وزارة التعليم العالي و البحث العلمي

Université Badji Mokhtar -Annaba-
Badji Mokhtar -Annaba- University

جامعة باجي مختار عنابة

Année : **2017**

Faculté des sciences de l'ingéniorat
Département d'informatique

THÈSE

Pour obtenir le diplôme de
Docteur $3^{ème}$ cycle

Modélisation et simulation d'un robot aspirateur domestique dans un environnement dynamique

Filière : Informatique
Spécialité : Reconnaissance des formes et intelligence artificielle R.F.I.A

Par :
Mohamed Amine YAKOUBI

Directeur de Thèse : **Mohamed Tayeb LASKRI** Prof. Université de Annaba

Devant le jury :

Président : Mohamed Tahar KIMOUR	Prof. Université de Annaba
Examinateur : Abdelkrim AMIRAT	Prof. Université de Souk Ahras
Examinatrice : Yamina MOHAMED-BENALI	Prof. Université de Annaba

Dédicace

Je souhaite dédier cette thèse, fruit de plusieurs années de recherches, à mes camarades doctorants et doctorantes auprès desquel(le)s, j'ai trouvé une bonne ambiance de travail et de convivialité. Je ne peux pas, bien évidemment, omettre de la présente dédicace tous les membres de ma grande famille, sans l'assistance multiforme de laquelle, ce travail n'aurait jamais été mené à son terme ni même vu le jour. Une mention particulière est dû à mon père, ma mère, mes frères et sœurs, mes oncles et tantes, mes cousins et cousines, ainsi que mes grands-parents.

.

Remerciements

Au terme de ma thèse que j'ai pu mener à bien, grâce à tous ceux et à toutes celles-parmi le corps enseignant, notamment et entourage universitaire, en général- qui m'ont aidé et/ou encouragé, je voudrai exprimer mes remerciements les plus sincères en particulier au professeur LASKRI Mohamed Tayeb, mon encadreur et mon soutien permanent, qui n'a pas ménagé son temps précieux et ses lourdes charges de sénateur d'état pour m'orienter et m'assurer toutes les conditions indispensables à mon travail.

Mes remerciements anticipés vont aussi aux honorables membres de Jyry qui ont bien voulu consentir de leur temps et de leurs autres obligations pour évaluer et juger le présent travail de recherche. Je veux citer nommément :

— Pr KIMOUR Mohamed Tahar, Professeur à l'université de Annaba (Président)
— Pr AMIRAT Abdelkrim, Professeur à l'université de Souk Ahras (Examinateur).
— Pr MOHAMED-BENALI Yamina, Professeur à l'université de Annaba (Examinatrice)
— Pr LASKRI Mohamed Tayeb, Professeur à l'université de Annaba (Rapporteur).

ملخص

منذ بروز المجتمع العلمي للذكاء الاصطناعي أصبحت الأنظمة و التطبيقات الروبوتيكية وسائل ضرورية في حياتنا اليومية ، فيما يخص مساعدة الأشخاص المسنين ، المركبات البحرية، روبوهات فلاحية، روبوهات التسوق المتحركة، روبوهات التنظيف ....الخ
في الوقت الحالي، ان مهمة تنظيف أي مساحة غرفة بواسطة روبوت منظف مستقل ، مبرمج أصلا باستعمال عدة مناهج.الا أن هذه الأخيرة ليست ناجعة ل وهذا بكل بساطة لأنها تهمل بعض العوامل كالمحيط المجهول و الحواجز المتحركة في الغرفة (محيط حركي) و تحسين معتبر في مسافة المسلك المقطوع و في عدد الدورات و مدة المهممة.
لاجتناب هذه التحديدات، فان أطروحتنا تقترح منهج جديد قصد ايجاد مسالك بدون اصطدام و التي تسمح للروبوت بواسطة أجهزة الاستشعار تغطية منطقة من مساحة الغرفة (تخطيط المسلك المتخطية).
يسمى المنهج المقترح (نموذج الشبكة العصبية ذات الذبذبات المتزاوجة) و المستوحاة عن نموذج ايكورن و الذي يضمن كذلك نتائج حسنة في عدة ميادين مثل التعرف على الصور و سرعة ايجاد المسلك الأقصر بأقل جهد.
ان نتائج دراسات المحاكاة و المقارنة تثبت فعاليات و صلابة المنهج المقترح حيث نلاحظ ان النموذج المقترح لا يستوجب معرفة مسبقة للمحيط المستهدف حيث يشتعل في زمن حقيقي و في محيط حركي الذي يسمح له بتقصير المسلك و مدة الحساب و عدد الدورات و عدد الخلايا الالمنظفة سابقا، قصد الحصول على مدة زمنية قصيرة مع استهلاك ضعيف للطاقة (البطاريات)

كلمات المفتاح: روبوت منظف، التغطيةالكاملةللمنطقة عند التجول ،نموذج الشبكة العصبية دات الدبدبات المتزاوجة ، محيط حركي.

Abstract

Since the emergence of the scientific community of Artificial Intelligence, robotic systems or applications have become indispensable tools in our daily life such as monitoring elderly people, underwater, aerial vehicles, agricultural robot, mobile shopping robot and vacuum cleaner robot.
The vacuum cleaner robot, for instance, depends on a mission which sweeps every accessible point in the entire environment. This mechanism is known as complete region coverage navigation (CRCN) of the target environment.

At present, a mission to clean a room floor with an Autonomous cleaner robot is typically planned by several methods or approaches. But, the latter, are not efficient for the CRCN simply because they neglect some parameters such as the unknown environment, the moving obstacles in the room (The Dynamic environment), the optimization of the generated path length, the number of turns and the time elapsed for the CRCN task.

To address these limitations, this thesis proposes new approach to find collision-free paths allowing a robot to cover an area of the room floor with its sensors, which is known as coverage path planning. The proposed approach is named the Pulsed Coupled Neural Network model (PCNN) which is inspired from Eckhorn model (1990) and it yielded also warrant results in many fields, such as image recognition and finding the shortest path rapidly with minimum effort.

The results of simulation and comparison studies demonstrate the effectiveness and efficiency of the proposed approach, where we notice that the proposed model does not require prior knowledge of the target environment, so it works in real time and in a dynamic environment in which it can reduce the length of the trajectory, the CPU time, the number of turns and repeated cells cleaned so as to get a short elapsed time of CRCN and a low consumption of robot batteries.

Key words: Vacuum cleaner robot, Complete Region Coverage Navigation (CRCN), Pulse Coupled Neural Network (PCNN), Dynamic Environment.

Résumé

Depuis l'émergence de la communauté scientifique de l'intelligence artificielle, les systèmes ou les applications robotiques sont devenus des outils incontournables dans notre vie quotidienne, tels que l'assistance aux personnes âgées, les véhicules sous-marins et aériens, les robots de l'agriculture, les robots shopping mobiles, les robots aspirateurs etc. Le robot aspirateur, par exemple, a la mission de balayer tous les points accessibles dans tout l'environnement. Cette tâche est connue en tant que Navigation par Couverture Complète d'une Région (CRCN) de l'environnement cible.

À l'heure actuelle, la mission de nettoyage de toute surface d'une pièce avec un robot aspirateur autonome, est typiquement planifiée par plusieurs méthodes. Mais, ces dernières, ne sont pas efficaces pour le CRCN simplement parce qu'elles négligent certains paramètres tels que l'environnement inconnu, les obstacles mobiles dans la pièce (environnement dynamique), l'optimisation de la longueur de trajectoire parcourue, le nombre de tours et le temps de mission CRCN.

