

# Asynchronous Computing of a Discrete Voronoi Diagram on a Cellular Automaton Using 1-Norm: Application to Roadmap Extraction

Nassim Kaldé

Inria, Villers-lès-Nancy, 54600, France.

Université de Lorraine,

LORIA, CNRS UMR 7503,

Vandoeuvre-lès-Nancy, 54506, France.

e-mail: nassim.kalde@loria.fr

Olivier Simonin

INSA-Lyon, CITI-INRIA,

Université de Lyon,

Villeurbanne, 69621, France.

e-mail: olivier.simonin@insa-lyon.fr

François Charpillet

Inria, Villers-lès-Nancy, 54600, France.

Université de Lorraine,

LORIA, CNRS UMR 7503,

Vandoeuvre-lès-Nancy, 54506, France.

e-mail: francois.charpillet@loria.fr

**Abstract**—This article addresses the problem of computing a Voronoi diagram in a distributed fashion without any synchronization heuristic. To our knowledge, no previous work asynchronously solves this problem. We investigate a simple case of asynchronism and tackle this challenge in a decentralized fashion on a grid of communicating cells with a von Neumann neighborhood. We describe algorithms for extracting single site, area and pseudo line Voronoi diagrams. These algorithms are implemented and executed on maps in which we consider different kinds of sites defined as simple polygonal shapes to extract roadmaps.

**Keywords**—Voronoi diagram; asynchronous computing; cellular automata; distributed systems; spatial computing;

## I. INTRODUCTION

Ambient intelligence [1] results from the interaction of numerous devices. Such interactions take place in sensor networks on which embedded processing units run and cooperate with each other. Self-organizing approaches can simplify the design of such systems when it comes to scalability and robustness. This is the objective of spatial computing [2] where the data are distributed in a network of processing units and the computing can be done in various synchronization modes. We will take advantage of the powerful and general Cellular Automaton model to represent our distributed network.

We address the problem of computing a Voronoi Diagram on this type of sensor network. A Voronoi diagram can be defined as the set of regions associated to sites. Each region represents the set of points that are closer to one site than any other site. Every node in the network has a local perception of the sites dataset and communicates with its neighboring nodes to compute a part of the final diagram.

Several studies already deal with distributed computing of Voronoi Diagrams, but most of them address this problem with an embedded synchronization, *e.g.* parallel or sequential computing in predefined orders of the units. These execution orders imply full control over the nodes. This assumption has several drawbacks, such as scalability and robustness. If a single unit fails to communicate, the overall process can

suffer from this situation. Moreover, synchronizing a high number of units may be really difficult. For the purpose of scalability and robustness, our approach considers no synchronization heuristic for computing a Voronoi Diagram.

Our study focuses on an *asynchronous mode* in which every unit is selected according to a uniform distribution. We also define a *synchronous mode* in which all units compute at the same time. We aim to design simple mechanisms that exhibit the same behavior whatever the synchronization and thus can be considered synchronization-proof.

Our contribution to compute Voronoi diagrams in "asynchronous" mode and using 1-norm is inspired by sequential image processing as it requires to build a convergent labeled distance transform. Section II defines the sensor network formalization, the synchronization modes, and the Voronoi diagram. Parallel or sequential computation of Voronoi tessellations, from the fields of Cellular Automata and Image Processing are discussed in Section III. Our approach to compute Voronoi diagrams in "asynchronous" mode is presented in Section IV. In Section V, we extend our study to new types of sites and provide some extracted roadmaps. We conclude with a short table comparing several methods.

## II. BACKGROUND

A typical 2D Cellular Automaton (CA) is defined as a tuple  $\mathbb{F} = \langle A, Q, u, f \rangle$ , with  $A$  a matrix of cells (dimension  $w \times h$ ),  $Q$  the set of cell states and  $u : A \rightarrow A^k$  a cellular neighborhood function (*e.g.*  $u_1$  von Neumann neighborhood: itself, North, East, South and West). We consider a cell  $c \in A$  and its state  $c^t \in Q$  at time  $t$ . The automaton evolves in discrete time as the cells  $c$  apply a local rule  $f : Q^k \rightarrow Q$  which considers the current neighbors state  $u^t$  to compute their next state  $c^{t+1} \leftarrow f(u^t(c))$ .

Synchronizing a CA consists in choosing which cells update their state at each time step. The first type of synchronization we consider is termed "synchronous": all cells simultaneously compute the next state of the automaton. The second one is termed "fully asynchronous": one cell randomly chosen contributes to the next automaton state.

