



**UNIVERSITÉ
DE LORRAINE**

RAPPORT DE STAGE M2 INFORMATIQUE SECTION RECHERCHE R.A.R (Février-Juin 2012)

Encadrants: Olivier Simonin
Équipe: MAIA - Loria

CALCUL SPATIAL POUR LA NAVIGATION D'UN ROBOT DANS UN ENVIRONNEMENT DYNAMIQUE.

29 juin 2012

Nassim Kaldé
Université Henri Poincaré - Nancy 1
Faculté des Sciences et Technologies - Université de Lorraine
nassim.kalde.1@etumail.uhp-nancy.fr

Table des matières

Remerciements	iii
Résumé / Abstract	iv
1 Introduction	1
Introduction	1
2 Définitions des diagrammes et squelettes de voronoi	3
2.1 Voronoi Basique [20]	3
2.1.1 Définition 1 «Diagramme basique de Voronoi dans le plan»	3
2.1.2 Formalisation 1	3
2.1.3 Définition 2 «Diagramme basique de Voronoi dans le plan» (par intersection de demi-plans)	4
2.1.4 Définitions intermédiaires	4
2.1.5 Formalisation 2	4
2.2 Voronoi Généralisé [20]	5
2.2.1 Définition	5
2.2.2 Formalisation	5
2.3 Voronoi Digital (ou Discret ²) [20]	6
2.3.1 Définition 1 Métrique	6
2.3.2 Formalisation 1 Voronoi Digital (ou Discret)	7
2.4 Squelette de voronoi	7
2.4.1 Définition	8
2.4.2 Formalisation [26]	8
3 Algorithmes de squelettisation existants	9
3.1 Numerical Potential Field Techniques for Robot Path Planning [4]	9
3.1.1 Principe général	10
3.1.2 Algorithmes	10
3.1.3 Limites	13
3.2 Voronoi Like Partition of Lattice in Cellular Automata [2]	14
3.2.1 Principe général	14
3.2.2 Algorithmes	16
3.2.3 Limites	17
3.3 Collision-Free Path Planning for a Diamond-Shaped Robot Using Two-Dimensional Cellular Automata [28]	19
3.3.1 Principe général	19
3.3.2 Algorithmes	21

3.3.3	Limites	22
4	Algorithmes proposés	23
4.1	Introduction	23
4.2	Squelette de Voronoi Digital d'Aire	23
4.2.1	Principe	23
4.2.2	Algorithmes	24
4.2.3	Amélioration	25
4.2.4	Limites	25
4.3	Squelette de Voronoi Digital d'Aire (+Concavité) (Ligne*)	25
4.3.1	Principe	26
4.3.2	Algorithmes	26
4.3.3	Limites	29
5	Simulation	30
5.1	Introduction	30
5.2	Simulateur	30
5.3	Cartes	31
5.3.1	Expérimentation modèle 1 : Area Voronoi Diagram	31
5.3.2	Expérimentation modèle 2 : Area Voronoi Diagram + Concavity	32
5.3.3	Expérimentation modèle 2 : nouvelle règle d'extraction (Dist*Id)	33
5.3.4	Comparatif des différentes méthodes abordées dans ce papier	34
6	Conclusion	36
	Conclusion	36
	Annexe	38
	Références	42

Remerciements

Je vais débuter cette partie en adressant des remerciements à mon encadrant de stage, Olivier Simonin qui a été le plus présent possible lors de ces derniers mois et avec lequel j'ai pu dialoguer régulièrement. En effet il a pris le temps de m'écouter et me conseiller en termes de lectures et de méthodes de travail malgré un emploi du temps surchargé. J'ai aussi été libre dans la plupart des décisions de recherche qui ont guidé mon étude durant ces derniers mois. Sans oublier, François Charpillet pour ses conseils et sa volonté clairement exprimée de faciliter la cohésion au sein de l'équipe, par une intégration rapide des nouveaux arrivants.

Je n'oublierai pas non plus l'ingénieur de recherche Olivier Rochel pour son aide précieuse lors de la prise en main du simulateur SmartTiles, et les nombreux échanges qui ont eu lieu par la suite concernant les retours d'utilisation et l'amélioration de l'outil.

Je finirai par remercier les autres membres permanents et non-permanents de l'équipe, pour leurs conseils et les nombreuses discussions qui ont pu animer nos repas au sein du laboratoire .

Résumé / Abstract

Resume : Dans le cadre de la navigation robotique en environnement dynamique, ce rapport présente une approche pour le calcul *asynchrone* d'un squelette de voronoi sur une grille ordinaire de cellules avec un voisinage de von neumann (maillage d4). Ce modèle est utilisé en robotique mobile pour la planification d'un chemin en environnement *dynamique* équipé de dalles intelligentes. Les méthodes données dans ce document décrivent aussi des algorithmes pour la détection d'obstacle, l'identification et le partage d'identifiant de bordures d'obstacle sur une grille cellulaire. Nous avons mené nos expérimentations sur le simulateur SmartTiles.

Mots-clés : Calcul spatial, diagramme de voronoi discret, squelette discret, évitement dynamique d'obstacle, planification en robotique mobile, calcul asynchrone, réseau de dalles intelligentes, automate cellulaire.

Abstract : This work is related to robotic navigation in a dynamic environment and this report presents a novel approach for *asynchronous* computing of a voronoi skeleton on a regular grid of cells with a von neumann neighborhood. This model is used in mobile robotic for path planification in a *dynamic* environment equipped with smart tiles. The methods given in this paper describe algorithms for obstacle detection, obstacle border identification and sharing id principles on a grid of cells. We have implemented our algorithms in simulation using the SmartTiles simulator.

Key-words : spatial computing, discrete voronoi diagram, discrete skeleton, dynamic obstacle avoidance, robot path planning, asynchronous computing, smart tiles, cellular automata.

1 Introduction

Le calcul spatial [18] repose sur un ensemble d'unités de calcul de structures potentiellement hétérogènes qui sont distribuées dans un environnement. Les données sont partout dans le réseau et les calculs se font selon des modes de synchronisation variables. Des algorithmes caractéristiques déployés sur les nœuds d'un réseau de ce type doivent permettre d'obtenir un comportement global illustrant une solution à un problème posé. Le sujet consiste ici à proposer un algorithme de suivi d'un individu dans un environnement dynamique.

Ce stage s'est déroulé dans le cadre du Master 2 Recherche Informatique R.A.R (Reconnaissance Apprentissage Raisonnement) de la Faculté des Sciences et Technologies Université Henri Poincaré Nancy 1, (Université de Lorraine) au sein de l'équipe MAIA (Autonomous Intelligence Machine) du Loria (Laboratoire IOrrain de Recherche Informatique et ses Applications). Le sujet de ce stage a été proposé par Olivier Simonin et François Charpillet et s'inscrit dans un des axes de recherche de l'équipe, centré sur l'habitat du futur. Un prototype d'habitat du futur est présent au sein des murs du laboratoire et a pris la forme d'un appartement intelligent fortement domotisé. Cet environnement est équipé d'un ensemble de capteurs (au sol, au plafond, sur les murs) afin de recueillir des données actimétriques (relative à l'activité des individus y évoluant). Ceci permettra à terme, de connaître les habitudes de vie d'une personne et proposer de manière intelligente des services d'assistance aux actions du quotidien.

Mon travail s'est surtout porté sur les capteurs présents au sol, en effet un réseau de dalles intelligentes [22] recouvre l'appartement. L'objectif était de mettre au point des algorithmes utilisant uniquement les perceptions du réseau, exploitant l'architecture de calcul fortement distribuée associée de manière asynchrone, et étant capable pour finir de supporter des changements dynamiques. Autrement dit, le réseau formé par les dalles étant assimilable à un automate cellulaire, le travail effectué consistait à proposer un algorithme de planification de trajectoire sur cette grille. Le critère retenu a été celui du chemin le plus dégagé, et une méthode ayant fait ses preuves pour ce type de critère est basée sur le diagramme de voronoi. Nous allons donc exposer les travaux réalisés au cours des derniers mois selon le plan suivant :

Dans un premier temps nous allons poser un ensemble de définitions concernant les diagrammes de voronoi et le squelette associé. Ceci nous permettra de mieux cerner le type de diagramme de voronoi sur lequel nous voulons travailler.

Par la suite nous présenterons certains algorithmes permettant d'extraire le squelette de voronoi dans des modes de calcul et des formalismes différents. Nous allons mettre en avant l'originalité de chaque approche présentée.

Dans la troisième partie de ce rapport nous proposons un modèle de calcul pour le diagramme de voronoi d'aire et le diagramme de voronoi d'aire augmenté, **en mode asynchrone**.

L'avant dernier chapitre présentera le simulateur utilisé pour implémenter les modèles de la section précédente, et certains résultats obtenus sur des cartes.

La dernière partie conclura ce stage, présentera les travaux en cours à l'heure actuelle et proposera de nouvelles perspectives concernant le sujet.

2 Définitions des diagrammes et squelettes de voronoi

Dans cette première partie du rapport nous allons exposer un ensemble de définitions concernant les diagrammes de Voronoi. Nous présenterons des illustrations régulièrement afin d'en saisir rapidement les significations.

2.1 Voronoi Basique [20]

2.1.1 Définition 1 «Diagramme basique de Voronoi dans le plan»

Soit un ensemble fini de points dans le plan euclidien. Les points de cet ensemble sont des sites, et à chaque site est associé l'ensemble de ses points les plus proches dans le plan euclidien selon la distance euclidienne. On obtient donc un pavage du plan euclidien en un ensemble de régions rattachées chacune à un site. Le dallage obtenue est le *diagramme basique de Voronoi dans le plan*, il a été généré à partir de l'ensemble des sites, et chacune des cellules obtenues est appelée *polygone basique de Voronoi*.

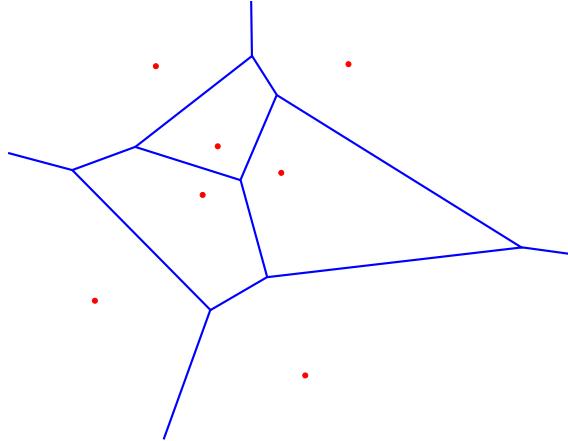


FIGURE 2.1: Diagramme basique de Voronoi dans le plan (Wikipedia)

2.1.2 Formalisation 1

On a :

- $P = \{p_1, \dots, p_n\}$, $n \geq 2$, $P \subset \mathbb{R}^2$, l'ensemble des *sites*.
- $V(p_i) = \{x / \|x - x_i\| \leq \|x - x_j\|, j \neq i, j \in [1, n]\}$ le *polygone basique* de Voronoi associé au point p_i .
- $V = \{V(p_1), \dots, V(p_n)\}$ le *diagramme basique* de Voronoi généré par P .

Chaque point de label p_i représente un vecteur de coordonnées cartésiennes $x_i = (x_{i1}, x_{i2})$ et on a pour tout $i, j \in [1, n]$, $i \neq j$, $x_i \neq x_j$.

2.1.3 Définition 2 «Diagramme basique de Voronoi dans le plan» (par intersection de demi-plans)

Le diagramme de Voronoi est constitué de polygones, un polygone se définit par des demi-plans, on peut donc proposer une définition alternative des polygones basiques de Voronoi par intersection de demi-plans.