Pour palier à ces limitations, notre approche dans cette thèse est de proposer une nouvelle méthode pour trouver des chemins sans collision permettant au robot avec ses capteurs de couvrir une région de la surface de la pièce (planification de la trajectoire de couverture). La méthode proposée est appelée le Modèle de Réseau Neuronal à Impulsions-Couplées (PCNN) qui est inspirée du modèle Eckhorn (1990) et qui garantit aussi de bons résultats dans de nombreux domaines, telles que la reconnaissance des formes et la rapidité de trouver le chemin le plus court avec un minimum d'effort.

Les résultats des études de simulation et de comparaison démontrent l'efficacité et la robustesse de la méthode proposée où nous remarquons que le modèle proposé ne nécessite pas une connaissance préalable de l'environnement-cible, de sorte qu'il fonctionne en temps réel et dans un environnement dynamique dans lequel il peut réduire la longueur de la trajectoire, le temps de calcul, le nombre de tours et le nombre des cellules déjà nettoyées, de manière à obtenir un temps court avec une faible consommation d'énergie (batteries).

Mot clés : Robot aspirateur, Navigation par Couverture Complète d'une Région (CRCN), réseau neuronal à impulsions-couplées (PCNN), environnement dynamique.

Table des figures

1.1 Le robot “SHAKEY” de l’Institut de Recherche de Stanford 3
1.2 L’architecture tripartie 5

2.1 "Tondeuse de la pelouse". La zone ombrée indique la zone déjà couverte (plus sombre) et la zone qui sera couverte (plus léger) lorsque le robot se termine en suivant le chemin. 12
2.2 Exemple de décomposition trapézoïdale d’un graphe d’adjacence correspondant. 13
2.3 Une décomposition avec moins de cellules. (a) Une bande supplémentaire est nécessaire dans la décomposition trapézoïdale (b) par rapport à la décomposition boustrophédonne. 14
2.4 Limites cellulaires de décomposition à base Morse sont placées à des points critiques, où la surface normale de l’obstacle est perpendiculaire à la tranche de balayage, et en parallèle à la direction de balayage. 16
2.5 Détermination cellulaire par la décomposition à base de Morse boustrophedonne. 17
2.6 Processus de Boustrophedon pour la construction du chemin, où δ est l’espacement inter-tour et λ est le paramètre de tranche. 18
2.7 Exemple de décomposition à base de Morse de graphe d’adjacence. 19
2.8 Décomposition Spirale à base Morse obtenue en utilisant la fonction de Morse. 20
2.9 Détection de point critique sur le rayon du robot. 21
2.10 Avec le décomposition à base Morse, une trajectoire en zigzag simple d’un robot, va manquer les points critiques dans la figure à moins qu’il effectue le mur suivant à la fois sur le haut et le bas d’une cellule. 22
2.11 Un chemin constitué de cycles rectangulaires permet la détection de tous les points critiques. Ce modèle est utilisé dans la décomposition à base Morse en ligne et l’algorithme de CCR. 23
2.12 Détection de point critique en utilisant l’algorithme de cycle. 24

2.13 Exemple de décomposition Morse avec le graphe de Reeb. Cp_1. . . CP_6 se sont les points critiques. 25
2.14 Un point critique concave ne sera pas détecté si la courbure de la limite est inférieure à la périphérie du robot et conduira à un graphe de Reeb incohérent. Tel est le cas de cet exemple, Cp_2 dans l'environnement, ce qui ne se produit détecté et un bord incorrect émanant Cp_3. 26
2.15 Combinaison de décomposition Morse et GVD pour la couverture du capteur de rayon étendu. 27
2.16 Événements (Repères) dans la décomposition de la tranche. 28
2.17 État diagramme de transition de l'algorithme de couverture topologique. . 29
2.18 CC_R utilise une décomposition cellulaire exacte pour les environnements rectilignes. 30
2.19 Un exemple de carte de grille. les cellules de la grille avec des obstacles présents sont grisés. 31
2.20 Planification de chemin de couverture en utilisant l'algorithme de bloc d'ondes pour un exemple d'environnement. 32
2.21 Planification de chemin de couverture en utilisant l'algorithme Spiral-STC. 34
2.22 Schéma du réseau de neurones utilisé par Luo, Yang et d'autres pour parvenir à une couverture. 36

3.1 structure de base du modèle PCNN 43
3.2 Modèle compartmental biologique. 43
3.3 Circuit équivalent du modèle compartmental. 44
3.4 Modèle mathématique d'un PCNN. 46
3.5 Applications de PCNN. 47
3.6 Ségmentation d'une image. 49
3.7 Topologie de PCNN proposée. 50
3.8 Le poids de connexion de réseau. 51
3.9 Planification de plus court chemin. 52

4.1 Roue standard fixe. 56
4.2 Roue standard orientable. 56
4.3 Roue de Caster. 57
4.4 Roue suédoise. 57
4.5 Roue sphérique. 58
4.6 Position du robot dans le repère de plateforme locale. 60
4.7 Position du robot dans le repère de plateforme locale. 61
4.8 Modélisation du robot. 62
4.9 Environnement du robot. 63

4.10 Façons de couverture complète (a) trajectoire spirale (b)trajectoire zigzag. 64
4.11 Sélection de la nouvelle position du robot. 66
4.12 La planification de chemin en utilisant l'algorithme de couverture complète. 67
4.13 une planification du chemin pour une couverture complète d'un environnement. 70

5.1 Couverture complète d'une surface connue (a) Chemin planifié généré (b) Temps d'activation pour la planification du chemin par le robot (c) Landscape de la fonction neuronale should θ lorsque le robot atteint $D(6, 17)$. . 75
5.2 Lorsque le robot atteint la position $D(6, 17)$ dans un environnement inconnu (a) Chemin planifié généré (b) Temps d'activation pour la planification du chemin par le robot (c) Landscape de la fonction neuronale should θ. 77
5.3 Couverture complète d'un environnement inconnu (a) Chemin planifié généré (b) Temps d'activation pour la planification du chemin par le robot (c) Landscape de la fonction neuronale should θ. 78
5.4 Lorsque le robot atteint la position $J(23, 13)$ dans un environnement dynamique (a) Chemin planifié généré (b) Temps d'activation pour la planification du chemin par le robot. 79
5.5 L'activité neuronale lorsque le robot évite le nouveau obstacle dans. (a) position (24, 15) (b) position d'obstacle (23, 16). 80
5.6 Couverture complète d'un environnement dynamique (a) Chemin planifié généré (b) Landscape de la fonction neuronale should θ. 81
5.7 Couverture complète par l'algorithme LSSP. 82
5.8 Couverture complète par l'algorithme BA*. 83
5.9 Couverture complète par l'algorithme CAC. 84
5.10 Les cas de couverture complète d'un environnement inconnu en utilisant (a)Méthode proposée. (b)Algorithme CAC. (c)Algorithme BA*. (d)Algorithme LSSP. 88
5.11 Les cas de couverture complète d'un environnement inconnu en utilisant (a)Approche proposée. (b)Modèle Luo et Yang. 89

Liste des tableaux

5.1 Configuration matérielle pour la simulation 73
5.2 Comparaison entre les algorithmes LSSP, BA*, CAC et la méthode proposée pour la couverture complète illustrée sur la figure 5.10 84
5.3 Comparaison entre le modèle Luo et Yang avec Approche proposée dans la couverture complète illustrée sur la figure 5.11 86

Table des matières

Dédicace i
Remerciements ii
Résumé en Arabe (Abstract in Arabic) iii
Abstract iv
Résumé v
Table des figures vii
Liste des tableaux x
Table des matières xi

1.