Our definition of the discrete Generalized Voronoi Diagram (VD) is taken from [3] and adapted to our CA. The VD is a tessellation of the matrix of cells  $A$  in Voronoi regions associated to Voronoi sites. Each Voronoi region  $VR(S_i)$  defines the set of cells which are closer to the site  $S_i$  than any other site  $S_j$ . The distance used can be any metric (e.g. Minkowski distances) and a site is defined as a subset of the lattice (e.g. contiguous cells).

### III. RELATED WORK

In 1964-67, Blum introduced a new thinning shape descriptor, the Medial Axis Transform (MAT) [4]. The first MAT definition referred to as the "grassfire metaphor" is time-based: fire fronts propagate from fire sources (shape contour) and travel at constant velocity across the grass field (into the shape). Two fire fronts annihilate each other when they meet on the medial axis. Distance-based definitions were also provided: if we consider a distance field generated from the shape contour, the MAT is defined alternatively as the locus of derivative discontinuities, centers of maximal disks, or minimum distances to many contour points.

#### A. Cellular Automata

In the field of cellular automata, the first spatio-temporal definition of MAT (grassfire metaphor) naturally translates to synchronous evolution rules. A 1-norm and an  $\infty$ -norm VD were synchronously computed in [5] and then completed in [6] by extracting the discrete bisectors between cellular sites. A 2-norm VD was investigated in [7].

These CA models were readily usable as simplified simulators of natural computing. An interesting retrospective of Voronoi diagram variations computed using intrinsically asynchronous chemical processors and modelled using synchronous CA rules is provided in [8], [9]. Even if these models do not always reflect ground truth, they are meaningful to explain the concepts involved. Moreover several applications of VD, from path planning [10]–[12] and classification [13] to shape recognition [14] are reported in the CA literature.

However, the time-based definition has some drawbacks. Results obtained when simulating the grassfire metaphor using 1-norm for the *synchronous* and *asynchronous* cases are different after convergence (Figure 1). Thus, when it comes to asynchronism, the metaphor can be expressed alternatively as a fire propagation altered by a random wind.

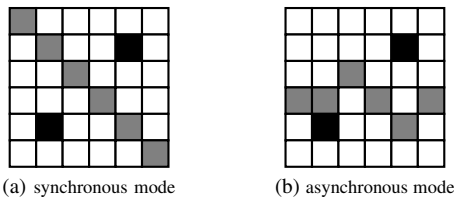


Figure 1. Fire propagation with two sites (black), two burnt regions (white), one discrete bisector (gray)

#### B. Image Processing

In the field of image processing, distance-based definitions were immediately investigated to extract shape skeletons (MAT). In [15], a 1-norm distance transform of the foreground objects in an image was computed using sequential local operations. They were applied on each pixel in forward and backward raster sequences. The distance skeleton was extracted as the set of local maxima pixels. A 2-norm distance transform was used to extract planar skeletons as the set of maximal disks centers in [16].

The Labeled Distance Transform (LDT) proposed in [17], provides semantic information about the contour (coordinates, directions, features) over the distance transform. Thus, the skeleton is the set of pixels with locally varying semantics. This method generalizes extraction and, as such any distance transform can be considered. In [18], a VD was extracted as a pruned 1-norm exoskeleton (skeleton of the background). More recent works take advantage of nowadays available GPU hardware to compute Voronoi diagrams using parallel local rules [19].

Concerning the distance-based definitions of the MAT, a norm-1 integer gradient can be asynchronously computed using a convergent rule found in [20]. In Figure 2a, darker shades of gray represent higher values of this gradient. However, the associated  $u_1$  gradient patterns colored in Figure 2b cannot determine a frontier between the two Voronoi regions. This "wave fusion" phenomenon — two waves collide and then combine — lets the gradient unable to discern a full bisector. Reference [21] reports a solution involving combined *von-Neuman* and *Moore* patterns.

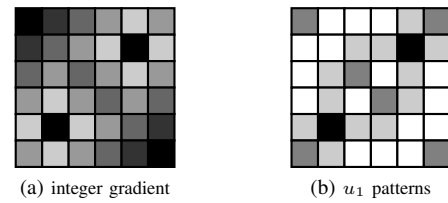


Figure 2. Wave expansion with two sites (black), no discernible bisector

Our work tackles the "random wind" and the "wave fusion" phenomena using only 1-norm and  $u_1$  neighborhood to asynchronously compute a VD. Therefore, we build a site semantic layer on top of a gradient layer as the LDT. This informative gradient is not affected by the asynchronism and provides a correct VD as it converges.