Soit un ensemble fini de points dans le plan euclidien. Les points de cet ensemble sont des sites, chaque paire de sites possède une bissectrice qui sépare la plan en deux demi-plans. Chacun des demi-plans possède donc un site distinct. Le polygone de voronoi d'un site peut donc être obtenu par intersection de tous les demi-plans existant entre ce site et les autres sites du plan.

2.1.4 Définitions intermédiaires

Bissectrice de Voronoi : On définit la bissectrice de Voronoi de deux sites s et t comme l'ensemble des points à équidistance des deux sites.

$$B(s, t) = \{x \in E / d(x, s) = d(x, t)\}$$

Demi-espace de Voronoi : On définit le demi-espace de Voronoi d'une cellule s par rapport à une cellule t comme le demi-espace délimité par la bissectrice de s et t , qui contient s .

$$H(s, t) = \{x \in E / d(x, s) \leq d(x, t)\}$$

2.1.5 Formalisation 2

On a

- $P = \{p_1, \dots, p_n\}$, $n \geq 2$, $P \subset \mathbb{R}^2$, l'ensemble des *sites*.
- $V(p_i) = \cap_{j, j \neq i} H(p_i, p_j)$ le *polygone basique* de Voronoi associé au point p_i .
- $V = \{V(p_1), \dots, V(p_n)\}$ le *diagramme basique* de Voronoi généré par P .

Il existe des définitions généralisant le diagramme de Voronoi en dimension supérieure quelconque mais nous ne nous y intéresserons pas ici. Nous venons d'exposer les deux grandes manières de formaliser les diagrammes basiques de voronoi dans le plan et nous

allons continuer sur notre lancée et introduire une nouvelle formalisation : celle des *diagrammes de voronoi généralisés*.

2.2 Voronoi Généralisé [20]

2.2.1 Définition

Dans le domaine de la planification de chemin en robotique mobile, le diagramme de voronoi dans le plan ne représente pas un modèle satisfaisant de génération de chemins sur une carte. En effet nous ne pouvons pas assimiler un obstacle à un site génératrice ponctuel car dans la majorité des cas, un obstacle est une région de la carte. Nous avons donc besoin d'un formalisme plus général.

On peut voir un diagramme de voronoi de manière abstraite comme une association entre chaque point d'un espace avec un site génératrice selon des règles d'association. Une règle d'association met en relation un point de l'espace avec au moins un site génératrice. De manière générale un site peut être une forme quelconque représentée par un sous-ensemble de points de l'espace et plus seulement un point isolé.

2.2.2 Formalisation

On a :

- S un espace quelconque non vide.
- $A = \{A_1, \dots, A_n\}$, $n \geq 2$, $A_i \subset S$ et $A_i \cap A_j = \emptyset, i \neq j$.
- δ_A une règle d'affectation.
- $V(A_i) = \{p \in S / \delta_A(p, A_i) = 1\}$.
- $e(A_i, A_j) = V(A_i) \cap V(A_j), i \neq j$.
- $V(A, \delta_A, S) = \{V(A_1), \dots, V(A_n)\}$.

A est appelé ensemble génératrice et les A_i sont des sites génératrices de S . Étant donnés un espace S et un ensemble A , on définit une **règle** δ_A qui permet d'affecter un point p de S à au moins un élément de A :

$$\delta_A(p, A_i) \leftarrow \begin{cases} 1 & \text{si } p \text{ est assigné à } A_i, \\ 0 & \text{sinon} \end{cases}$$

Et nous avons les 4 conditions suivantes :

1. $\sum_{i=1}^n \delta_A(p, A_i) \geq 1, p \in S$,
2. $|\{V(A_i) / V(A_i) \neq \emptyset, V(A_i) \in V(A, \delta_A, S)\}| \geq 2$,
3. condition de région non vide et de mesure positive pour les A_i

4. $e(A_i, A_j) = bV(A_i) \cap bV(A_j)$ pour $bV(A_i) \cap bV(A_j) \neq \emptyset, i \neq j$ ¹

Ces 4 conditions permettent de s'assurer que chaque point de l'espace est associé à au moins un site générateur, qu'il existe au moins deux régions de voronoi, que l'aire de chaque région est positive et que l'arête entre deux régions de voronoi est obtenue par intersection des frontières de ces deux régions.

Ainsi, selon la règle d'affectation δ et la vérifications des quatres conditions énoncées ci-dessus, les ensembles $V(A, \delta_A, S)$ sont exhaustifs par union et mutuellement exclusifs à l'exception des frontières, $V(A, \delta_A, S)$ forme donc un pavage de S . Ce pavage est appelé *diagramme de Voronoi généralisé*.

A partir de la formalisation du diagramme de voronoi généralisé, nous allons définir le diagramme de voronoi digital (ou discret) à métrique discrète.

2.3 Voronoi Digital (ou Discret²) [20]

Le diagramme de voronoi digital permet de représenter un pavage de voronoi dans un plan discret, ici une grille de cellules carrées.

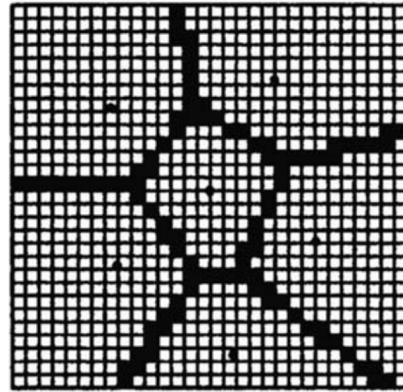


FIGURE 2.2: Diagramme de voronoi digital pour des cellules de diffusion isolées [20]

2.3.1 Définition 1 Métrique

Une métrique sur un ensemble F se définit comme une fonction de $F \times F \rightarrow \mathbb{R}^+$ et elle vérifie les 3 propriétés suivantes :

1. $d(x, x) = 0$

1. b boundary
2. appellation par Watanabe, Schwarzkopf, Melter...

2. $d(x, y) = d(y, x)$
3. $d(x, z) \leq d(x, y) + d(y, z)$

Nous allons ici limiter notre étude à des distances discrètes générant des valeurs entières. Une métrique discrète est donc une fonction de $F^2 \rightarrow \mathbb{N}^+$, respectant les 3 propriétés ci-dessus.

Il existe de nombreuses distances discrètes communes, comme les distances de manhattan, d_8 , d_{inf} , ou de chanfrein par exemple [27].

2.3.2 Formalisation 1 Voronoi Digital (ou Discret)

On a :

- C une matrice de cellules de dimension $l \times c$, ($l, c \in \mathbb{N}$)
- C_{ij} une cellule de cette matrice ($0 < i \leq l, 0 < j \leq c$)
- $P_c = \{P_{c1}, \dots, P_{cn}\}$, $cn \geq 2$, $P_{ci} \subset C$
- $d(a, b)$ une métrique sur cet espace
- $V(P_{ci}) = \{c_{km}/d(c_{km}, P_{ci}) \leq d(c_{km}, P_{cj}) \text{ pour } j \neq i, k \in [1, l], m \in [1, c]\}$

Le diagramme de Voronoi digital est donc l'ensemble $V(P_c) = \{V(P_{c1}), \dots, V(P_{cn})\}$.

Les règles d'**affectation** basées sur un critère algébrique sont répandues dans le domaine du calcul de diagramme de voronoi, on peut donc noter l'utilisation ci-dessus, d'une métrique utilisée comme critère de décision.

2.4 Squelette de voronoi

La dernière notion à introduire dans cette partie est celle de squelette de voronoi. Cette notion apparaît dès les années 60 dans des travaux de H. Blum [6] pour l'analyse de formes. L'intérêt de la squelettisation d'objet est de fournir un descripteur de l'objet en question. Comme la plupart des descripteurs, il présente des propriétés d'invariance à certaines transformations et réduit la quantité d'information nécessaire à la description de l'objet en fournissant une signature de ce dernier. Le squelette généré peut permettre dans certains cas de reconstruire l'objet de base. Nous utiliserons la squelettisation dans une optique différente, afin de définir un chemin équidistant des sites (obstacles). Un exemple d'application est présenté au chapitre 5 de ce rapport.

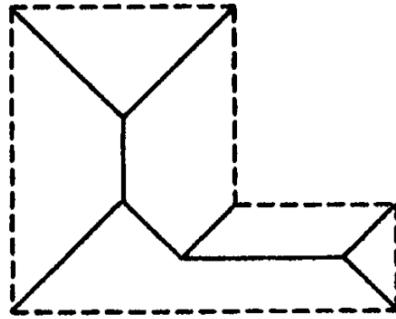


FIGURE 2.3: Exemple de squelette [20]

2.4.1 Définition

Nous allons présenter une première définition informelle de la notion de squelette en se basant sur l'image du feu de forêt initié sur les bordures. Le squelette représentera au final l'ensemble des points où les différents fronts de flamme vont se rencontrer. Dans le cas d'un objet, le squelette représente donc les points à équidistance du contour. En voici une définition plus rigoureuse :

2.4.2 Formalisation [26]

On a :

- F une figure
- C son contour
- p un point intérieur de cette figure
- v la vitesse de propagation de l'onde depuis C
- $t(p)$ l'instant où l'onde atteint p
- $d(p) = v \cdot t(p)$ la distance de P au contour C
- $Squelette(C) = \{p / |\{q \in C / d(p, q) = d(p)\}| \geq 2\}$

Les notions de diagramme de voronoi et squelette de voronoi se rejoignent, on comprend donc que le squelette est inclus dans les arrêtes du diagramme de voronoi.

Nous arrivons au terme de la première partie de ce rapport, nous avons présenter des notions essentielles à la compréhension de la suite de notre exposé. Nous focaliserons notre étude sur le diagramme de voronoi digital à métrique discrète et le squelette associé. Nous apporterons des précisions complémentaires dès que cela sera nécessaire.

3 Algorithmes de squelettisation existants

Dans cette partie nous allons présenter certains algorithmes existants pour le calcul du diagramme de voronoi digital ou du squelette associé. Il existe 4 grandes approches pour les algorithmes de calcul du squelette de voronoi digital : celles par simulation de feu de forêt [16], par diagramme de Voronoi [19], par transformée de distance [8], ou encore par amincissement de formes [25] que l'on rencontre surtout dans le domaine de l'imagerie.

3.1 Numerical Potential Field Techniques for Robot Path Planning [4]

Le papier dont est issu l'algorithme que nous allons présenter, traite de navigation en robotique mobile. Ce document présente différentes méthodes pour établir un champ de potentiel sur une carte afin de permettre à un robot de rallier un but en se frayant un chemin parmi les obstacles. Établir un champ de potentiel dans un plan revient à créer un relief sur la carte afin de guider le robot vers un minimum global représentant le but. Ainsi le robot n'a plus qu'à suivre le gradient de l'information de relief produite.

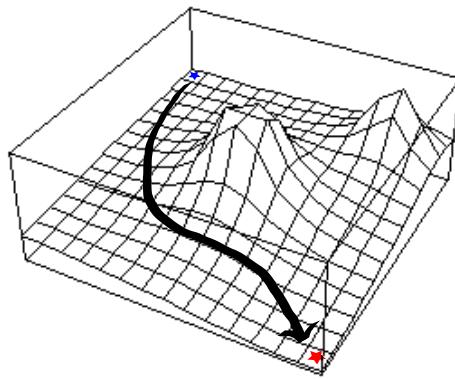


FIGURE 3.1: Principe navigation par champ de potentiel [13]

3.1.1 Principe général

L'une des méthodes présentées (Improved W-potential) détourne l'utilisation basique de l'algorithme d'établissement du champ de potentiel (Algorithm 1 wavefront expansion) afin de générer le squelette de voronoi d'une carte. Dans un premier temps, le calcul du squelette se fait par propagation d'un champ de potentiel depuis les obstacles. Une fois le squelette de navigation obtenu, le but est rattaché à la route de navigation la plus proche appartenant au squelette. Un champ de potentiel est propagé depuis le but dans le squelette, puis dans le reste de la carte.