Introduction Générale 1
1.1 Introduction 2
1.2 Contexte et problématique de la thèse 2
1.3 Organisation de la thèse 6

2. *Chapitre 2*
La couverture complète d'un environnement (planification du chemin) 8
2.1 Introduction 9
2.2 Le problème de la planification du chemin pour la couverture complète . . 9
2.3 Décomposition cellulaire exacte 11
2.3.1 Décomposition trapézoïdale 12
2.3.2 Décomposition boustrophédonne 13
2.4 Décomposition cellulaire à base de morse 14
2.4.1 Décomposition boustrophédonne basée Morse en ligne 18
2.4.2 Décomposition cellulaire basée-Morse Combinée avec le diagramme de Voronoi généralisé 21
2.5 Couverture topologique basée repère 23
2.5.1 Décomposition de Tranche 24
2.5.2 Algorithme de couverture topologique en ligne 25
2.6 Couverture des environnements rectilignes basée sur le contact du capteur (CC_R) 27

2.7 Méthodes basées-grille 28
2.7.1 Couverture basée-grille en utilisant l'algorithme de bloc d'ondes . . 30
2.7.2 Couverture basée-grille en utilisant la construction des arbres 31
2.7.3 Couverture basée-réseau de neurones sur cartes de grille 33
2.7.4 Décomposition hexagonale de Grille pour les robots équipés de capteurs de vision latérale 35
2.8 Couverture basée-graphe 36
2.9 Couverture optimale 37
2.10 Couverture dans l'incertitude 38
2.11 Discussion & Conclusion 39

3. Chapitre 3
PCNN : Un réseau neuronal à impulsions-couplées 41
3.1 Introduction 42
3.2 Architecture du réseau 42
3.3 Modèle mathématique d'un PCNN 44
3.4 Modifications au niveau du modèle PCNN 45
3.5 Applications du modèle PCNN 46
3.5.1 La segmentation par le modèle PCNN 47
3.5.2 Planification du chemin entre deux positions par le modèle PCNN . 48
3.6 Conclusion 51

4. Chapitre 4
Modélisation et simulation du modèle PCNN 53
4.1 Introduction 54
4.2 Modélisation cinématique 54
4.2.1 Contraintes cinématiques 55
4.2.2 Conception des roues 55
4.2.3 Robots mobiles à roues 58
4.2.4 Position du robot 59
4.2.5 Robot mobile de type tricycle 60
4.2.5.1 Centre instantané de rotation (CIR) 60
4.3 Architecture du robot 62
4.4 Modélisation de l'environnement 62
4.5 La couverture complète 63
4.5.1 Le modèle PCNN 64
4.5.2 Algorithme de Couverture complète 66
4.6 Planification du chemin pour la couverture complète 68
4.7 Conclusion 69

5. Chapitre 5

Tests et Résultats 71
5.1 Introduction 72
5.2 Application développée 72
5.3 Configuration du matériel 73
5.4 Configuration paramétrique 73
5.5 Tests dans différents environnements 74
5.5.1 Dans un environnement connu 74
5.5.2 Dans un environnement inconnu 75
5.5.3 Dans un environnement dynamique 76
5.6 Comparaison avec autres méthodes 78
5.6.1 Comparaison avec les algorithmes LSSP, BA* et CAC 79
5.6.1.1 Algorithme LSSP 80
5.6.1.2 Algorithme BA* 80
5.6.1.3 Algorithme CAC 80
5.6.1.4 Test de comparaison 81
5.6.2 Comparaison avec la méthode Luo et Yang 84
5.7 Conclusion 86

6. Chapitre 6

Conclusion Générale 90
6.1 Conclusion générale 91
6.2 Résumé des parties traitées tout au long de cette thèse 91
6.3 Perspectives 93

Bibliographie 102

Introduction Générale

1.1 Introduction

Depuis toujours, l'homme a beaucoup réfléchi pour résoudre ses différents problèmes. Parfois, il se retrouve incapable de réaliser des actions difficiles, surtout dans des environnements complexes ou dangereux. L'évolution de son intelligence lui a permis d'avoir envie et besoin de disposer d'un outil performant pour pouvoir exécuter diverses actions. Ce besoin humain a incité et encouragé la recherche scientifique qui a abouti à l'invention du robot.

Mais du rêve à la réalité, il a fallu aller toujours plus loin dans la recherche, une fois ouvert le domaine de la robotique.

Avec l'apparition des nouvelles technologies dont l'électronique, la communauté scientifique de l'intelligence artificielle a vu dans la robotique un domaine d'application particulièrement riche pour élaborer ses propres recherches.

Aujourd'hui, les systèmes robotisés sont devenus des outils indispensables dans toutes les branches de l'industrie moderne. Leur intérêt réside dans leur mobilité qui ouvre des applications dans divers domaines. Ils sont destinés à des missions complexes telles que, le nettoyage dans des environnements dangereux, y compris dans le domaine spatial.

Nous nous sommes intéressés, dans le cadre de notre travail de recherche, à la conception, la modélisation et à la simulation d'une approche de planification du chemin dans un environnement dynamique pour le nettoyage par un robot aspirateur.

Dans cette partie introductrice de notre thèse, nous présenterons d'abord le contexte et la problématique de notre thèse ; par la suite nous décrivons notre contribution dans ce travail. La méthode proposée de cette planification du chemin pour le robot aspirateur a été validée par les résultats de plusieurs simulations des environnements. Elle est considérée comme une partie importante dans la réalisation de ce travail. Nous décrirons ensuite le plan de cette thèse.

1.2 Contexte et problématique de la thèse

Bien que les ordinateurs soient aujourd'hui capables de résoudre de nombreux problèmes, ils restent encore en-de-ça de l'intelligence humaine.

La communauté de l'intelligence artificielle tente de se rapprocher de l'intelligence humaine en conservant et en élaborant des machines de plus en plus pratiques et plus

puissantes .

Ce rêve a pu enfin se réaliser dés l'apparition du premier robot mobile SHAKEY de l'Institut de Recherche de Stanford, en 1968 [1]. Figure1.1

Le mot robot, dont l'origine est robota, signifie travail obligatoire. Il a été inventé par l'écrivain tchèque Karel Cpaek pour les besoins de sa pièce de théâtre intitulé Rossum's Unversal Robot, en 1929.

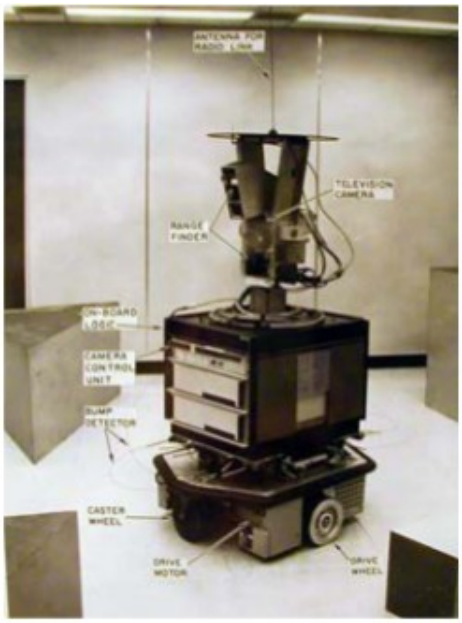


FIGURE *1.1* – Le robot "SHAKEY" de l'Institut de Recherche de Stanford

L'Association Japonaise de Robots Industriels (AJRI) a réparti les robots en 6 classes : robot de manipulation manuelle, à séquence fixe,à séquence variable, robot playback, robot à commande numérique et robot intelligent.