### IV. INFORMATIVE GRADIENT AND BISECTOR EXTRACTION

First we present the rules for computing an informative gradient on the CA. It provides correct Voronoi regions and can be computed using the *synchronous* or *asynchronous* modes. In a second part, we discuss bisector extraction between two Voronoi regions based on semantic information

from the informative gradient. For the sake of simplicity, we deal with a single cell site. Multi-cell sites are addressed in the next section.

#### A. Informative Gradient

The informative gradient is a combination of a distance transform and a semantic overlay. The distance map provides for each cell  $c$  of the automaton, the 1-norm distance to its closest site in  $S$ . The semantic layer provides, for each cell, the identifier of the closest site, thus it determines the Voronoi region.

1) *Distance layer*: First of all, we consider two kinds of cells at each time, the *sites* and the *nonsites*. The objective is to compute the distance for each *nonsite* cell to its closest *site*. The distance transform  $d_1$  is initialized with 0 for the *sites* and a random positive value  $M$  for the *nonsites*. Each cell participates in computing the 1-norm integer gradient using the following rule  $G$  with  $m \leftarrow \min_{k \in u_1(c) \setminus \{c\}} (k.d_1)$ .

$$G(c) : c.d_1 \leftarrow \begin{cases} 0, & \text{if } \text{site}(c) \\ m+1, & \text{otherwise.} \end{cases}$$

When a *site* is selected at time  $t$ , it never updates its distance to the closest *site*. On the opposite, for a *nonsite*, its closest *site* distance  $d_1$  is updated by taking the minimum  $d_1+1$  of its  $u_1$  neighbors.

If we consider static sites, as long as we update every cell often enough, the distance transform  $d_1$  will converge at a time  $T$ . In other words, for each pair of cell and closest site  $(c_{ij}, S_{kl})$ , with  $(i,j)$  and  $(k,l)$  their respective coordinates in  $A$ , we have:  $\lim_{t \rightarrow T} c_{ij}.d_1 = d(c_{ij}, S_{kl}) = |i - k| + |j - l|$ .

If we consider dynamic sites, *e.g.* a site moves on the lattice, some cells will change their site state from *site* to *nonsite* and inversely, and compute  $d_1$  accordingly. The distance transform will eventually converge depending on the speed of change and number of state transitions relative to  $T$ , the correction time of the distance transform.

As an illustration, for the initialization in Fig.3a, the computed gradient is represented after convergence in Fig.3b. It can be computed in the *synchronous* mode as well as the *asynchronous mode*. The wavefront expansion algorithm from [20] is time-step equivalent to our *synchronous* mode but here we allow multiple updates of each cell. Also, their expansion algorithm maintains a list of currently considered cells (the wavefront) at each time step, whereas we do not. Sequential methods from the image processing field would consider multiple scans of the entire automaton in predefined orders. In our *asynchronous mode*, a cell is chosen according to a uniform distribution at each time  $t$ .

2) *Semantic Layer*: Now we incorporate a semantic information related to a cell: its identifier  $id$ . Spreading a *site* identifier across the automaton can help marking the corresponding Voronoi region. For this purpose, we take advantage of the gradient layer to ensure a correct semantic

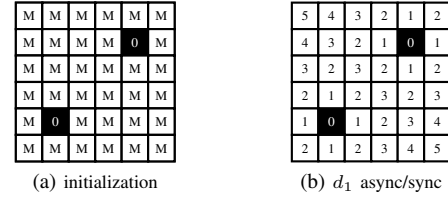


Figure 3. 1-norm integer gradient with sites (black) and nonsites (white)

layer computation. Thus, we use both the *gradient value* ( $d_1$ ) and *identifier* ( $id$ ) as a pair of information to propagate.

