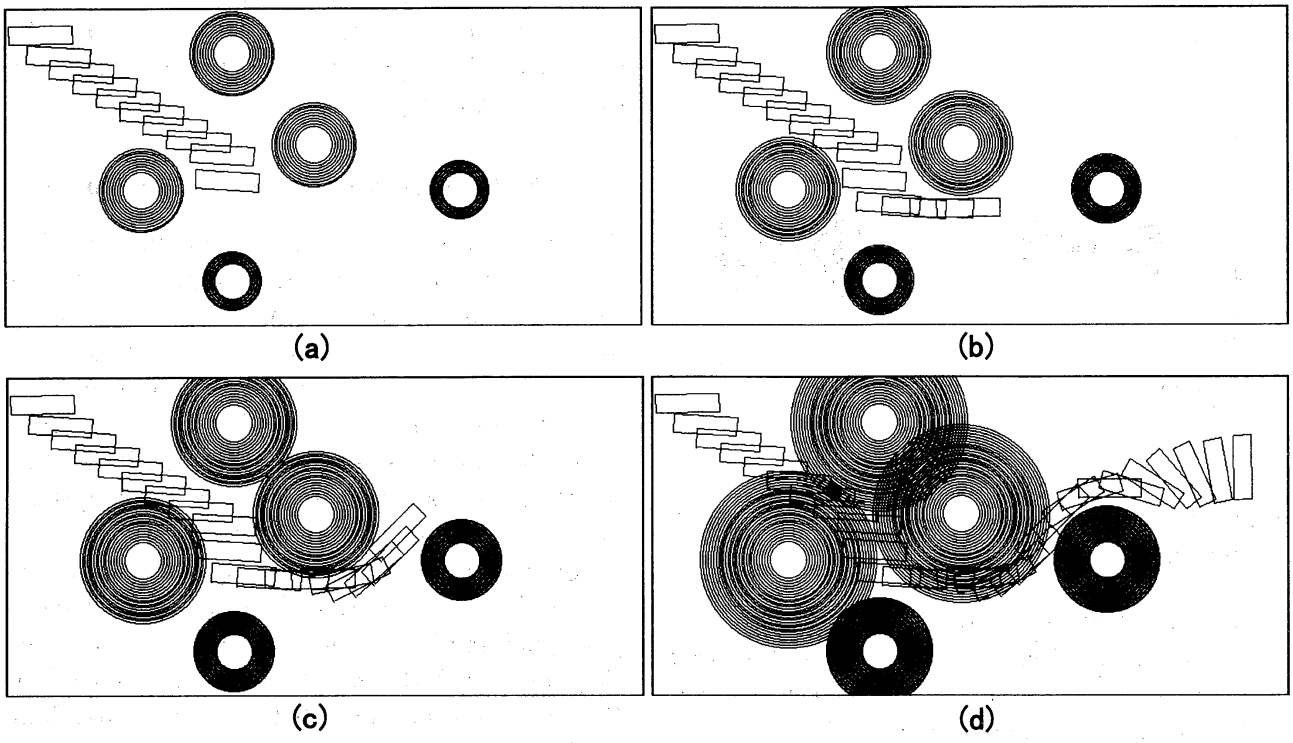


Figure 9: Optimal answers of the robot navigation problems.

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. Hence our new method will be more efficient for a large number of crystals.

Furthermore, we extend our method in two directions. First, we generalize the crystal Voronoi diagram in such a way that the crystals can grow also in other crystal areas though the speed might be slower. We also generalize our method to compute this Voronoi diagram. Secondly, we apply 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 couse grid together with interpolation techniques. We might also decrease the memory cost by discurding the u values except around the frontiers of the crystals.

In our application to the path planning among competitive robots, we assume that the enemy robots are circles in their shape. To generalize our method for arbitrary enemy shapes is another important problem for future. A naive method is to increase the dimension of the search space, one for each enemy robot, but this is not a cleaver strategy from a computational-cost point of view.

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 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 Second Canadian Conference in Computational Geometry*, North Conway, (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.

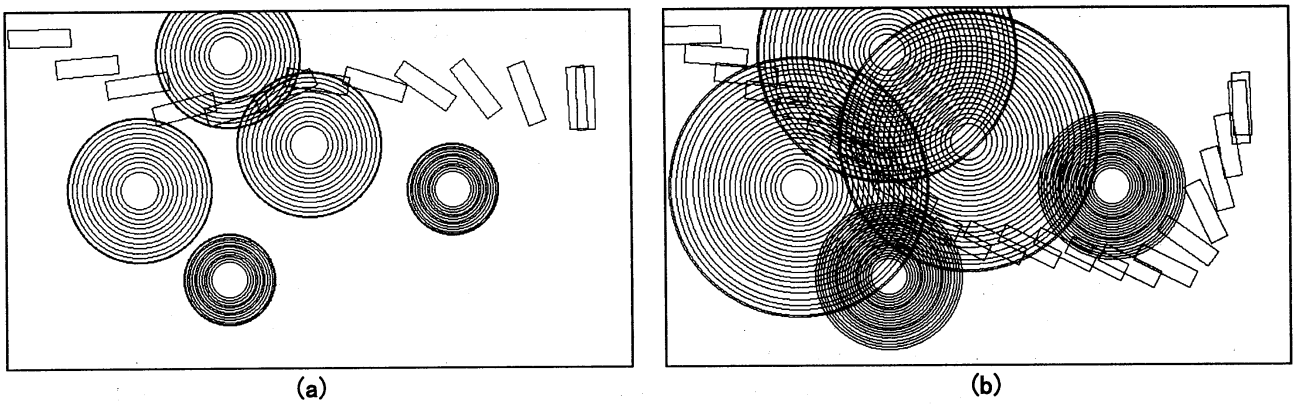


Figure 10: Optimal answers of the robot navigation problems for other robot velocities.

[3] A. Okabe, B. Boots, and K. Sugihara: *Spatial Tesellations—Concepts and Applications of Voronoi Diagrams.* John Wiley, Chickester, 1992.

[4] J.A. Sethian: Fast marching methods. *SIAM Review*, vol. 41, no. 2 (1999), pp. 199–235.

[5] J.A. Sethian: *Level Set Methods and First Marching Methods, Second Edition.* Cambridge University Press, Cambridge, 1999.

[6] 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.

[7] D. Pedoe: *Geometry—A Comprehensive Courese.* Cambridge University Press, London, 1970.

[8] M. Berger and B. Gostiaux: *Differential Geometry—Manifolds, Curves, and Surfaces.* Springer-Verlag, New York, 1988.

[9] A. Rosenfeld and J. Pfalts: Sequential operations in digital picture processing. *Journal of ACM*, vol. 13 (1966), pp. 471–494.

[10] M.N. Kolountzalcis and K.N. Kutulakos: Fast computation of the Euclidean distance maps for binary images. *Infor. Process. Lett.*, vol. 43 (1992), pp. 181–184.

[11] L. Chen and H.Y.H. Chuang. A fast algorithm for Euclidian distance maps of a 2-d binary image. *Infor. Process. Lett.* vol. 51 (1994), pp. 25–29.

[12] H. Breu, J. Gill, D. Kirkpatrick and M. Werman: Linear time Euclidean distance transform algorithms. *IEEE Transactions on pattern Analysis and Machine Intelligence*, vol. 17 (1995), pp. 529–533.

[13] T. Hirata: A unified linear-time algorithm for computing distance maps. *Infor. Process. Lett.*, vol. 58 (1996), pp. 129–133.