Le relief obtenu permet donc d'atteindre le but en 3 temps :

1. rallier le squelette
2. naviguer sur le squelette
3. rejoindre le but

3.1.2 Algorithmes

Les algorithmes présentés prennent en entrée des cartes bitmap à résolution variable, le calcul y est centralisé.

Voici une liste des termes apparaissant dans la suite :

- W représente le monde,
- GW représente le monde augmenté d'une grille,
- GW_{empty} représente l'ensemble de la grille non occupée par des obstacles,
- $BM : W \rightarrow \{0, 1\}$ associe à un point du monde, sa valeur bitmap 0 (vide), 1 (occupé),
- $d1$ représente la distance discrète L^1 (*Manhattan*),
- *neighbors* les voisins Nord, Est, Sud, Ouest d'un point.

Algorithme centralisé de la vague (wavefront expansion) pour le calcul des distances des points de GW_{empty} aux obstacles :

Algorithm 1 Pseudo-code Wavefront expansion computing the d_1 map

```

1: {1 Initialization}
2: for all  $x \in GW_{empty}$  do
3:    $x.d_1 \leftarrow inf$ 
4: end for
5: {2 Frame Boundaries and Obstacle Borders Identification}
6:  $L_0 \leftarrow list()$ 
7: for all ( $x \in GW$  such that  $BM(x) = 1$ ) and ( $\exists n \in x.neighbors/n \in GW_{empty}$ ) do
8:    $x.d1 \leftarrow 0$ ,  $L_0 \leftarrow L_0 \cup x$ 
9: end for
10: {3 Propagation}
11:  $L_i \leftarrow L_0$ 
12: for all  $i = 0, 1, 2$  until  $L_i = \emptyset$  do
13:    $L_{i+1} \leftarrow list()$ 
14:   for all  $x \in L_i$  do
15:     for all ( $y \in x.neighbors$  such that  $y.d1 = inf$ ) do
16:        $y.d1 \leftarrow i + 1$ ,  $L_{i+1} \leftarrow L_{i+1} \cup y$ 
17:     end for
18:   end for
19: end for
```

Lors du calcul des distances d_1 , chaque point mémorise aussi la source de la propagation qui l'a atteint. La dernière phase est la phase d'extraction des points du squelette et voici l'algorithme d'extraction du squelette à partir de la carte des distances :

Algorithm 2 Pseudo-code Algorithm Skeleton Extraction

```

1:  $\{(skeleton)S \leftarrow \emptyset, threshold \leftarrow 2\}$ 
2:  $\{1\ Sources\ Initialization\}$ 
3: for all  $x \in L_0$  do
4:    $x.O \leftarrow x$ 
5: end for
6:  $\{2\ Sources\ Propagation\ and\ Skeleton\ Point\ Selection\}$ 
7: for all  $L_i \in \{L_0, \dots\}$  do
8:   for all  $x \in L_i$  do
9:     for all  $y \in x.neighbors$  do
10:      if  $y.d_1 = inf$  then
11:         $y.O \leftarrow x.O$ 
12:      else if  $x \notin S$  and  $|d(y.O - x.O)| \geq threshold$  then
13:         $S \leftarrow S \cup y$ 
14:      end if
15:    end for
16:  end for
17: end for
```

Voici quelques résultats d'exécution obtenus pour cet algorithme :

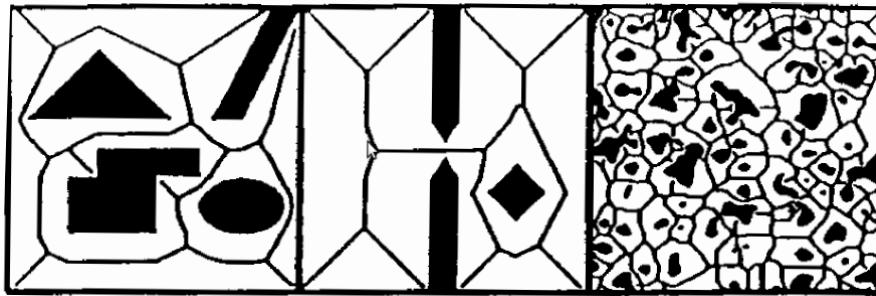


FIGURE 3.2: Exemples d'extraction de squelette [4]

Le pseudo-code de l'algorithme complet d'établissement du champ de potentiel exploitant le squelette *Improved W-potential* est disponible en (*algorithm 3*), pour ne pas surcharger la lecture nous rappelons juste les grandes étapes de cet algorithme pour le calcul du champ de potentiel exploitant le squelette de voronoi :

1. Initialisation du champ de potentiel, le relief est infini partout.
2. Connection du but au squelette S, on remonte le gradient d_1 du but jusqu'à atteindre le squelette.
3. Calcul du champ de potentiel dans le squelette, un relief croissant est propagé depuis le but dans le squelette.
4. Diffusion du champ de potentiel dans le reste de la carte, un relief croissant est propagé depuis le squelette dans le reste de la carte.

Voici un exemple de résultat obtenu :



FIGURE 3.3: Contours Improved W-potential [4]

3.1.3 Limites

La méthode présentée fonctionne de manière centralisée, il faut donc pouvoir l'adapter au calcul décentralisé dans un premier temps. Mais nous retenons tout de même le système de navigation présenté qui est basé sur le squelette de voronoi et les champs de potentiel.

3.2 Voronoi Like Partition of Lattice in Cellular Automata [2]

Nous quittons le domaine de la navigation robotique en abordant ce nouvel article. Nous traitons toujours de calcul de diagramme de voronoi digital, mais cette fois-ci la structure de calcul est celle d'un automate cellulaire. La méthode présentée montre une procédure en milieu discret, avec un mode de calcul décentralisé parallèle **synchrone**.

3.2.1 Principe général

Le principe général reste celui de la propagation d'onde, et de la détection des points de rencontre des fronts pour établir le squelette. Nous allons au préalable rappeler le formalisme de l'automate cellulaire.

Rappel : Un Automate Cellulaire 2D est un tuple $\mathbb{F} = \langle A, Q, u, f \rangle$:

- A : matrice de cellules de dimension $m \times m$
- Q : ensemble des états d'une cellule, $|Q| = q$
- $u : A \rightarrow A^k$ voisinage cellulaire
- $f : Q^k \rightarrow Q$ fonction d'évolution locale

Le voisinage u_1 ou de *von-neumann* est un mapping qui associe à toute cellule $c_{i,j}$ de A , $c_{i,j}$ elle-même ainsi que ses voisines Nord, Est, Sud et Ouest.

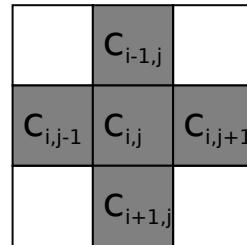


FIGURE 3.4: Voisinage Von Neumann

L'automate évolue en temps discret, chaque cellule calcule donc son état à $t + 1$ en appliquant la fonction d'évolution locale sur son voisinage courant au temps t .

$$c^{t+1} = f(u_1(c)^t)$$

Nous en avons terminé avec ce rappel de formalisme pour automate cellulaire, nous allons à présent nous intéresser aux limites de la définition de bissectrice dans le milieu cellulaire.

Bissectrice dans le plan

Le papier met tout d'abord en avant les limites de la définition de la “bissectrice dans le plan” dès que l'on passe sur une grille cellulaire, avec une distance de manhattan d_1 ($d_1(a, b) = |x_a - x_b| + |y_a - y_b|$).

$$B1(a, b) = \{c \in A : d_1(a, c) = d_1(b, c)\}$$

Caractéristique : $B1$ ne permet pas d'obtenir la bissectrice lorsque $\text{parité}(|y_a - y_b|) \neq \text{parité}(|x_a - x_b|)$.

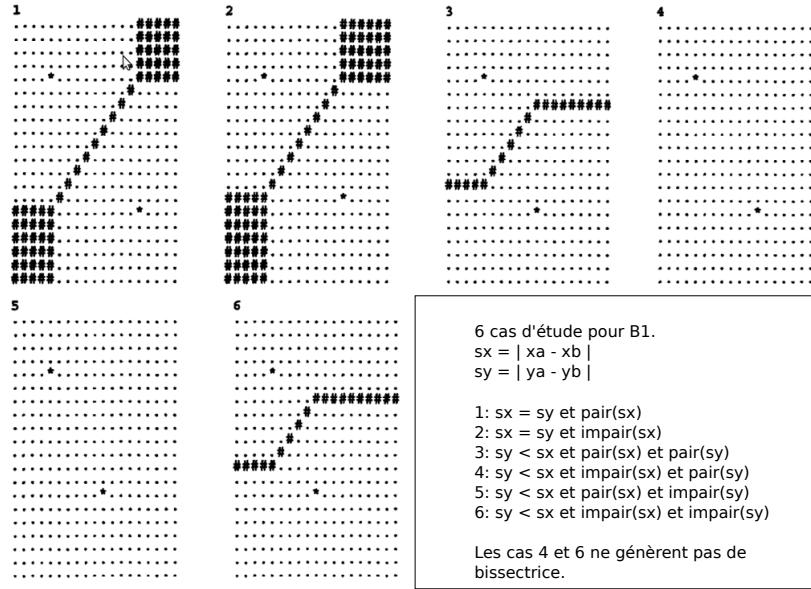


FIGURE 3.5: Bissectrice $B1$ [2]

Bissectrice sur une grille

L'auteur redéfinit la bissectrice de la manière suivante, et un résultat satisfaisant est finalement obtenu car toutes les bissectrices apparaissent. Les bissectrices peuvent prendre la forme de lignes épaisses.

$$B2(a, b) = \{c \in A : |d_1(a, c) - d_1(b, c)| \geq 1\}$$

Caractéristique : Pour tout $a, b \in A$, $B_2(a, b) \neq \emptyset$.

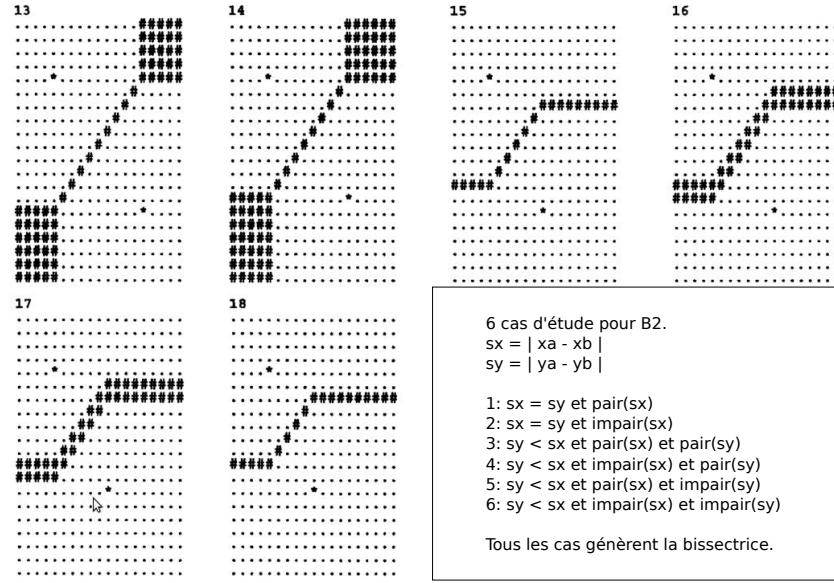


FIGURE 3.6: Bissectrice B2 [2]

Nous allons dans la section suivante présenter l'algorithme utilisé pour le calcul du squelette.

3.2.2 Algorithmes

Nous allons présenter un des deux algorithmes proposés dans ce papier. L'espace d'états de chaque cellule est $Q = \{1, 2, \dots, n, -, \bullet, \#\}$, les numéros représentent les états excités, \bullet est l'état de repos, $-$ l'état réfractaire, et $\#$ l'état de bissectrice d'une cellule.