Let us initialize every cell  $id$  with a unique constant identifier  $id_i$ . *Sites* reset their  $id$  to  $id_i$  at each step. *Nonsites* collect these *sites* identifiers into a set  $id_c$  by neighboring mechanisms. Collected identifiers are taken from *von-Neumann* neighbors  $k$ , meeting two conditions:  $\delta \leftarrow k.d_1 = m$  (*i.e.* lowest gradient value among the neighbors), and  $\epsilon \leftarrow |k.id| = \min_{k \in u_1(c) \setminus \{c\}} |k.id|$  (*i.e.*  $\delta$ -cells with smallest set of identifiers). Once collected, these identifiers are merged into  $id_c \leftarrow \{k.id | k \in u_1(c) \setminus \{c\} \text{ s.t. } \delta \wedge \epsilon\}$ . Rule  $IG$  synthesizes the gradient and semantic layers computation:

$$IG(c) : (c.d_1, c.id) \leftarrow \begin{cases} (0, id_i), & \text{if } \text{site}(c) \\ (m+1, id_c), & \text{otherwise.} \end{cases}$$

Figure 4 presents the computed *gradient* layer and two *semantic* layers after convergence of rule  $IG$ . Cells are numbered row by row, from left to right to provide unique constant identifiers  $id_i$  for initialization. Then, we wait until the *gradient* has converged (Fig. 4a) to get the good identifiers  $id$  on the lattice. Three regions are noticeable after convergence in Fig.4b and 4c; each region is composed by the cells possessing the same set of identifiers ( $\{11\}$ ,  $\{26\}$  and  $\{11,26\}$ ). The semantic parsimony condition  $\epsilon$  provides larger regions for the *sites* ( $\{11\}$ ,  $\{26\}$ ) and a thin boundary region between them ( $\{11,26\}$ ), see Fig. 4c.

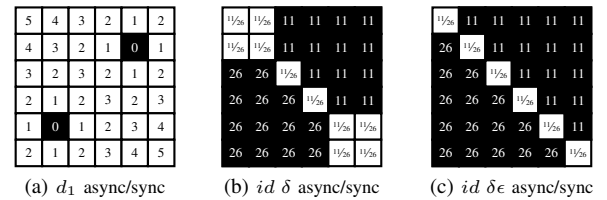


Figure 4. Gradient and Semantic layers with sites 11 and 26

We have designed a rule that can tackle the "random wind" and "wave fusion" phenomena using a two layer CA in the *asynchronous* mode. The first layer of the automaton, the *distance layer* computes a 1-norm integer gradient from the *site* cells. The second layer, the *semantic layer* can then correct the identifier propagation until convergence of the distance map. Based on this synchronization-proof rule  $IG$ , we will now study bisector extraction between two sites.

## B. Bisector Extraction

1) *Intuition*: An intuitive way to extract the frontier between two Voronoi regions is by marking the boundaries of these regions. Thus a *nonsite* possessing a different identifier from at least one of its neighbors is a frontier, see the skeleton generation algorithm from [17]. The following rule *Bis* formalizes this intuition:

$$Bis_{thick}(c) : c.bis \leftarrow \begin{cases} 1, & \text{if } (\exists k \in u_1(c)/k.id \neq c.id) \\ 0, & \text{otherwise.} \end{cases}$$

In order to exhibit each kind of bisector we reproduce the examples from [5]. Six cases are generated by varying the spacing between two simple sites according to odd or even 1-norm distance. Let us consider two sites  $a$  and  $b$ , and their horizontal and vertical distances  $s_x = |x_a - x_b|$ , and  $s_y = |y_a - y_b|$ . If  $s_x = s_y$ ,  $s_x$  is either an odd or an even distance (2 cases). Otherwise, we have  $|\{odd, even\}^2| = 4$  other cases. Bisectors extracted using *Bis<sub>thick</sub>* are given in Figure 5.

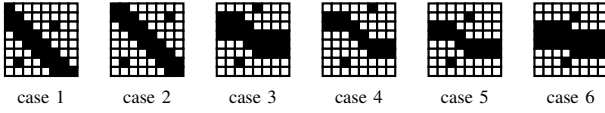


Figure 5. Thick Bisector Extraction Cases for two sites

Cases 1, 2, 3 and 6 are for odd distances between the sites; the bisector has a thickness of three cells. For even distances between the sites, the bisector has a thickness of two cells. A complete bisector is extracted for all cases but is too thick for odd distances. Therefore, we design a thin bisector extraction rule which exploits more of the available semantics. This rule is based on the discrete bisector definition.