Au file des années, le domaine de la recherche en robotique s'est élargi et couvre désormais non seulement des applications industrielles mais également un large éventail pour des applications, en transformant des tâches répétitives industrielles en des tâches plus autonomes, sans interaction entre les robots et les utilisateurs.

L'autonomie et l'intelligence dans ce cas signifient que le robot est capable de déterminer sa tâche par son propre raisonnement, plutôt que d'en suivre un prédéfini par des instructions.

Lors de notre investigation de l'état de l'art sur la robotique mobile, nous avons rencontré divers types de robots destinés à diverses applications telles que robots à roues, bipèdes, quadrupèdes, sous-marins, aériens ainsi que des robots aspirateurs dédiés au nettoyage.

Ces robots ne peuvent pas exécuter une tâche correctement sans avoir "une idée" sur l'état de son environnement ; ce qui exige un système de localisation et une modélisation de cet environnement.
La localisation se base sur la fonction de perception ; cette dernière s'appuie sur la physique des capteurs, le traitement du signal et de l'image.
Un robot mobile autonome a alors besoin de plusieurs capteurs *proprioceptifs* et *extéroceptifs* pour la perception et la localisation.

A partir de ses capteurs, le robot peut avoir la capacité de se localiser, c'est à dire de connaitre en permanence sa position et l'état de son environnement basée sur une cartographie globale et une cartographie locale. Dans le cas de la cartographie globale, l'environnement est entièrement connu avant l'exécution de la tâche. Par contre, dans la cartographie locale, l'information sur l'environnement est inexistante ou incertaine (semi-complète ou toute incomplète).

L'environnement le plus compliqué dans le domaine de la robotique mobile est celui qui contient des obstacles susceptibles de changer leurs positions (se déplaçant), leurs formes (un piéton tenant un chien), ou d'apparaitre et disparaitre (une porte coulissante dans un mur) durant l'activité du robot. Ce type d'environnement s'appelle environnement *dynamique*.

Telle que affirmée par les auteurs concernés [2], il existe une relation entre ces trois phases (la tâche , le robot et l'environnement) ; Cela signifie que le robot dépend de l'environnement où il effectue sa tâche. Ces trois (03) phases dépendent les uns des autres et s'influencent mutuellement (Figure. 1.2).

Un robot mobile autonome a alors la capacité de se déplacer dans son environnement pour effectuer un certain nombre de tâches différentes ; il est aussi capable de s'adapter aux changements de son environnement, d'accueillir un apprentissage lui permettant de modifier son comportement en conséquence, et de construire des représentations internes de son monde qui peuvent être utilisées pour des processus de raisonnement comme la navigation.

Cette navigation ou planification d'une trajectoire (entre autres) est un problème com-

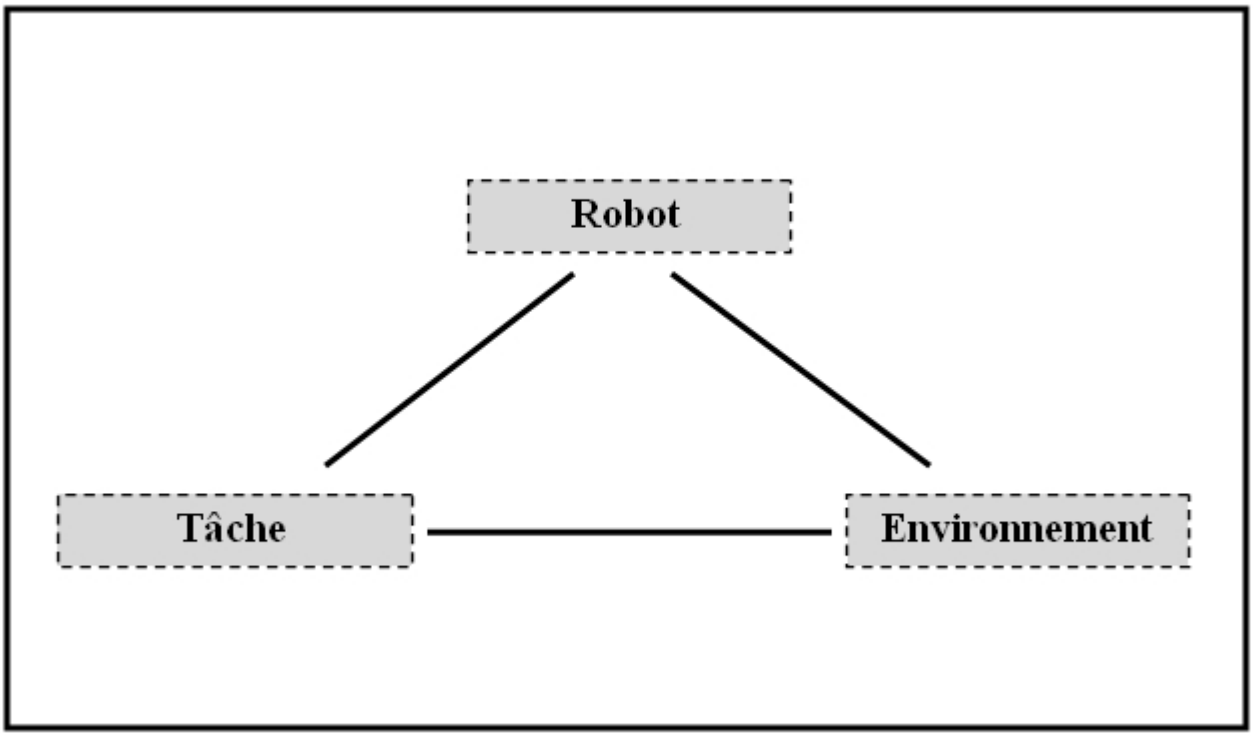


FIGURE *1.2* – L'architecture tripartie

plexe dans l'autonomie et la mobilité des robots.

Ce problème de la navigation d'un robot mobile autonome est un sujet d'investigation intéressant au plus haut point. Tout notre travail consistera donc à simuler ce robot autonome.

La problématique abordée dans notre thèse est celle du : *le nettoyage d'un environnement dynamique par un robot aspirateur mobile autonome.*
La question qui se pose est *est-ce-que l'intelligence artificielle de ce robot aspirateur peut être proche de l'intelligence de l'être humain ou non ?* En d'autres termes, l'objectif est de développer une ou des technique(s) qui permettent au robot aspirateur le déplacement dans un environnement, d'une manière autonome, c-à-d dans un environnement qui comporte des obstacles dans des situations où l'intervention humaine est absente.

A propos de cette problématique, il faut signaler que la tâche exacte du robot aspirateur s'appelle la couverture complète d'une pièce dans laquelle le robot doit déterminer un chemin qui passe sur tous les points d'une surface ou d'un espace d'intérêt, tout en évitant les obstacles.