Le principe de l'algorithme est simple, les sites de diffusion sont identifiés par un numéro (état excité) et ce numéro est transmis lors de la propagation qui va suivre. Chaque cellule dans l'état excité réveille ses cellules voisines avant de passer dans l'état réfractaire. Les voisines vont à leur tour réveiller leurs voisines non réfractaires et ainsi de suite. Les cellules réfractaires retournent à l'état de repos à l'étape suivante.

Le squelette est identifié par l'ensemble des cellules à l'état $\#$ (frontière entre des cellules d'excitations différentes). Nous présentons la méthode sous deux formes proposées dans l'article.

Règle d'évolution locale :

$$x^{t+1} \leftarrow \begin{cases} \#, & (x^t \in \{1, \dots, n, \bullet\} \wedge |I(x)^t| \geq 2) \vee (x^t = \#), \\ +, & (x^t = \bullet) \wedge I(x)^t = \{+\}, \\ -, & (x^t = +) \wedge (I(x)^t = \{+\} \vee I(x)^t = \{\emptyset\}), \\ \bullet, & (x^t = -) \vee (I(x)^t = \{\emptyset\} \wedge (x^t = \bullet)). \end{cases}$$

Avec $+ \in \{1, \dots, n\}$ et $I(x) = \{h \in Q / \exists y \in u_1(x), y^t = h, h \in (N)\}$

Table de transition locale :

Transition	Condition
$+ \rightarrow -$	$ I(x)^t = 0 \vee I(x)^t = \{+\}$
$\bullet \rightarrow \bullet$	$I(x)^t = \{\emptyset\}$
$\bullet \rightarrow +$	$I(x)^t = \{+\}$
$\bullet \rightarrow \#$	$ I(x)^t \geq 2$
$+ \rightarrow \#$	$ I(x)^t \geq 2$

Voici un exemple de résultat obtenu en utilisant cette méthode :

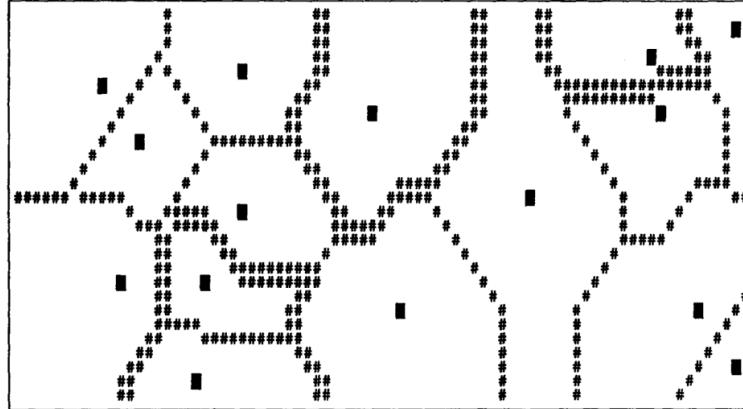


FIGURE 3.7: Exemple d'extraction de squelette [2]

3.2.3 Limites

Tout d'abord, ce modèle ne prend pas en compte des obstacles non ponctuels, ceci est problématique dans le cadre d'une application de navigation en robotique mobile.

Il ne passe pas non plus en mode asynchrone car la temporalité des états est centrale pour l'établissement des règles d'évolution. Un modèle asynchrone se doit donc de simuler cette temporalité avec une information figée dans le système.

Nous nous intéressons aussi à la connectivité des cellules du squelette extrait car nous désirons faire circuler de l'information au sein du squelette. Ici les cellules du squelette

ne sont pas des voisines directes les unes avec les autres, il faudra donc assurer cette connectivité en dilatant le squelette.

3.3 Collision-Free Path Planning for a Diamond-Shaped Robot Using Two-Dimensional Cellular Automata [28]

Pour clore cette partie sur les algorithmes de squelettisation existants, nous allons revenir dans le domaine de la robotique mobile. Les algorithmes sont proposés dans le même formalisme pour automates cellulaires que celui présenté en 2.2.

Tout d'abord, le document présente un algorithme de détection de bordures et d'orientation d'obstacles basé sur des filtres de motifs.

Par ailleurs, le système de diffusion utilisé diffère des précédents par l'utilisation d'un système de drapeaux qui fige les cellules sur la grille, nous le présenterons aussi.

3.3.1 Principe général

Le papier étudié présente une méthode de navigation pour un robot en forme de *diamond* sur une carte statique. La forme de diamant vient du fait que la métrique d_1 utilisée en diffusion synchrone et dans un voisinage de von-neumann produit des formes de losange (*Fig 3.8*).

Nous allons commencer par exposer le système de diffusion d'une valeur V depuis une cellule isolée, il permettra de mieux aborder l'expansion multi-cellules présentée dans la section suivante. L'utilisation du drapeau permet de n'activer une cellule qu'une seule fois.

1. on sélectionne une cellule source avec la valeur V, son drapeau est levé,
2. on répète 3 et 4 sur la grille pendant un certain temps (période de test)
3. chaque cellule vérifie la présence d'un drapeau dans son voisinage de von-neumann
4. si un drapeau est levé, la cellule récupère V et lève son drapeau

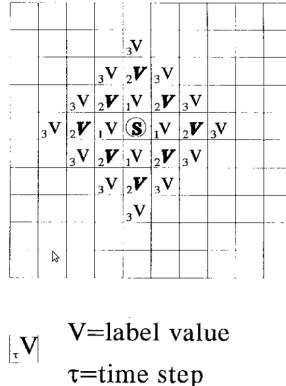


FIGURE 3.8: Évolution de l'automate cellulaire avec une seule source

Algorithm 3 Pseudo-code for the expansion from a single cell

```

1: {1}  $c_{k,l}.V \leftarrow val$ ,  $c_{k,l}.flag \leftarrow 1$ 
2: {2}
3: repeat
4:   for all  $c_{i,j} \in A / c_{i,j}.Flag = 0$  do
5:     {3}
6:     if  $\exists c_i \in c_{i,j}.neighbors / c_i.Flag = 1$  then
7:       {4}  $c_{i,j}.V \leftarrow val$ ,  $c_{i,j}.flag \leftarrow 1$ 
8:     end if
9:   end for
10:  until Testperiod ends

```

Le calcul du squelette de voronoi sur une carte est réalisé en 3 étapes :

1. détection et identification des types de cellules par filtration sur motifs (*Fig 3.9 b*)
2. diffusion des identifiants des cellules en bordure d'obstacles sur la grille (*Fig 3.9 c*)
3. extraction du squelette dans les zones de mixage d'identifiants (*Fig 3.9 d*)

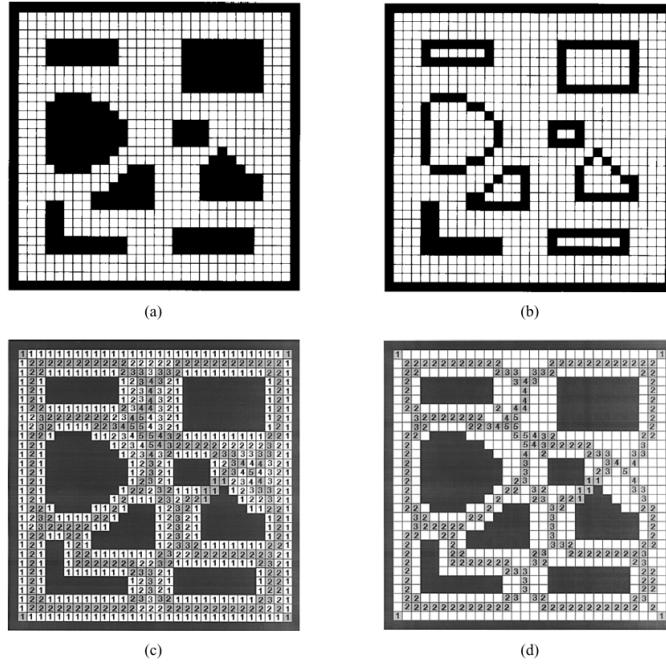


FIGURE 3.9: a) carte initiale b) filtration des bordures c) diffusion d) extraction du squelette [28]

Ces algorithmes sont présentés dans la section suivante.

3.3.2 Algorithmes

La première étape consiste à détecter les types de cellules de l'automate, des filtres sont appliqués sur chaque cellule pour leur fournir un identifiant. Nous allons commencer par présenter le système des filtres de motifs dont le principe d'application est inspiré des détecteurs d'orientation en imagerie numérique.

Un motif est représenté ci-dessous, seul le voisinage de von-neumann est considéré :

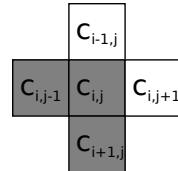


FIGURE 3.10: Exemple de motif $t_c = (0, 1, 1, 0, 1)$

La totalité des motifs **proposés dans l'article** est présentée en annexe (*Fig. 6.1*), ils permettent donc d'identifier 12 configurations de cellules (bordures d'obstacles, cellules libres, bruit). Les filtres sont appliqués sur chaque cellule de l'automate, et lorsqu'un masque correspond, le code associé au masque devient l'identifiant de la cellule. Voici l'algorithme d'identification des cellules :

Algorithm 4 Pseudo-code for object boundary detection

```

1: {templates  $t = \{t_a, \dots, t_l\}$ }
2: for all  $c_{i,j} \in A$  do
3:    $t_c = (c_{i-1,j}, c_{i+1,j}, c_{i,j-1}, c_{i,j+1}, c_{i,j})$ 
4:   if ( $\exists t_i \in t / t_i = t_c$ ) then
5:      $c_{i,j}.edge\_code \leftarrow i$ 
6:   end if
7: end for

```

Une fois le type des cellules connu, nous entrons dans la phase de diffusion des identifiants de bordure d'obstacles. La diffusion est synchrone parallèle, l'identifiant de bordure est transmis de proche en proche. Les cellules libres voisines directes d'une cellule *obstacle* ou ayant levé son drapeau reçoivent l'identifiant de diffusion. Elles lèvent leur drapeau à leur tour et diffusent l'identifiant dans leur voisinage.

Algorithm 5 Pseudo-code for the construction of the Voronoi Diagram

```

{1 Procedure for loading the edge_codes from the object boundary detection}
for all  $c_{i,j} \in A$  {load edge_code as the cell value and set its flag} do
    if ( $c_{i,j}.edge\_code > 4$ ) {values < 5 homogeneous areas or noise} then
         $c_{i,j}.V \leftarrow c_{i,j}.edge\_code$ ,  $c_{i,j}.flag \leftarrow 1$ 
    end if
end for
{2 Procedure for expansion around multiple source points.}
repeat
    for all  $c_{i,j} \in A$  do
         $F = (ni, si, wi, ei) \leftarrow (c_{i-1,j}.flag, c_{i+1,j}.flag, c_{i,j-1}.flag, c_{i,j+1}.flag)$ 
        if  $|Y| = 1$  { $Y = \{y \in c_{i,j}.neighbors / y.flag = 1\}$ } then
            {if only one flag in the neighbourhood then expansion in free space,
            current cell receives the value of the neighbour whose flag is set.}
             $c_n \leftarrow Y.pophead()$ 
             $c_{i,j}.V \leftarrow c_n.V$ 
        else if ( $|Y| > 1$ ) and ( $\exists c_{n1}, c_{n2} \in Y / c_{n1}.V \neq c_{n2}.V$ ) then
            {if more than one flag in the neighbourhood, and the edge_code values
            of these neighbors are different, the time step value is stored as the
            Voronoi label Vor}
             $c_{i,j}.Vor \leftarrow t$ 
        end if
    end for
until Testperiod ends

```

Au final, les zones de mixage, **c'est-à-dire l'ensemble des cellules qui reçoivent des identifiants de diffusion différents**, sont sélectionnées dans le squelette (*Fig 3.9 d*).