2) *Definition and Thin Bisector*: We refer to [5], where the discrete bisector *Bis*, of cells  $a$  and  $b$ , is defined as the set of cells  $c$  such that the 1-norms  $d(a, c)$  and  $d(b, c)$  differ by at most 1.

$$Bis(a, b) \leftarrow \{c \in A : |d(a, c) - d(b, c)| \leq 1\}$$

We can discern two types of bisector cells. The first type corresponds to *nonsites* exactly equidistant to several *sites*. The second type detects (modulo 1)-equidistance to multiple *sites*. From a cell-centered view, first-type bisector cells collected several *site* identifiers (set *bis* to 1). Second-type bisector cells collected only one identifier and have a neighbor which owns only one, but different, identifier (set *bis* to 2). Now, we can introduce rule *Bis<sub>thin</sub>* for extracting thin bisectors.

$$Bis_{thin}(c) : c.bis \leftarrow \begin{cases} 1, & \text{if } |c.id| > 1 \\ 2, & \text{if } |c.id| \leq 1 \\ & \wedge (\exists k \in u_1(c)/|k.id| = 1 \\ & \wedge k.id \neq c.id) \\ & \wedge (\nexists k \in u_1(c)/site(k)) \\ 0, & \text{otherwise.} \end{cases}$$

Figure 6 illustrates thin bisector extraction for the cases of two-sites spacing. Thinner bisectors are extracted for cases 1, 2, 3 and 6 compared to the previous intuitive rule. Every bisector has the minimum thickness allowed in our model.

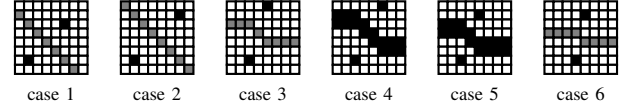


Figure 6. Thin Bisector Extraction Cases for two sites (black dots), cells with bis 1 are in black and cells with bis 2 in gray.

Based on the *informative gradient* and the *complete bisectors*, we can already compute simple cellular sites *VD* in a distributed manner without synchronization requirements. To recapitulate this section, we provide an example of such computation in Figure 7. Sites are represented as black cells in Figure 7a. The *gradient layer* from each site is updated in a distributed fashion using rule *IG* (Fig.7b). The *semantic layer* is updated according to the current *gradient layer*, each color represents an identifier. Identically colored cells share the same identifier and form a Voronoi region (Fig.7c). Based on this *informative gradient*, bisector cells are identified locally using the intuitive rule *Bis<sub>thick</sub>* or the thinner rule *Bis<sub>thin</sub>* (Fig.7d,7e). The Voronoi diagram and the bisectors are correct as soon as the gradient layer converges to the real closest site distances. Moreover, the *synchronous* and *asynchronous* modes provide the same results when the gradient converges.

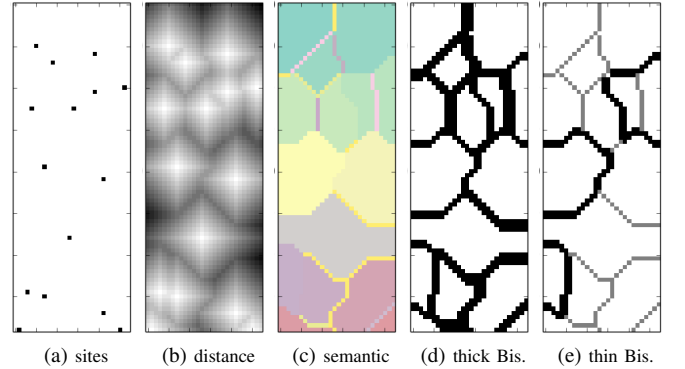


Figure 7. Discrete Voronoi Diagram Extraction with cell sites

Next section extends these principles for extracting Area Voronoi Diagrams and Line Voronoi Diagrams. Our extension considers sites of contiguous cells forming areas or simple polygonal edges. We show how these contiguous regions are discovered locally by neighboring mechanisms. And also how the cells in a same region exhibit a consensus-decision making while computing a common identifier.

## V. APPLICATION

First, we present the typical maps on which we want to extract maximum clearance roads for robotic navigation. Then, we extend our study step by step from cellular sites to area sites and then to line sites Voronoi Diagrams. To achieve this, new rules are required to allow each cell from a same generalized site to propagate a unique semantic. In other words, contiguous site cells will decide on a common identifier to propagate as one single entity on the lattice.

### A. Simulator and Maps

The rules for identification, propagation and bisector extraction presented in this study were implemented using Python and executed in the *synchronous* and *asynchronous* modes.