Pour traiter cette problématique, nous avons proposé une méthode de planification de

chemin basée sur un modèle de réseau de neurones appelé réseau neuronal à impulsions-couplées "Pulsed-Coupled Neural Network" (PCNN).

Dans ce cadre, ce modèle a fourni un moyen simple et efficace pour étudier la dynamique des impulsions synchrones dans les réseaux.
Par ce travail, il s'agit pour nous d"essayer de contribuer aux autres travaux qui se sont intéressés à la planification du chemin pour le nettoyage d'une salle par un robot aspirateur autonome.

1.3 Organisation de la thèse

Comme indiqué précédemment, cette thèse traite du thème *d'un robot aspirateur qui nettoie un environnement dynamique* ; elle vise à présenter les résultats de notre travail en-guise référence, qui pourrait inspirer d'autres travaux de recherches sur la question. Elle est structurée en six chapitres comme suit :

La thèse débute par une introduction générale relative au traitement de la problématique.

Le deuxième chapitre présente un état de l'art sur l'opération de la couverture complète d'un environnement par un robot mobile autonome. Le chapitre, dans une première partie, présente les notions de base de la couverture complète (types d'environnement, exigences d'un robot aspirateur, chemin optimal). La seconde partie de ce chapitre est consacrée à la présentation des projets et des travaux, d'autres chercheurs,effectués dans le domaine de cette planification du chemin.

Le troisième chapitre porte sur la présentation du modèle proposé appelé réseau neuronal à impulsions-couplées (PCNN). Dans une première partie de ce chapitre, nous présentons son architecture et son modèle mathématique durant toutes ses modifications. Nous concluons ce chapitre par une seconde partie qui énumère quelques applications et techniques de ce modèle PCNN (Traitement d'image, déplacement d'un robot).

Le quatrième chapitre est consacré à la conception et la modélisation de notre solution proposée (PCNN). En fait, après la modélisation cinématique du robot aspirateur, une représentation de l'environnement est validée en le divisant en plusieurs disques (soit un obstacle ou un emplacement libre). Ce chapitre décrit notre nouvelle approche ou technique de planification du chemin pour la couverture complète d'un environnement. Dans la suite de ce chapitre, nous décrivons l'architecture du modèle PCNN proposé. Par la suite nous présentons de manière détaillée l'algorithme et la planification du chemin pour l'opération de nettoyage avec évitement d'obstacles.

Le cinquième chapitre, décrit l'évaluation de notre approche analytiquement par une simulation de notre algorithme proposé dans des états différents de l'environnement (une

pièce connue, complètement inconnue, dynamique.). L'objectif de ces simulations consiste à étudier et tester l'efficacité et la robustesse de notre algorithme et modèle proposés dans le chapitre précédent, une autre analyse comparative est présentée à la fin de ce chapitre.

Enfin, nous terminons cette thèse par une conclusion générale qui synthétise l'ensemble de nos contributions et suggère quelques perspectives que nous envisageons de poursuivre dans nos futurs travaux de recherche.

Chapitre 2

La couverture complète d'un environnement (planification du chemin)

2.1 Introduction

Ce chapitre présente un panorama des travaux antérieurs dans la planification de chemins pour une couverture complète d'un environnement par un robot mobile, l'objet de la thématique de notre thèse. Comme mentionné précédemment, la planification du chemin de couverture est la tâche qui consiste à déterminer un chemin qui passe sur tous les points d'une surface ou d'un espace d'intérêt, tout en évitant les obstacles. Cette tâche fait partie intégrante de nombreuses applications robotiques, tels que la surveillance de personnes âgées [3], sous-marins [4], véhicules aériens [5], robots agricoles [6], robot de shopping [7] ainsi que robot aspirateur [8]. L'état de l'art que nous présentons couvre la théorie de la planification de la trajectoire pour la couverture et des applications dans le domaine général de la robotique.
Après l'introduction au problème de la planification du chemin pour la couverture complète dans la section 2.2, nous présentons un examen approfondi des méthodes de planification de chemin de couverture pour des espaces à 2 dimensions (Sections 2.2,-2.7), qui comprennent des blocs de construction fondamentaux des algorithmes développés dans cette thèse. Suivront un examen des méthodes de planification de chemin de couverture aboutissant à la réalisation d'une couverture optimale (section 2.9) et la prise en compte des incertitudes et de la nécessité de réduire l'accumulation d'erreur de localisation lors de l'exécution du chemin de couverture prévue (section2.10).
Nous terminons ce chapitre par une discussion sur les méthodes examinées ouvrant des voies permettant de poursuivre la recherche (Section 2.11).

2.2 Le problème de la planification du chemin pour la couverture complète

Comme indiqué plus haut, la problématique de la planification du chemin pour la couverture complète d'un environnement nécessite comme condition indispensable de trouver un chemin sans collision qui permet à un robot de passer au-dessus de tous les points dans un espace ou volume d'intérêt cible. Dans l'une des premières œuvres sur cette planification dans la littérature, les exigences que doit respecter un robot pour effectuer une opération de couverture ont été définies dans [9].
Dans le présent document nous allons traiter du robot mobile qui se déplace dans un environnement à deux dimensions et qui est soumis aux différentes exigences suivantes :

1. Le robot doit passer par tous les points couvrant complètement la zone cible.
2. Le robot doit couvrir la surface sans chevauchement des chemins.
3. L'opération de couverture continue et séquentielle sans aucune répétition des chemins, est nécessaire.

4. Des trajectoires avec un mouvement simple (par exemple, des lignes droites ou des cercles) doivent être utilisés pour plus de simplicité dans le contrôle.
5. Le robot doit éviter tous les obstacles.
6. Un chemin optimal est souhaité dans certaines conditions.

Cependant, dans des environnements complexes, il n'est pas toujours possible de satisfaire tous ces critères. Par conséquent, parfois une considération prioritaire est nécessaire. Le problème de la planification du chemin de couverture est liée au problème de satisfaction de contraintes (CSP), une variante du problème du voyageur de commerce (TSP), où, un agent doit visiter toutes les villes [10]. Rappelons que, compte tenu de la liste des villes et les distances entre chaque paire de villes, la TSP appelle à la route la plus courte qui visite chaque ville exactement une fois et revient à la ville de départ. Durant cette planification du chemin, l'agent doit passer par tous les points dans la zone cible au lieu de visiter tous les voisins. Depuis, la TSP est NP-complet, le temps de calcul nécessaire pour résoudre le problème augmente considérablement lorsque la dimension du problème augmente aussi.
En effet, la planification d'un chemin pour tendre toute l'herbe d'une surface donnée couverte est connue sous l'expression «problème de la tondeuse", et prouvé par NP-complet [11]. A noter que le problème de la tondeuse ne tient pas compte des obstacles. Il en est de même pour "le problème de piano-mover» qui est approuvé pour être PSPACE-complet, ce qui implique NP-complet [12].
Deux autres problèmes similaires supplémentaires liés à cette planification sont le problème de "la galerie d'art" et "le veilleur de la route". Le premier exige un nombre minimum de gardiens à la station dans une galerie polygonale de sorte que chaque point dans la galerie est visible par au moins un gardien [13]. Le deuxième exige le chemin le plus court entre un point donné et le veilleur lui-même de telle sorte que chaque point dans un espace donné est visible à partir d'au moins un point de la route [14].
En général, nous pouvons considérer que ces deux problèmes de la galerie d'art et du veilleur de la route sont NP-complet et qu'ils ont des algorithmes pour la couverture complète d'un environnement.
Les algorithmes de couverture peuvent être classés comme heuristique ou complet qui garantissent ou non mathématiquement une couverture complète de l'espace libre. Indépendamment de cela, ils peuvent être classés comme étant soit offline ou online.
Cette classification a été initialement proposée dans [15]. Les algorithmes offline reposent uniquement sur les informations stationnaires ; dans ce cas, l'environnement est supposé être connu à l'avance. Cette pleine connaissance préalable de l'environnement pourrait être irréaliste dans de nombreux scénarios. Tandis que les algorithmes online ne tiennent pas en pleine connaissance préalable l'environnement à couvrir et utilisent les mesures de capteurs en temps réel pour balayer l'espace cible. Ces algorithmes online sont également