Le principe de navigation se base sur la connaissance du rayon du robot, ainsi sur l'ensemble des routes du squelette, seules celles empruntables par le robot sont sélectionnées. De plus le rayon du robot détermine la distance de déviation à une route sélectionnée. Un système de seuillage basique suffit à sélectionner les routes navigables.

3.3.3 Limites

Avec cette approche, il est impossible de détecter des lignes, les obstacles filiformes sont donc ignorés et considérés comme du bruit sur la carte, ainsi le système des motifs présenté est **incomplet**. L'algorithme présenté dans le papier ne permet pas d'obtenir les résultats d'exécution présentés dans le même document, il est soit incomplet lui aussi, soit faux. L'utilisation du système des drapeaux fige l'état de la grille après un seul calcul de squelette ainsi le passage au dynamique est impossible.

4 Algorithmes proposés

4.1 Introduction

À notre connaissance, il n'existe pas de méthode de calcul du squelette de voronoi en mode de calcul parallèle **asynchrone** sur une grille cellulaire. Le mode asynchrone choisi ici est un asynchronisme élémentaire, une seule cellule est choisie aléatoirement sur la grille à chaque instant et sa règle d'évolution est activée. Notre objectif dans cette section est donc de présenter l'approche choisie pour réaliser ce calcul.

Dans un premier temps nous allons présenter une méthode permettant d'obtenir un diagramme de voronoi digital d'aire en métrique d_1 . Puis nous proposerons une première amélioration de ce travail.

Dans un second temps nous présenterons une approche inspirée de 2.3 afin de pallier les limites de la première version.

Pour finir nous montrerons aussi les premières limites de ce modèle.

4.2 Squelette de Voronoi Digital d'Aire

Le diagramme de Voronoi Digital d'Aire est en fait le diagramme de Voronoi Généralisé présenté en chapitre 2 de ce rapport, une section y est consacrée dans [20].

4.2.1 Principe

Nous posons 4 étapes essentielles dans la réalisation de ce premier travail :

1. détection des sites de diffusion
2. consensus pour l'identification des sites de diffusion voisins
3. calcul de la distance aux sites de diffusion sur la grille
4. définition d'une règle d'extraction du squelette de voronoi basée sur un critère de frontière

Bien entendu, la séquentialité des étapes présentées ci-dessus existe au niveau local pour chaque cellule mais n'a pas de réelle valeur en tant qu'observateur extérieur. En effet, chacune de ces étapes exécutées localement doit permettre d'obtenir un résultat global de manière asynchrone.

La fonction de diffusion de la distance dans le réseau est réalisée par une règle simple. Chaque cellule récupère la valeur minimale de son voisinage et y ajoute 1. Cette méthode permet d'obtenir la carte des distances d_1 aux obstacles.

Nous avons adopté cette approche pour qu'elle génère le squelette même en mode asynchrone. En effet, le calcul de distances des cellules par cette méthode, converge quel que soit l'ordre d'activation des cellules.

4.2.2 Algorithmes

Les cellules présentes sur notre grille possèdent chacune une information d'état, d'identifiant et de distance. Nous allons montrer comment ces valeurs sont définies et mises à jour par des règles locales d'évolution.

La détection des sites de diffusion nécessite la prise en compte d'une perception de chaque cellule. Ainsi une cellule possédant un site de diffusion, devient une cellule source :

$$Detection(c) : c.state \leftarrow \begin{cases} 1 & \text{si } \exists \text{site de diffusion} \in c \\ 0 & \text{sinon.} \end{cases}$$

La procédure de consensus pour le choix d'un identifiant commun d'aire de diffusion entre cellules sources voisines, peut s'exprimer de la manière suivante :

$$Consensus(c, d) : c.id \leftarrow \begin{cases} c.id \cup d.id & \text{si } c.state + d.state = 2 \\ c.id & \text{sinon.} \end{cases}$$

Le calcul des distances d_1 aux cellules sources et la propagation des identifiants de diffusion sur l'automate cellulaire se fait selon la règle suivante :

$$D1(c) : (c.d1, c.id) \leftarrow \begin{cases} (n.d1 + 1, n.id) \text{ pour } n = argmin_{n_i \in c.neighbors}(n.i.d1) & \text{si } c.state = 0 \\ (0, c.id) & \text{sinon.} \end{cases}$$

On sélectionne ensuite les cellules aux frontières des différentes zones de diffusion. La règle d'extraction du squelette basé sur les identifiants peut s'écrire sous la forme suivante :

$$Extraction(c) : c.skeleton \leftarrow \begin{cases} 1 & \text{si } (\exists n \in c.neighbors / n.id \neq c.id) \\ 0 & \text{sinon.} \end{cases}$$

Cette première approche permet d'obtenir un diagramme d'aire de voronoi dans un milieu cellulaire. Il faut comprendre que le squelette n'est bon qu'à partir du moment que la carte des distances n'évolue plus dans le réseau. Il existe d'autres règles permettant d'obtenir les mêmes résultats, l'intérêt ici est juste de présenter le modèle de calcul.

La procédure locale complète exécutée à chaque sélection d'une cellule c est donc la suivante :

Algorithm 6 Pseudo-code area voronoi diagram skeleton extraction

```

{1 Detection}
Detection(c)
{2 Identifier Consensus}
for all  $d \in c.neighbors$  do
    Consensus(c,d)
end for
{3 Distance Computation}
D1(c)
{4 Skeleton Extraction}
Extraction(c)

```

4.2.3 Amélioration

Une première amélioration permet de limiter le temps de consensus en faisant intervenir moins de sites dans cette étape de prise de décision. En effet il suffit de ne faire intervenir que les bordures d'obstacles et leur jonction.

$$Consensus2(c, d) : c.id \leftarrow \begin{cases} c.id \cup d.id & si (c.state + d.state = 2) and \\ & (\exists n \in d.neighbors \cup c.neighbors / n.state = 0), \\ c.id & sinon. \end{cases}$$

En suivant la même méthode, on comprend que de nombreuses améliorations restent possibles. Il suffit de considérer chacune des étapes séparément et de tenter d'optimiser les critères de décision présentés. Nous n'allons pas essayer de pousser ce modèle plus loin.

4.2.4 Limites

Dans le cadre de la navigation en robotique mobile, nous identifions une limite importante de ce modèle basé sur le diagramme de voronoi d'aire : la navigation dans un labyrinthe est impossible, les frontières internes d'une zone de diffusion sont non définies donc la concavité des obstacles n'est pas prise en compte

Nous allons proposer une nouvelle méthode d'extraction du squelette afin de pallier à cette première limite.

4.3 Squelette de Voronoi Digital d'Aire (+Concavité) (Ligne*)

Le modèle précédent ne prend en compte que les frontières inter-zones de diffusion. L'intersection des vagues émises depuis les faces consécutives d'une même aire de diffusion

ne sont pas extraites. Nous allons donc augmenter le modèle précédent afin d'observer des bissectrices intra-zone.

Un site de diffusion peut être assimilé à un polygone sur l'espace cellulaire. Ceci va nous permettre de proposer une méthode d'extraction des segments du polygone. Chacun de ces segments sera donc considéré comme une portion de diffusion indépendante possédant un identifiant propre.

Nous essayons d'obtenir des résultats similaires à la définition des diagrammes de voronoi de ligne donnée dans [20].

4.3.1 Principe

Différentes modifications du modèle précédent sont alors possibles afin de prendre en compte les frontières intra-zone de diffusion :

1. modification de la fonction de consensus (points de rupture)
2. modification de la règle d'extraction (critère des distances)

Dans un premier temps, nous avons choisi de réintroduire le système de filtres présenté en 2.3, il sera complété et nous définirons des sous-classes de motifs.

Puis nous allons modifier la fonction de consensus en explicitant les interactions possibles entre les différentes classes de motifs. Ceci nous permettra d'identifier les faces des polygones de diffusion afin de créer des sous-zones de diffusion à identifiant commun.

La suite de cette méthode reprend les étapes de calcul de distance et d'extraction de squelette du modèle précédent.

4.3.2 Algorithmes

Voici M l'ensemble des filtres des sites de diffusion en voisinage de von-neumann. Un motif de diffusion m_i est un quadruplet $(0, 1)^4$ et il en existe 16 que nous répartissons en 6 classes de symétries.

Classes	Motifs			
I isolé	1			
II coin	7	8	9	11
III ligne	6	10		
IV extrémité	2	3	4	5
V direction	12	13	14	15
VI plein	16			

FIGURE 4.1: Classes et motifs locaux de diffusion

Une nouvelle fonction de perception utilisant ces motifs permet d'identifier le type d'une cellule c :

$$Detection2(c) : c.pattern \leftarrow \begin{cases} i & \{\exists m_i \in M / (s_n, s_e, s_s, s_o) = m_i\} \quad si \ c.state = 1, \\ 0 & sinon. \end{cases}$$

La fonction de consensus d'identification s'exprime à présent en prenant en compte les différentes classes de sites de diffusion. Ainsi l'interaction entre classes permettra d'identifier des faces des polygones de diffusion¹

$$Consensus(c, d) : c.id \leftarrow \begin{cases} ConsensusI(c, d) & si \ c \in I \\ ConsensusII(c, d) & si \ c \in II \\ ConsensusIII(c, d) & si \ c \in III \\ ConsensusIV(c, d) & si \ c \in IV \\ ConsensusV(c, d) & si \ c \in V \\ ConsensusVI(c, d) & si \ c \in VI \\ c.id & sinon. \end{cases}$$

Pour que le lecteur puisse se représenter le principe d'identification des faces nous allons proposer un exemple simple. On va commencer par présenter le modèle de diffusion d'identifiant des cellules de classe V et on montrera un exemple de résultat obtenu grâce à cette règle.

1. Le détail des fonctions de consensus est fourni en annexe car nous ne voulons pas surcharger cette partie du rapport.

Pour finir, on donnera la règle ConsensusV qui a permis d'obtenir ce résultat.

La figure 4.2 (a) représente une cellule c de classe 5, possédant un identifiant mis à disposition pour toutes ses voisines directes. Toute cellule du voisinage obtient donc la même information d'identifiant en provenance de c . Ce n'est pas toujours le cas, en effet certaines classes d'identifiant possèdent jusqu'à 4 identifiants distincts.

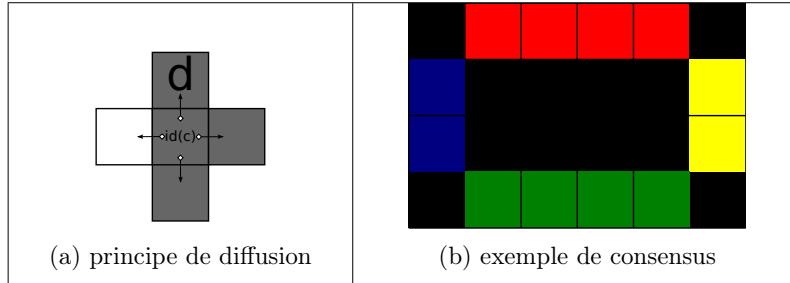


FIGURE 4.2: Classe V identifiant et exemple de consensus

La figure 4.2 (b) représente une aire de diffusion cellulaire rectangulaire de taille $4*6$. Les cellules de classeV sont colorées et sur chaque face du rectangle, une couleur unique apparaît. Cette couleur représente le résultat obtenu après consensus, les quatres faces ont bien été identifiées.