Some of the maps used for the experiments in Figure 8 are reproduced from [5], [10]. These maps exhibit multiple obstacles, narrow passages, and maze-like structures with concavities. We first show how Area Voronoi Diagrams will help to create simple roadmaps but are not satisfying for maze-like structures and concavity.

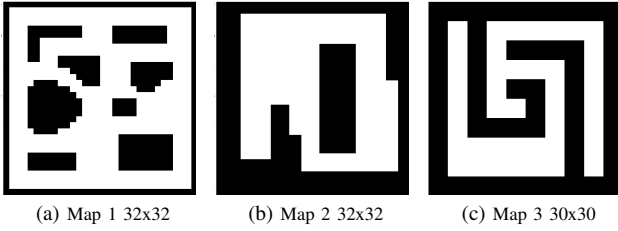


Figure 8. Maps (Lattices) with pixel obstacles (cell sites) in black

### B. Area Voronoi Diagram

To compute an Area Voronoi Diagram (AVD) [3], we now consider *site areas* aggregating contiguous *cellular sites*.

1) *Area Identification*: We design a component labeling procedure to identify *areas* of contiguous *cellular sites*. We assume each cell has a unique initial identifier  $id_i$  and the set of unique identifiers is totally ordered. The site area identifier is locally computed using rule  $AId$  by retaining the minimum identifier  $id_a$  of the neighboring sites ( $id_a \leftarrow \min_{k \in u_1(c); site(k)}(k.id)$ ).

$$AId(c) : c.id \leftarrow \begin{cases} c.id, & \text{if } \neg site(c) \\ id_a, & \text{otherwise.} \end{cases}$$

The *area identification consensus* terminates when every cell in a same area shares a common identifier.

2) *Algorithm*: When a cell is selected, it executes three steps: the first one is the *area identification*, which is followed by the *propagation*; and finally the *bisector extraction* (resp. processed by the rules  $AId$ ,  $IG$  and  $Bis$ ). An intermediate step is introduced to allow a site cell to update

---

### Algorithm 1 Area Voronoi diagram (AVD)

---

```

AId(c)
if site(c) then
    c.id ← c.id {update initial to area identifier}
end if
IG(c)
Bis(c)

```

---

its initial identifier as the newly computed area identifier. Thus, each cell executes Algorithm 1 when selected.

This algorithm was executed by every cell on the three lattices from Fig.8. The first lattice provides 9 Voronoi regions corresponding to the 9 areas (border and 8 simple polygons). The second lattice provides 2 regions and the third lattice only 1. These AVDs are complete and expected.

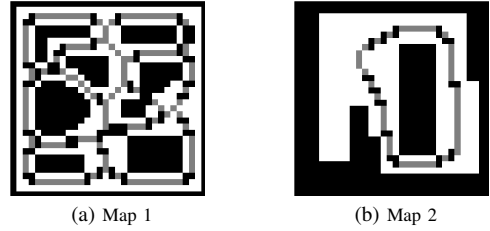


Figure 9. Thin Skeleton for Area Voronoi Diagram

3) *Limitation for robotic navigation*: These examples bring forward the fact that the bisectors extracted using an area Voronoi diagram are not satisfying for establishing a maximum-clearance roadmap useful in robotic navigation. In the context of robotic navigation on the areas bisectors, a robot could not navigate into the concavities of the environment. As presented above, no road was extracted for the maze-like structure in Map 3 and the concavities were ignored in the first two maps. Therefore, we now consider Voronoi diagrams computed for line sites. The idea is to subdivide the boundary of a site area into its edges. Thus, we adapt our current work by designing edge identification rules.

### C. Line Voronoi Diagram

In this section, we compute Line Voronoi Diagrams (LVD) [3]. To achieve this, we generate bisectors between the edges of a shape. First, we discuss *edge patterns* on the contour of a site area. Then we provide *edge identification* mechanisms to reduce the number of unwanted bisectors. Finally we compute LVDs using these corrected identifiers.

1) *Edge Patterns*: In our approach we consider an identifier for each (horizontal, vertical or diagonal) edge of an area. We use edge detectors as in [10], but in strict  $u_1$  neighborhood. Every cell detects its edge pattern  $p$  from one class of the possible  $u_1$  masks, see Figure 10. These classes represent rotational symmetry. However, as mentioned in

[7], using only  $p$  as identifier will result in some artifact bisectors. We aim to reduce the number of such artifacts.