appelés algorithmes de couverture à base de capteurs.
Dans certains scénarios, une approche valable pour résoudre le problème de couverture complète est de le rendre aléatoire. Cette approche a connu certains avantages : pas de capteurs de localisation complexes ni de ressources informatiques coûteuses.
Cependant, pour couvrir une surface complexe, en particulier pour le nettoyage d'une chambre en utilisant un robot aspirateur autonome qui traite un espace plein d'obstacles, il est difficile de penser qu'un "algorithme" randomisé pourrait être utilisable, pour cela nous devons réfléchir à une approche ou à un algorithme très puissant pour planifier le chemin de ce robot aspirateur.
La plupart des algorithmes de planification de chemin de couverture décomposent l'espace cible en sous-régions (appelées cellules) pour obtenir une couverture, Choset dans [15] a classé les algorithmes de couverture selon le type de décomposition utilisé. Par conséquent, sa taxinomie comprend des approches heuristiques et randomisées, des décompositions approximatives, semi-approximatives et exactes des cellules. Cette taxinomie de Choset est couramment utilisée dans la littérature, dans le présent document nous fournissons également la classification Choset correspondant pour les méthodes examinées.

2.3 Décomposition cellulaire exacte

La méthode de décomposition cellulaire exacte divise l'espace libre (i.e, l'espace sans obstacles) vers le bas dans les régions simple, sans-chevauchement (appelées cellules). L'union de toutes ces cellules remplit exactement l'espace libre. Ces cellules, qui ne contiennent pas des obstacles, sont «faciles» à être couvertes et balayées par le robot à l'aide de simples mouvements. Par exemple, chaque cellule pourrait être couverte en utilisant un zigzag, [16–18]. et comme illustré dans la figure2.1.

Deux cellules sont dites adjacentes si elles partagent une frontière commune. Un graphe d'adjacence peut être utilisé pour représenter la décomposition cellulaire, où un nœud représente une cellule et un bord représente une relation d'adjacence entre deux cellules (voir 2.2). La décomposition cellulaire exacte peut être générée en balayant une ligne à travers l'espace (par exemple de gauche à droite). Les limites des cellules sont ensuite formées quand un événement est rencontré par la ligne de balayage. Par exemple, un changement du nombre de fois que la ligne de balayage intersecte les limites d'obstacles peut être utilisé comme un événement.
Typiquement, un planificateur basé sur la décomposition cellulaire exacte génère un chemin de couverture en deux étapes. Tout d'abord, il décompose l'espace libre en cellules et stocke cette décomposition comme un graphe d'adjacence. Ensuite, il calcule un chemin exhaustif à travers le graphe d'adjacence (i.e, une séquence qui visite chaque noeud dans le graphe exactement une fois). Il est à noter que le chemin exhaustif obtenu est une

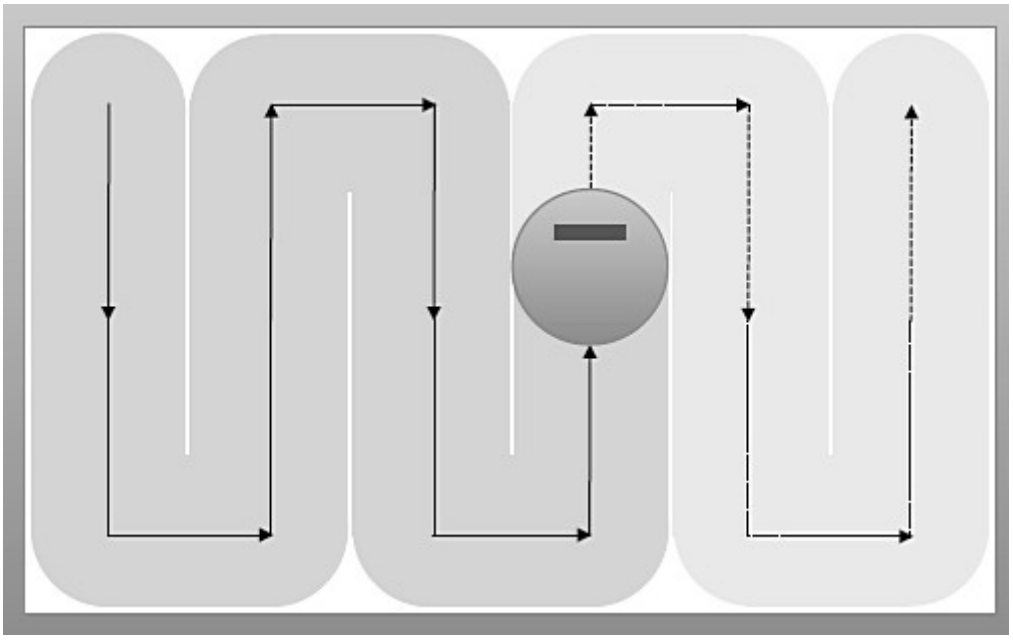


FIGURE *2.1* – "Tondeuse de la pelouse". La zone ombrée indique la zone déjà couverte (plus sombre) et la zone qui sera couverte (plus léger) lorsque le robot se termine en suivant le chemin.

séquence de noeuds (par exemple, une séquence de cellules), et non pas un chemin de couverture réelle. Par conséquent, un chemin explicite pour couvrir chaque cellule doit être calculé à l'aide de simples mouvements tel que expliqué ci-dessus.
Ensuite, Nous allons discuter sur deux approches populaires qui ont un but dans la décomposition cellulaires hors ligne d'un espace cible.