La fonction de consensus associée à cette classe est très simple, en effet, seules les cellules de ClasseV et de motif similaire participent à la procédure de consensus. Les autres sont simplement ignorées et n'ont pas d'influence sur l'identifiant de base de la cellule. En fait nous pouvons remarquer que cette fonction de consensus permet d'identifier les faces horizontales et verticales des polygones. Nous avons obtenu le résultat de la figure 3.2 (b) en appliquant cette règle :

$$ConsensusV(c, d) : c.id \leftarrow \begin{cases} c.id \cup d.id & si \ d.pattern = c.pattern, \\ c.id & sinon. \end{cases}$$

Nous conservons les règles de calcul de distance et d'extraction définies dans la partie précédente et proposons la procédure complète locale à chaque cellule :

Algorithm 7 Pseudo-code area voronoi diagram (+Concavity) skeleton extraction

```
{1 Detection}
Detection(c)
Detection2(c)
{2 Identifier Consensus}
for all d ∈ c.neighbors do
    Consensus2(c,d)
end for
{3 Distance Computation}
D1(c)
{4 Skeleton Extraction}
Extraction(c)
```

4.3.3 Limites

Le modèle proposé possède ses propres limites, en effet la détection de faces sur des polygones complexes n'est pas satisfaisante. Des polygones dont les faces sont des lignes horizontales, verticales ou diagonales semblent par contre fournir des résultats corrects, la concavité des obstacles est ici prise en compte.

De plus la règle d'extraction présentée ici fournit des routes de navigation trop dilatées ce qui rend le squelette trop épais. D'autres critères d'extraction basés sur des mixages entre le critère d'identifiant et celui des distances permet d'obtenir un squelette plus fin. Nous présenterons une règle d'extraction plus efficace dans la section suivante de ce rapport.

Nous allons à présent passer à la quatrième partie de ce rapport, concernant la simulation des modèles présentés. Les simulations ont été réalisées avec SmartTiles une application en développement actuellement au sein de l'équipe MAIA.

5 Simulation

5.1 Introduction

Nous allons commencer cette partie en rappelant l'objectif de ce stage : l'élaboration d'une méthode de suivi d'un individu par un robot dans un environnement dynamique [9]. Les algorithmes présentés dans ce rapport pour l'élaboration d'un squelette de voronoi sur une carte permettent de créer un ensemble de routes de navigation sûres (à équidistance des obstacles). Le principe de navigation exploitant le squelette que nous retenons ici est le même que celui présenté dans [4]. Il s'agit ensuite de rattacher le but au squelette, propager un champ de potentiel depuis le but vers le squelette, puis dans le reste de la carte. Le robot n'aura plus qu'à suivre l'information de gradient du champ de potentiel pour rejoindre le squelette, naviguer sur le squelette, et finalement atteindre le but.

Nous allons surtout nous intéresser ici à la simulation de l'extraction du squelette sur une carte. Comme annoncé dans l'introduction de ce rapport, l'équipe MAIA possède un appartement intelligent équipé de capteurs au sol. Les capteurs au sol sont organisés en un réseau de dalles assimilable à un automate cellulaire, et chaque dalle est ainsi une unité de calcul à part entière. La méthode élaborée devra donc permettre de faire naviguer un robot dans cet appartement.

Cependant dans le cadre de ce stage, l'appartement n'est pas encore totalement équipé du dernier modèle de dalle intelligente, nous nous sommes donc orienté vers la simulation dans un premier temps.

5.2 Simulateur

Le simulateur utilisé pour les expériences présentées dans cette section est le simulateur SmartTiles (projet gforge.inria.fr), il s'agit d'une émulation de réseau de dalles intelligentes. Il est actuellement en développement au sein de l'équipe et l'ingénieur chargé de ce projet est Olivier Rochel. Mon travail a consisté à implémenter en python les modes de calcul synchrone et asynchrone simple, puis certains modèles d'extraction de squelette proposés pour validation par l'expérimentation. Dans la section suivante nous présenterons un ensemble de cartes, sur lesquelles nous avons extrait des squelette de voronoi.

5.3 Cartes

Les cartes que nous utilisons dans cette section proviennent de certains articles présentés ou de nos propres tests.

Dans un premier temps nous allons reprendre les cartes des tests pour l'extraction des bissectrices proposées dans [2], celles de [28] et une carte présentant un obstacle concave.

Nous lancerons l'exécution des 2 modèles (aire, aire+concavité) de la section précédente sur ces cartes et présenterons les résultats obtenus. Nous montrerons aussi les résultats obtenus après modification de la fonction d'extraction des deux premiers modèles. La nouvelle règle d'extraction associe un critère de distance à un critère de zone de diffusion. L'un permet d'isoler la bissectrice en distance impaire et l'autre en distance paire.

5.3.1 Expérimentation modèle 1 : Area Voronoi Diagram

On observe ci-après les bissectrices obtenues par application de la règle d'extraction basée uniquement sur les identifiants de diffusion, pour les modèles 1 et 2 de diagramme de voronoi d'aire.

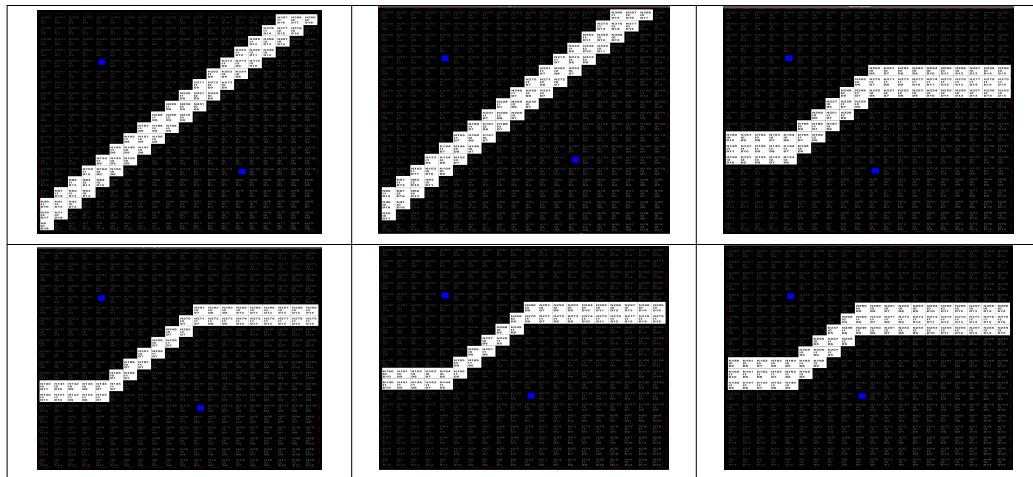


FIGURE 5.1: Bisector Extraction Cases for the Area Voronoi Diagram Model 1 and 2 (+Concavity)

La figure suivante représente des cartes sur lesquelles les algorithmes du modèle 1 de diagramme de voronoi d'aire ont été appliqués. Nous pouvons observer les squelettes extraits.

Les cartes a et d ne permettent pas d'obtenir le squelette, car une seule zone d'obstacle est identifiée. Les cartes c et e produisent un résultat particulier, deux zones seulement ont été identifiées.

Dans la carte e, le squelette n'apparaît pas dans l'obstacle de forme concave.

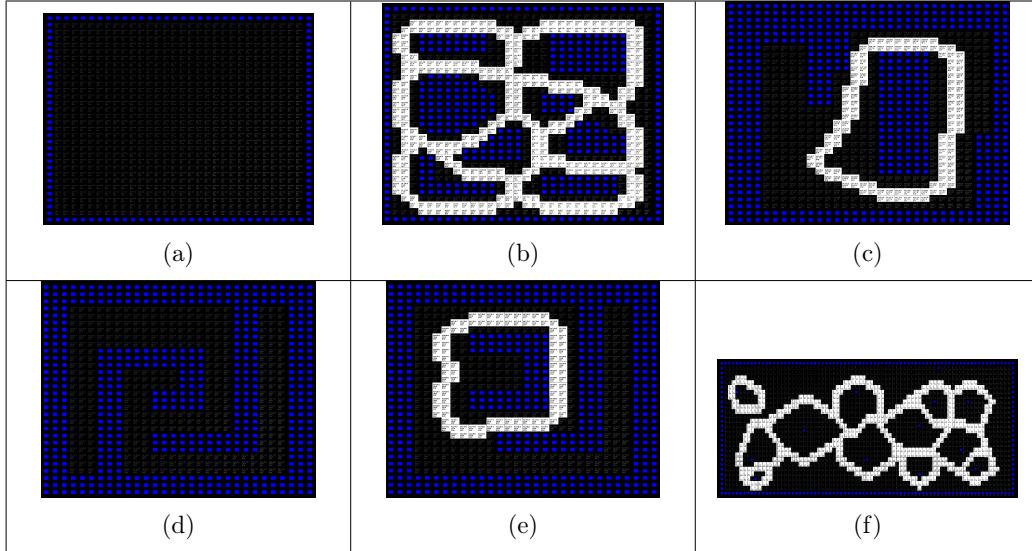


FIGURE 5.2: Map Skeleton Area Voronoi Diagram Model 1

5.3.2 Expérimentation modèle 2 : Area Voronoi Diagram + Concavity

Cette nouvelle figure représente des cartes sur lesquelles les algorithmes du modèle 2 de diagramme de voronoï d'aire augmenté ont été appliqués. Nous pouvons observer les squelettes extraits.

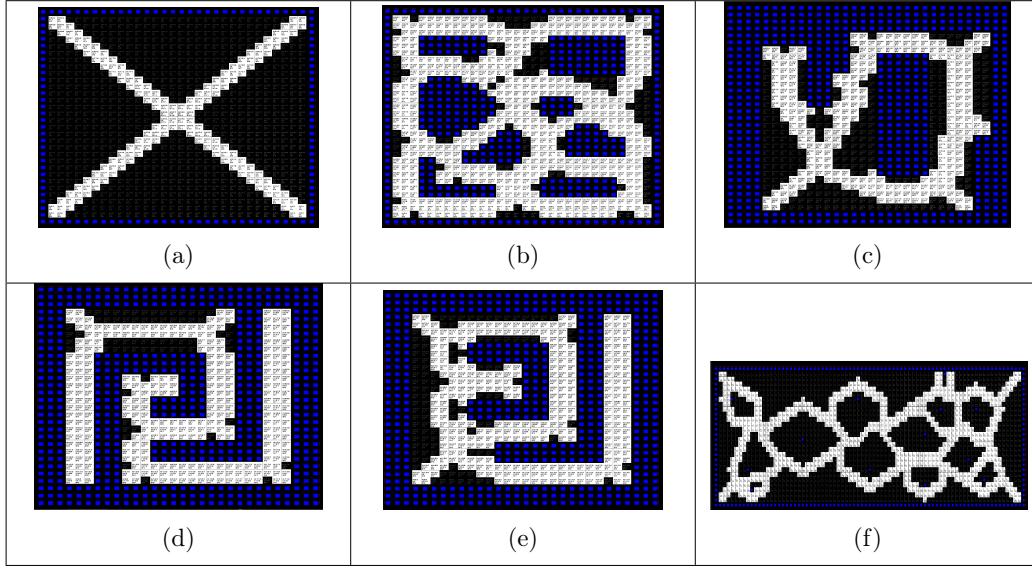


FIGURE 5.3: Map Skeleton Area Voronoi Diagram Model 2 (+Concavity)

Les résultats peuvent sembler satisfaisants, cependant le squelette est trop épais pour des cartes comme les cartes b, d et e, où les obstacles sont très proches sur la grille.

5.3.3 Expérimentation modèle 2 : nouvelle règle d'extraction (Dist*Id)

Nous utilisons ici une nouvelle règle d'extraction qui associe un critère de distance à un critère de zone de diffusion. L'un permet d'isoler la bissectrice en distance impaire et l'autre en distance paire. Cette nouvelle règle d'extraction permet d'obtenir un squelette plus fin que pour les modèles précédents. Nous pouvons observer en Fig. 5.4 les bissectrices en blanc sur la figure suivante, et la dilatation des bissectrices en jaune.