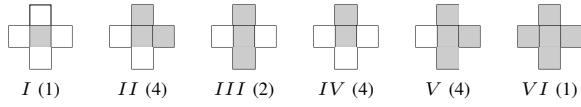


Figure 10. Six classes of site patterns  $p$ , and their cardinality

2) *Edge Identification*: Reducing the number of artifacts involves local detection and correction of conflicting edge patterns causing these artifacts. Conflicts are typical cases of neighboring edge patterns that create unwanted bisectors. If a conflict is locally detected then it is resolved by choosing another identifier instead of the edge pattern  $p$ :

$$EId(c) : c.id \leftarrow \begin{cases} solve(c), & \text{if } conflict(c) \\ c.p, & \text{otherwise.} \end{cases}$$

Conflict resolution is enabled via some cells acting as conciliators ( $V$ ) for two nonneighbor cells, while others alternatively act as yielders or cooperators ( $II$ ,  $IV$ ). Detailed conflict detection and resolution are given in Fig.13.

In Figure 11, three conflicts ( $II/IV$ ), ( $IV/V$ ) and ( $III/IV$ ) are illustrated. If each cell transmits its edge pattern on the lattice, it will result in a high number of bisector cells (drawn in black in Fig. 11a). But, if some cells provide different identifiers according to the directions, we can solve the local conflicts and reduce the number of artifacts. Cells from classes  $II$  and  $IV$  exhibit such behavior, see Fig.11b.

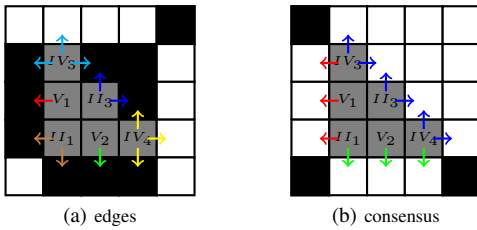


Figure 11. Conflicting edge patterns and resolution

3) *Algorithm*: By replacing  $AId$  with  $EId$  in Algorithm 1, we are able to compute  $LVD$ s. The extracted roadmaps are provided in Fig.12, concavities are now entered, and the number of artifacts is reduced.

---

**Algorithm 2** Line Voronoi diagram ( $LVD$ )

---

```

EId(c)
if site(c) then
     $c.id_i \leftarrow c.id$  {update initial to edge identifier}
end if
IG(c)
Bis(c)

```

---

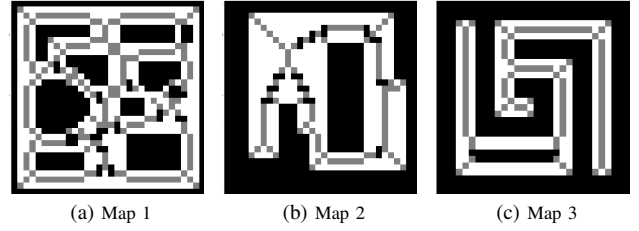


Figure 12. Thin Skeleton for Line Voronoi Diagram

## VI. CONCLUSION

In this study, we proposed to compute Voronoi diagrams in a distributed fashion without any explicit synchronization. We tackled the "random wind" variations due to the asynchronism and the "wave fusion" phenomenon of the gradient by combining a distance layer and a semantic layer. This enabled to extract complete bisectors and generate simple sites discrete Voronoi diagrams in 1-norm. This work was then extended to Area Voronoi diagrams and simple Line Voronoi diagrams.

Our application was oriented towards robotic navigation. We extracted maximum clearance roadmaps using a network of regularly distributed sensors simulated by the CA. This work could be extended to dynamic routing in distributed networks considering overloaded sites, multi-robot planning, and natural computing simulation if any synchronization heuristic is provided.

Our approach satisfies the criterion of synchronous/asynchronous computing on a static or dynamic map (moving obstacles). The following table compares our method to others depending on qualitative criteria.

	Sync.	Async.	Centr.	Decentr.	Stat.	Dyn.
LVD	Ok	Ok	Ok	Ok	Ok	Ok
Barraquand [20]	Ok	-	Ok	-	Ok	-
Tzionas [10]	Ok	-	-	Ok	Ok	-
Adamatzky [5]	Ok	-	-	Ok	Ok	-

## REFERENCES

- [1] F. Boekhorst, "Ambient intelligence, the next paradigm for consumer electronics: how will it affect silicon?" *2002 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.02CH37315)*, pp. 28–31.
- [2] L. Maignan and F. Gruau, "Convex hulls on cellular spaces: Spatial computing on cellular automata," *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pp. 67–72, Oct.
- [3] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, 2009, vol. 501.
- [4] H. Blum, "A transformation for extracting new descriptors of shape," *Models for the perception of speech and visual form*, vol. 19, no. 5, pp. 362–380, 1967.