2.3.1 Décomposition trapézoïdale

Parmi les plus simples techniques d'une décomposition cellulaire exacte qui peuvent donner un chemin complet pour une couverture d'un espace est la technique de décomposition trapézoïdale [19, 20] qui ne gère que des espaces plats et polygonaux, étant donné qu'elle n'utilise pas des informations du capteur ; ce qui peut être considérée comme offline. Dans la décomposition trapézoïdale, chaque cellule est un trapèze, comme représenté sur la figure 2.2. Ainsi, de simples mouvements de va-et-vient peuvent être utilisés pour couvrir chaque cellule. Une couverture complète est garantie par la recherche d'un chemin exhaustif à travers le graphe d'adjacence associée à la décomposition (la couverture sur la décomposition de la figure 2.2). A la fin, un chemin en zigzag spécifique pour couvrir chaque cellule est généré. Le chemin exhaustif détermine l'ordre dans lequel les cellules sont visités pour obtenir une couverture complète : dans [21] les auteurs ont proposé un algorithme hors ligne basé sur la décomposition trapézoïdale pour la planification de chemin de couverture dans le cas de champs agricoles. Leur algorithme applique une décomposition trapézoïdale du champ suivi d'une procédure de fusion cellulaire. Les cellules

résultantes sont similaires à celles produites par la décomposition de boustrophédon, présenté dans la section suivante 2.3.2. Pour optimiser le chemin, ils utilisent une fonction basée sur le coût des chemins pour évaluer la plus grande cellule résultante à partir de six décompositions trapézoïdales différentes obtenues à partir de l'utilisation d'une ligne de balayage incliné à intervalles de 30°. Ensuite, les trois directions les plus favorables sont sélectionnés et le processus sera répété avec des décompositions supplémentaires à 15° de chaque côté des rubriques sélectionnées. Le processus se poursuit de façon itérative jusqu'à l'amélioration par l'étape de chute en dessous d'un seuil qui, pour son application a été achevée après 5 étapes (à environ 1° de précision), nécessitant 36 décompositions distinctes. Ensuite, la plus grande cellule de la décomposition, qui a un coût minimale, est retirée de la zone cible ; et le processus est répété pour le reste du domaine jusqu'à ce que toute la zone soit recouverte par le chemin. Ce système produit des chemins de couverture efficaces optimaux depuis un champ convexe et des chemins de haute qualité pour un champ dont les limites se composent de segments longs et droits aussi.

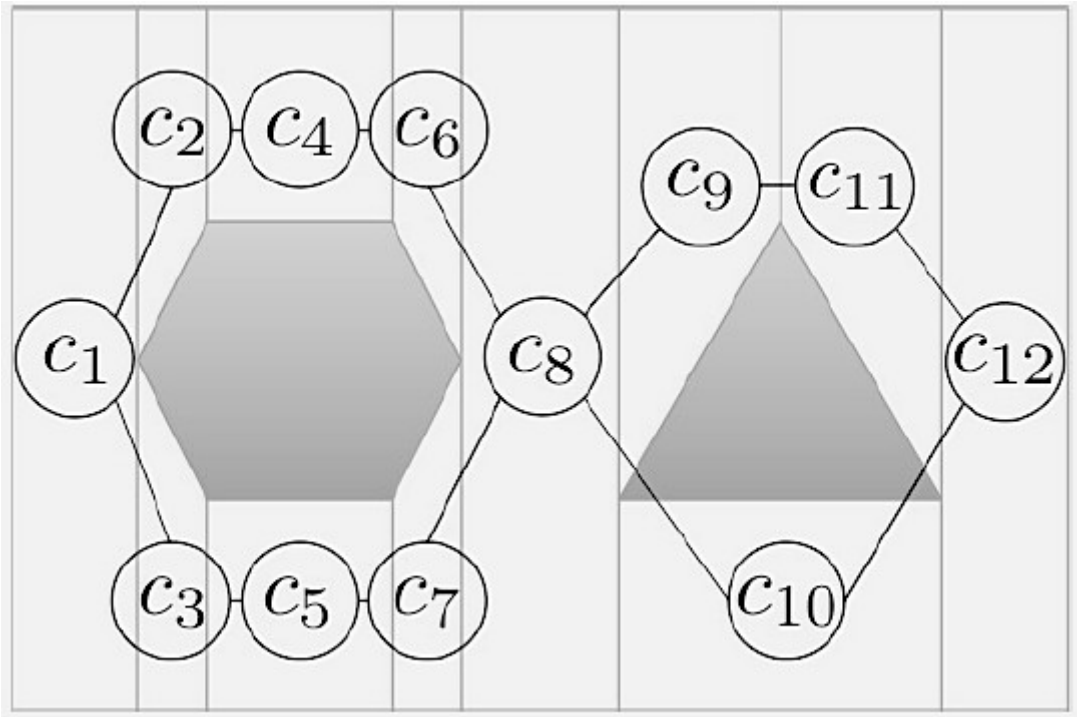


FIGURE *2.2* – Exemple de décomposition trapézoïdale d'un graphe d'adjacence correspondant.

2.3.2 Décomposition boustrophédonne

Un inconvénient de la décomposition trapézoïdale est qu'il génère de nombreuses cellules qui, intuitivement, peuvent être fusionnés ensemble pour former des cellules plus grandes. Ceci est clairement un inconvénient, car les cellules sont plus présentes ; de plus, la trajectoire finale de couverture devient longue, comme représenté sur la figure 2.3. Cela

arrive parce que la décomposition trapézoïdale ne crée que des cellules convexes. Cependant, certaines cellules non-convexes peuvent également être entièrement couvertes par de simples mouvements. Pour franchir cette limitation, Choset & Pignon ont proposé la décomposition boustrophédon cellulaire [17, 22]. Le mot «boustrophédon» vient de l'ancien grec et il signifie littéralement «la voie du bœuf", signifiant le modèle dans lequel un bœuf traîne une charrue en avant et en arrière. La décomposition boustrophédon est similaire à la décomposition trapézoïdale présenté ci-dessus, mais il ne considère que les sommets dans l'environnement dans lequel un segment vertical peut être étendue au-dessus et au-dessous du sommet. Les sommets où ils se produisent sont appelés points critiques.
En adhérant à cette stratégie, la décomposition boustrophédon réduit efficacement le nombre de cellules dans la décomposition trapézoïdale. Par conséquent, les plus courts chemins de couverture sont obtenus. A signaler que, comme la décomposition trapézoïdale, cette méthode suppose que les obstacles polygonaux et le terrain soient connus à priori, et donc classifié comme une méthode offline.

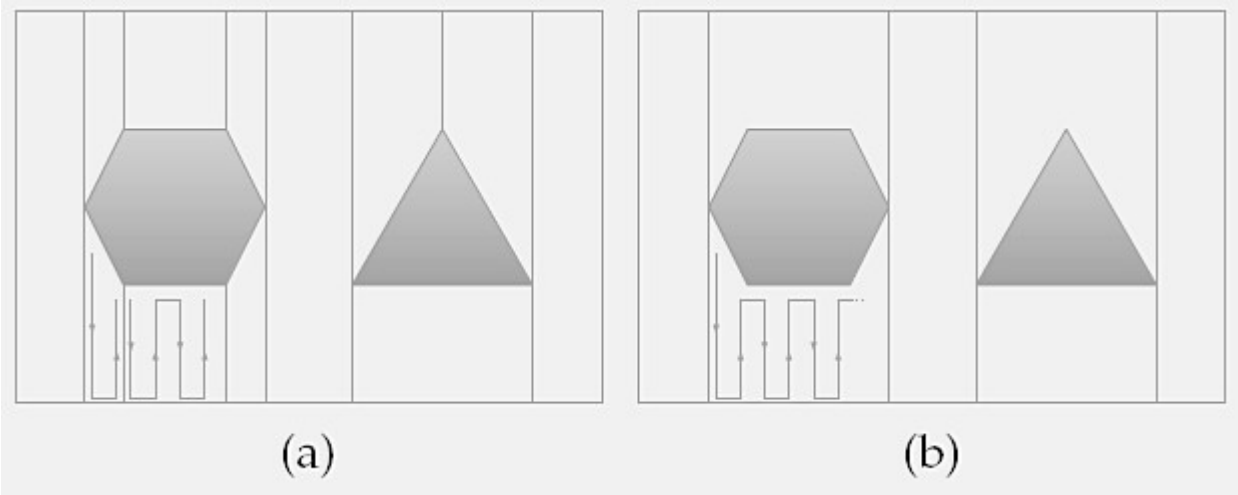


FIGURE *2.3* – Une décomposition avec moins de cellules. (a) Une bande supplémentaire est nécessaire dans la décomposition trapézoïdale (b) par rapport à la décomposition boustrophédonne.