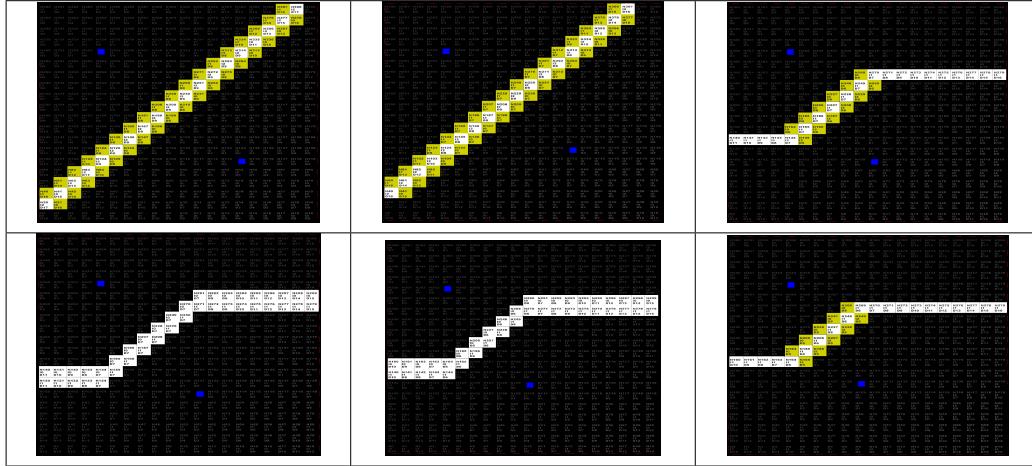


FIGURE 5.4: Bissector Extraction Cases for Dist*Id extraction rule(white)

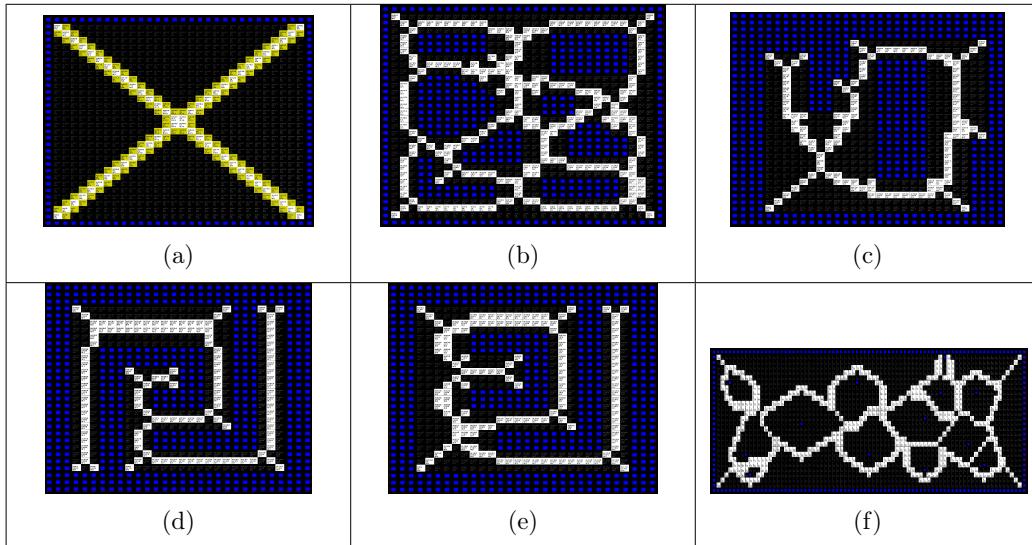


FIGURE 5.5: Map Skeleton Dist*Id (white)

Un squelette fin a été extrait sur chacune des cartes présentées.

5.3.4 Comparatif des différentes méthodes abordées dans ce papier

Cette section présente un tableau récapitulatif des différentes méthodes abordées dans ce rapport. Notre méthode est AreaVD2, elle satisfait les critères de calcul synchrone, asynchrone, sur une carte statique ou dynamique (déplacement d'obstacle). De plus une

règle de dilatation du squelette permet de satisfaire le critère de 4-connexité. Voici le tableau en question :

	Sync.	Async.	Centr.	Décentr.	Stat.	Dynam.	4-Conn.
AreaVD2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
Barraquand [4]	Ok	-	Ok	-	Ok	-	?
Tzionas [28]	Ok	-	-	Ok	Ok	-	-
Adamatzky [2]	Ok	-	-	Ok	Ok	-	-

6 Conclusion

Dans le cadre de ce stage, nous avons présenté un modèle de calcul du diagramme de voronoi sur une grille cellulaire en mode de calcul *asynchrone* contrairement aux autres modèles existants dans la littérature. Ce modèle s'implémente donc sur tout réseau cellulaire asynchrone comme par exemple les dalles intelligentes de l'appartement intelligent du Loria.

De plus, ce modèle a été implémenté en simulation, nous avons donc pu vérifier par l'expérimentation que ces algorithmes pouvaient extraire le squelette de voronoi sur des cartes d'obstacles.

L'expérimentation réalisée avec un premier modèle de déplacement d'obstacle (pas de déplacement Nord, Sud, Est et Ouest), montre que le squelette parvient à supporter cette dynamique de déplacement en se réorganisant (non illustré dans le rapport).

De nombreuses améliorations peuvent enrichir le modèle proposé, et les perspectives restent très larges.

D'un point de vue formel, la priorité à l'heure actuelle, serait de définir un certain nombre de critères permettant d'évaluer la convergence des algorithmes et leur robustesse.

Pour ce qui est de la partie pratique, une application de ce modèle à la navigation en milieu réel dans l'appartement intelligent de l'équipe MAIA, serait une bonne chose. Le retour d'expériences permettrait de cibler les parties du modèle à améliorer afin de continuer dans cette voie ou l'abandonner.

Aussi, il serait intéressant par exemple de proposer une approche utilisant des métriques discrètes approximant mieux la distance euclidienne que la distance de Manhattan. Les métriques basées sur les masques de chanfrein peuvent présenter une alternative intéressante, encore faut-il pouvoir les appliquer dans un mode de calcul asynchrone.

Une autre ouverture proposée serait d'utiliser un voisinage plus ou moins grand selon l'étape de calcul considérée. On pourrait aussi essayer d'élargir le sujet en proposant une méthode à déployer sur une grille non régulière.

Durant ce stage, j'ai pu acquérir de nombreuses connaissances théoriques dans des domaines variés de l'informatique, que ce soit au fil de mes lectures, ou lors des conférences et séminaires auxquels j'ai pu assister. D'un point de vue pratique, le codage des algorithmes vus et proposés a permis d'acquérir des connaissances en langage de développement python.

Annexe

Algorithm 8 Pseudo-code Algorithm computing the improved W-potential

```
1: {1 Initialization}
2: for all  $x$  in  $GW_{empty}$  do
3:    $x.V_p \leftarrow inf$ 
4: end for
5: {2 Connection of the goal to the skeleton}
6:  $x \leftarrow x_{goal}$ 
7: while  $x \notin S$  do
8:    $S \leftarrow S \cup x$ 
9:    $x \leftarrow argmax_{y \in x.neighbors}(y.d1)$ 
10: end while
11: {3 Computation of the W-potential in the augmented skeleton}
12:  $x_{goal}.V_p \leftarrow 0$ 
13:  $Q \leftarrow list(x_{goal})$ 
14:  $L_0 \leftarrow list()$ 
15: { $Q$  is a queue of points sorted by decreasing values of  $d1.$ }
16: repeat
17:    $x \leftarrow Q.pophead()$ 
18:    $L_0.append(x)$ 
19:   for all  $y \in x.dneighbors$  and  $y \in S$  do
20:     if  $y.V_p = inf$  then
21:        $y.V_p \leftarrow x.V_p + 1$ 
22:        $Q \leftarrow Q \cup y$ 
23:     end if
24:   end for
25: until  $Q = \emptyset$ 
26: {At the end of this step  $L_0$  contains all the points in  $S$  accessible from  $x_{goal}$ }
27: {4 Computation of the W-potential in the rest of  $GW_{empty}.$ }
28: for all  $i = 0, 1, 2$  until  $L_i = \emptyset$  do
29:    $L_{i+1} \leftarrow list()$ 
30:   for all  $x \in L_i$  do
31:     for all  $y \in x.neighbors$  and  $y \in GW_{empty}$  do
32:       if  $y.V_p = inf$  then
33:          $y.V_p \leftarrow x.V_p + 1$ 
34:          $L_{i+1} \leftarrow L_{i+1} \cup y$ 
35:       end if
36:     end for
37:   end for
38: end for
```

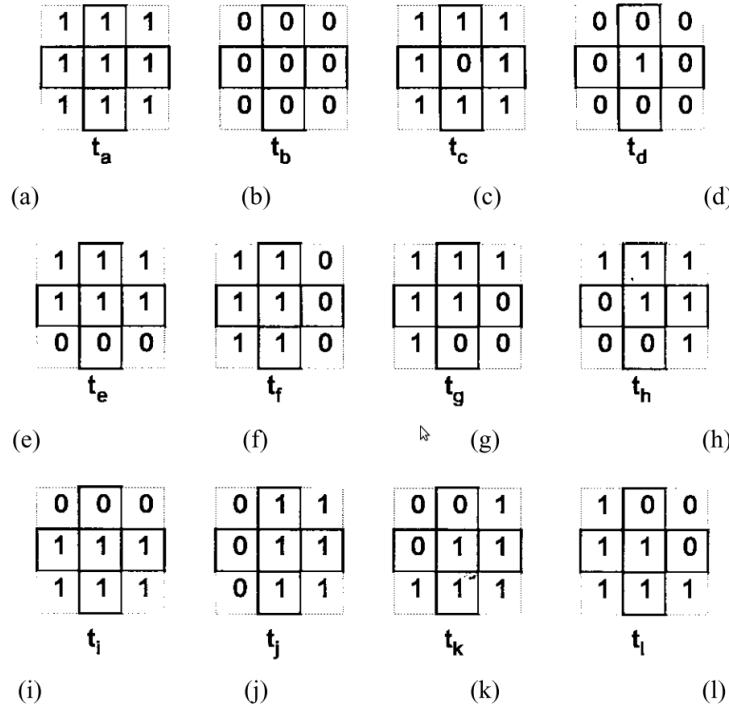


FIGURE 6.1: Ensemble des motifs proposés pour le codage des orientations de bordures [Tzionas97]