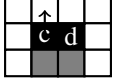
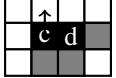
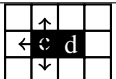
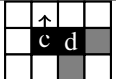
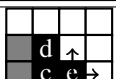
$conflict(c)$ pre-conditions	cases	$solve(c)$ id provided in direction (s)
$c.p \in I$		$c.id_i$
$c.p \in II \wedge d.p \in II$		$uniq(c, d)$
$c.p \in II \wedge d.p \in \{III, IV, V\}$		$d.p$
$c.p \in III \wedge d.p \in III$		$uniq(c, d)$
$c.p \in III \wedge d.p \in \{IV, V\}$		$d.p$
$c.p \in V \wedge d.p \in II \wedge e.p \in III$		$inform(e, d.p)$

Figure 13. Edge Identification Consensus, column 1 gives the pre-conditions for a conflict detection, column 2 are the conflict cases requiring consensus identification, column 3 provides the identifier chosen for the arrow direction(s).

- [5] A. I. Adamatzky, "Voronoi-like partition of lattice in cellular automata," *Mathematical and Computer Modelling*, vol. 23, no. 4, pp. 51–66, 1996.
- [6] W. Maniatty and B. Szymanski, "Fine-grain discrete Voronoi diagram algorithms in L1 and Linf norms," *Mathematical and Computer Modelling*, vol. 26, no. 4, pp. 71 – 78, 1997.
- [7] N. Sudha, S. Nandi, and K. Sridharan, "A parallel algorithm to construct Voronoi diagram and its VLSI architecture," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, 1999, pp. 1683–1688.
- [8] B. de Lacy Costello, I. Jahan, and A. Adamatzky, "Sequential Voronoi diagram calculations using simple chemical reactions," *ArXiv e-prints*, Nov. 2012.
- [9] B. de Lacy Costello, "Calculating Voronoi diagrams using simple chemical reactions," *ArXiv e-prints*, Feb. 2014.
- [10] P. G. Tzionas, A. Thanailakis, and P. G. Tsalides, "Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata," *IEEE Transactions on Robotics*, vol. 13, no. 2, pp. 237–250, 1997.
- [11] A. Adamatzky and B. de Lacy Costello, "Reaction-diffusion path planning in a hybrid chemical and cellular-automaton processor," *Chaos, Solitons & Fractals*, vol. 16, no. 5, pp. 727–736, 2003.
- [12] —, "Collision-free path planning in the Belousov-Zhabotinsky medium assisted by a cellular automaton," *Naturwissenschaften*, vol. 89, no. 10, pp. 474–478, 2002.
- [13] P. G. Tzionas, P. G. Tsalides, and A. Thanailakis, "A new, cellular automaton-based, nearest neighbor pattern classifier and its VLSI implementation," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 2, no. 3, pp. 343–353, 1994.
- [14] A. Adamatzky, B. de Lacy Costello, and N. M. Ratcliffe, "Experimental reactiondiffusion pre-processor for shape recognition," *Physics Letters A*, vol. 297, no. 5, pp. 344–352, 2002.
- [15] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM (JACM)*, vol. 13, no. 4, pp. 471–494, 1966.
- [16] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and image processing*, vol. 14, no. 3, pp. 227–248, 1980.
- [17] M. A. Fischler and P. Barrett, "An iconic transform for sketch completion and shape abstraction," *Computer Graphics and Image Processing*, vol. 13, no. 4, pp. 334–360, 1980.
- [18] C. Arcelli and G. Sanniti di Baja, "Computing Voronoi diagrams in digital pictures," *Pattern Recognition Letters*, vol. 4, no. 5, pp. 383–389, 1986.
- [19] K. E. Hoff, III, J. Keyser, M. Lin, D. Manocha, and T. Culver, "Fast computation of generalized Voronoi diagrams using graphics hardware," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH, 1999, pp. 277–286.
- [20] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 2, pp. 224–241, 1992.
- [21] C. Arcelli, D. Baja, and G. Sanniti, "A width-independent fast thinning algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 4, pp. 463–474, 1985.