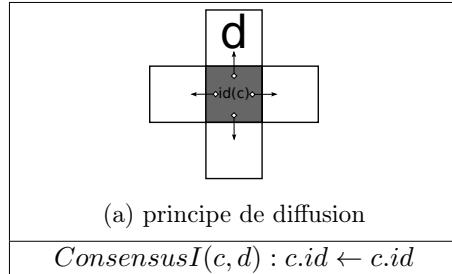


FIGURE 6.2: Classe I exemple

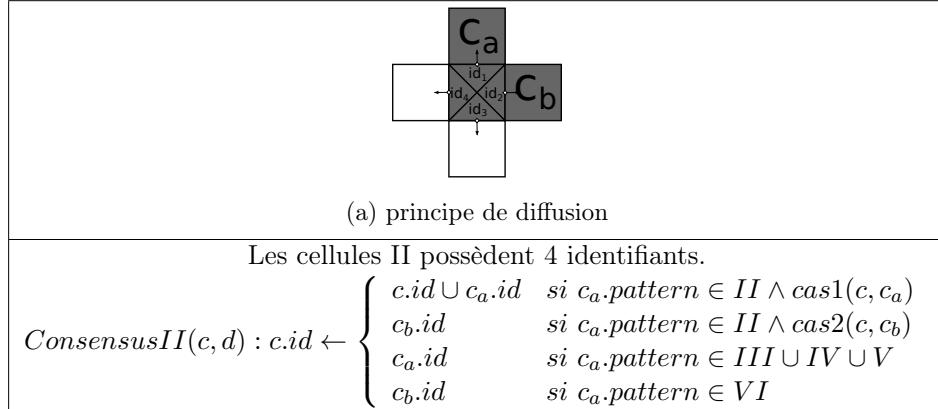


FIGURE 6.3: Classe II exemple

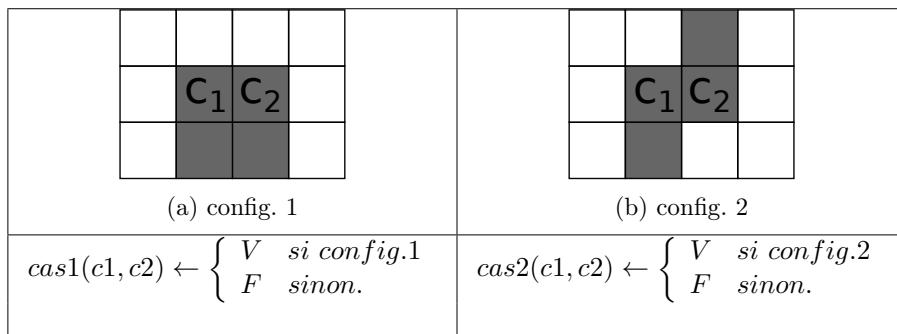


FIGURE 6.4: cas des coins

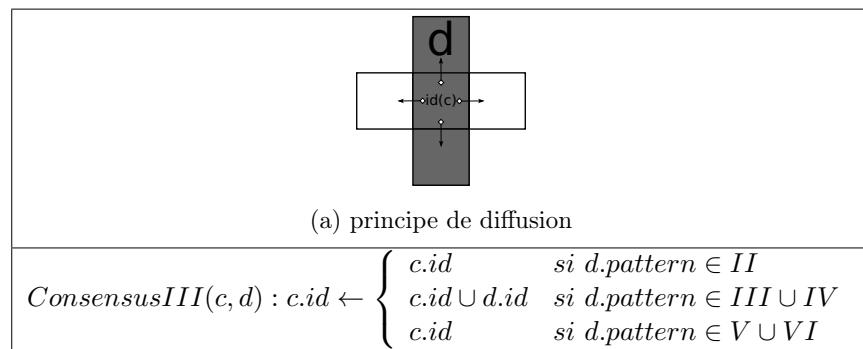
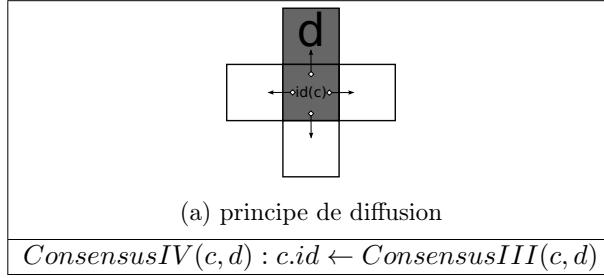
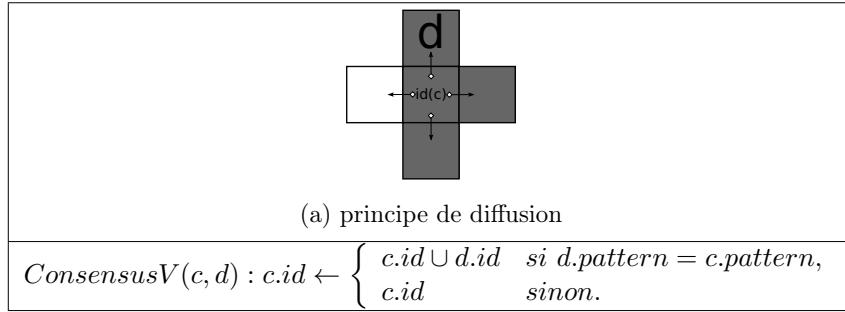


FIGURE 6.5: Classe III exemple



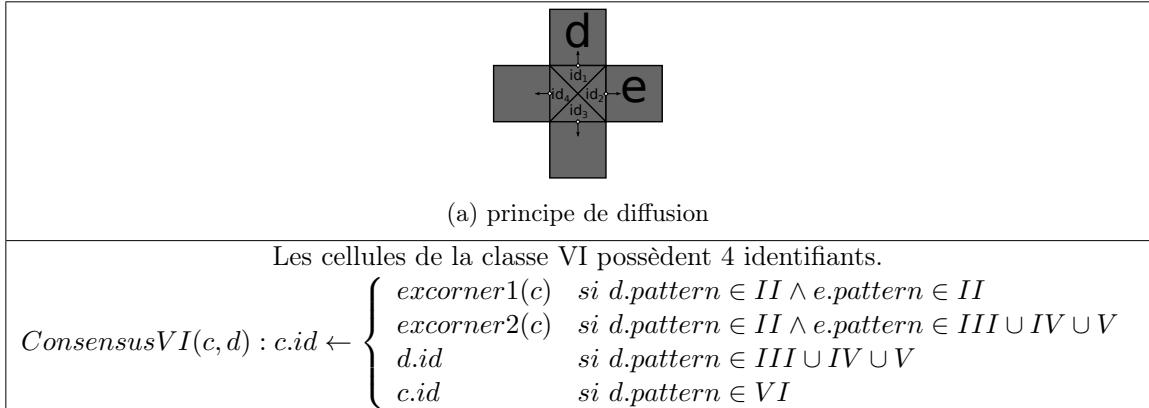
$$\text{ConsensusIV}(c, d) : c.id \leftarrow \text{ConsensusIII}(c, d)$$

FIGURE 6.6: Classe IV exemple



$$\text{ConsensusV}(c, d) : c.id \leftarrow \begin{cases} c.id \cup d.id & \text{si } d.pattern = c.pattern, \\ c.id & \text{sinon.} \end{cases}$$

FIGURE 6.7: Classe V exemple



$$\text{ConsensusVI}(c, d) : c.id \leftarrow \begin{cases} \text{excorner1}(c) & \text{si } d.pattern \in II \wedge e.pattern \in II \\ \text{excorner2}(c) & \text{si } d.pattern \in II \wedge e.pattern \in III \cup IV \cup V \\ d.id & \text{si } d.pattern \in III \cup IV \cup V \\ c.id & \text{si } d.pattern \in VI \end{cases}$$

FIGURE 6.8: Classe VI exemple

Bibliographie

- [1] Andrew Adamatzky and Benjamin de Lacy Costello. On some limitations of reaction-diffusion chemical computers in relation to Voronoi diagram and its inversion. *Physics Letters A*, 309(5-6) :397–406, March 2003.
- [2] a.I. Adamatzky. Voronoi-like partition of lattice in cellular automata. *Mathematical and Computer Modelling*, 23(4) :51–66, February 1996.
- [3] a.I. Adamatzky. Voronoi-like partition of lattice in cellular automata. *Mathematical and Computer Modelling*, 23(4) :51–66, February 1996.
- [4] J Barraquand, B Langlois, and J C Latombe. Numerical potential field techniques for robot path planning, 1991.
- [5] Priyadarshi Bhattacharya and Marina L. Gavrilova. Voronoi diagram in optimal path planning. *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, 1(c) :38–47, July 2007.
- [6] Harry Blum. A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form*, pages 362–380, 1967.
- [7] 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)*, 1 :28–31, 2002.
- [8] Gunilla Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3) :344–371, June 1986.
- [9] Arne Bosien, Marcus Venzke, and Volker Turau. A rewritable rfid environment for agv navigation. *Proceedings of the 5th*, 2008.
- [10] J.W. Brandt and V.R. Algazi. Computing a stable, connected skeleton from discrete data. *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 666–667, 1991.
- [11] Wai-Pak Choi, Kin-Man Lam, and Wan-Chi Siu. Extraction of the Euclidean skeleton based on a connectivity criterion. *Pattern Recognition*, 36(3) :721–729, March 2003.
- [12] Michel Couprie, David Coeurjolly, and Rita Zrour. Discrete bisector function and Euclidean skeleton in 2D and 3D. *Image and Vision Computing*, 25(10) :1543–1556, October 2007.
- [13] Jacques Ferber. *Multi-agent systems - an introduction to distributed artificial intelligence*. Addison-Wesley-Longman, 1999.

- [14] Marina Gavrilova and Jon Rokne. Swap conditions for dynamic Voronoi diagrams for circles and line segments. *Computer Aided Geometric Design*, 16(2) :89–106, February 1999.
- [15] M.L. Gavrilova and J. Rokne. Updating the topology of the dynamic Voronoi diagram for spheres in Euclidean d-dimensional space. *Computer Aided Geometric Design*, 20(4) :231–242, July 2003.
- [16] Frédéric Leymarie and Martin D Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1) :56–75, 1992.
- [17] Luidnel Maignan and Frederic Gruau. Convex Hulls on Cellular Spaces : Spatial Computing on Cellular Automata. *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 67–72, October 2011.
- [18] Marco Mamei and Franco Zambonelli. Spatial computing : The tota approach. *Computing*, pages 307–324, 2005.
- [19] R Ogniewicz and M Ilg. Voronoi skeletons : theory and applications. *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (June) :63–69, 1992.
- [20] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations : Concepts and applications of Voronoi diagrams*. Probability and Statistics. Wiley, NYC, 2nd edition, 2000. 671 pages.
- [21] Mark OzBench, Klaus Regenauer-Lieb, Dave R. Stegman, Gabriele Morra, Rebecca Farrington, Alina Hale, Dave a. May, Justin Freeman, Laurent Bourguin, Hans Mühlhaus, and Louis Moresi. A model comparison study of large-scale mantle–lithosphere dynamics driven by subduction. *Physics of the Earth and Planetary Interiors*, 171(1-4) :224–234, December 2008.
- [22] Nicolas Pépin, Olivier Simonin, and François Charpillet. Intelligent tiles - putting situated multi-agents models in real world. In *ICAART*, pages 513–519, 2009.
- [23] John L. Pfaltz. Sequential Operations in Digital Picture Processing. *Journal of the ACM*, 13(4) :471–494, October 1966.
- [24] Avneesh Sud, Naga Govindaraju, Russell Gayle, Ilknur Kabul, and Dinesh Manocha. Fast proximity computation among deformable models using discrete Voronoi diagrams. *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06*, page 1144, 2006.
- [25] H Sundar, D Silver, N Gagvani, and S Dickinson. Skeleton based shape matching and retrieval. *2003 Shape Modeling International*, 2003(15-12 may 2003) :130–139.
- [26] E. Thiel. *Les distances de chanfrein en analyse d'images : fondements et applications*. Thèse de Doctorat, Université Joseph Fourier, Grenoble 1, Sept 1994. <http://pageperso.lif.univ-mrs.fr/~edouard.thiel/these> .
- [27] E. Thiel and D. Coeurjolly. Distances discrètes. In D. Coeurjolly, A. Montanvert, and JM. Chassery, editors, *Traité IC2, Géométrie discrète et images numériques*, chapter 5, pages 127–150. Hermès, 2007. ISBN 978-2-7462-1643-3.

- [28] Panagiotis G Tzionas, Adonios Thanailakis, and Philippos G Tsalides. Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata. *IEEE Transactions on Robotics*, 13(2) :237–250, 1997.
- [29] Mirko Velić, Dave May, and Louis Moresi. A Fast Robust Algorithm for Computing Discrete Voronoi Diagrams. *Journal of Mathematical Modelling and Algorithms*, 8(3) :343–355, December 2008.
- [30] Ying-fung Wu. Rectilinear Shortest Paths and Minimum Spanning Trees in the Presence of Rectilinear Obstacles. *IEEE Transactions on Computers*, C-36(3) :321–331, March 1987.