2.4 Décomposition cellulaire à base de morse

Plus tard, dans [18], les auteurs ont généralisé la décomposition boustrophédon en proposant une nouvelle approche de décomposition cellulaire basée sur les points critiques de la fonction de Morse [23]. En effet, ils montrent que la décomposition boustrophédon est un cas particulier de la décomposition Morse. Avec l'observation de la décomposition originale de boustrophédon, la décomposition basée sur Morse a l'avantage de traiter également des obstacles non-polygonaux. Avec le choix de différentes fonctions de Morse, nous obtenons différentes formes de cellules, par exemple des cellules circulaires ou à

pointes. Théoriquement, les décompositions Morse peuvent être appliquées à tout espace à n dimensions. En outre, ils ont présenté une méthode pour effectuer une couverture des espaces plats par la détection des points critiques en utilisant les informations des capteurs , et des modèles de mouvement basés sur un algorithme qui assure la rencontre de tous les points critiques dans la zone-cible. Par conséquent, cette méthode permet une couverture complète d'un espace en ligne [24, 25].
La décomposition Morse est basée sur une méthode de feuille de route pour la planification d'un chemin depuis un point de départ vers l'arrivée proposée par Canny [26, 27]. Les points critiques d'une fonction Morse restreinte aux limites d'obstacle sont utilisés pour déterminer la décomposition cellulaire. Rappelons qu'étant donné une fonction à valeur réelle $h : \mathbb{R} \to \mathbb{R}$, son différentiable en $p \in \mathbb{R}^m$ est $dh(p) = [\frac{\delta h}{\delta x_1}(p)...\frac{\delta h}{\delta x_m}(p)]$. Un point critique est une valeur $p \in \mathbb{R}^m$ où la fonction n'est pas différentiable où tous ses dérivés partielles sont : σ, i.e $dh(p) = \frac{\delta h}{\delta x_1}(p) = ... = \frac{\delta h}{\delta x_m}(p) = \sigma$, et son Hessian $(\frac{\delta^2 h}{\delta x_i \delta x_j}(p))$ n'est pas singulière. Par exemple, dans le cas d'une fonction variable unique, un point critique correspond soit à un maximum local, soit à un minimum local ou à une inflexion. Une fonction de Morse est un des points critiques qui ne sont pas dégénérés [23]. En pratique, cela signifie que les points critiques sont isolés les uns des autres.
Pour déterminer la décomposition cellulaire, une tranche est balayée à travers l'espace cible. Formellement, la tranche est une codimension, un collecteur défini sous forme de pré-image d'une fonction réelle Morse, $h : W \to \mathbb{R}$ où W est l'espace de travail du robot ; c'est à dire l'espace à être couvert. Par exemple, dans le plan $W = \mathbb{R}^2$ en choisissant $h(x, y) = x$ fera la tranche être effectivement une ligne verticale. Des Changements sur la connectivité de la tranche se produisent aux points critiques de la fonction Morse restreinte aux limites d'obstacles. Pour parler plus simplement, à un point critique la ligne de balayage rencontre un obstacle dont la droite normale de la surface est perpendiculaire à la ligne de balayage, comme le montre la Figure 2.4. La théorie Morse garantit que, entre des points critiques, la connectivité de la tranche reste inchangée. Ainsi, aucun obstacles n'est situés entre des points critiques ; l'espace entre elles peut être recouvert facilement par de simples mouvements et des points critiques peuvent être utilisés pour déterminer les limites de la cellule.
Le choix de différentes fonctions de Morse produit différentes formes de tranche et donc différents modèles de décomposition cellulaire. Pour plus de simplicité, nous allons décrire la décomposition boustrophédon base-Morse [28], ce qui produit dans le plan. Plus loin, nous donnerons des exemples de différents modèles de décomposition obtenus par l'utilisation de différentes fonctions de Morse.

Dans la décomposition boustrophédon, une tranche verticale définie sous forme de fonction de Morse $h(x, y) = x$ est balayée de gauche à droite dans l'espace de travail, i.e,

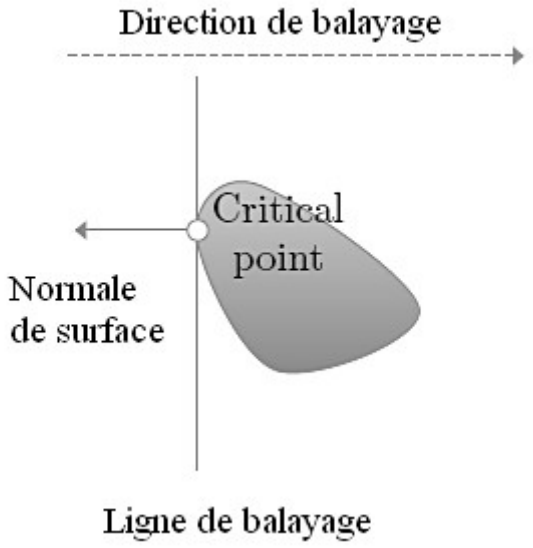


FIGURE *2.4* – Limites cellulaires de décomposition à base Morse sont placées à des points critiques, où la surface normale de l'obstacle est perpendiculaire à la tranche de balayage, et en parallèle à la direction de balayage.

le long de l'axe des abscisses. Ainsi, la tranche verticale est déterminée par la pré-image de cette fonction-Morse, $W_\lambda = h^{-1}(\lambda)$. La tranche est paramétrée par $\lambda \in \mathbb{R}$ qui fixe à son emplacement dans l'espace cible. L'augmentation de la valeur du paramètre de tranche, λ, balaye la tranche de gauche à droite à travers l'espace de travail.
Comme la tranche balaie l'espace, elle intersecte les obstacles qui la divisent en petits morceaux lorsque la tranche rencontre un obstacle ; c'est,la connectivité de la tranche dans l'espace libre qui augmente. Aussi, immédiatement après que la tranche laisse un obstacle, des petits morceaux de tranches sont fusionnés en gros morceaux (la connectivité de la tranche dans l'espace diminue). Les points où ces changements se produisent de connectivité sont les points critiques. (Rappelons que les points critiques sont toujours situés sur les frontières d'obstacles.) Ainsi, aux points critiques, la tranche est utilisée pour déterminer les cellules dans la décomposition. A noter que dans une cellule, la connectivité de la tranche reste constante. La figure 2.5.a montre comment, au point critique, la connectivité de la tranche varie d'un à deux, et par conséquent l'ancienne cellule est fermée et deux nouvelles cellules sont créées. Dans la figure 2.5.b, au point critique, la connectivité de la tranche passe de deux à un, et donc deux vieilles cellules sont fermées et une nouvelle cellule sera créée.

Une fois que la décomposition de la cellule est construite, un chemin exhaustif est déterminé par le planificateur à travers son graphe d'adjacence associé. Après, il génère le chemin de la couverture explicite dans chaque cellule. Le modèle de couverture dans chaque cellule comporte trois parties : le mouvement le long d'une tranche, le mouvement

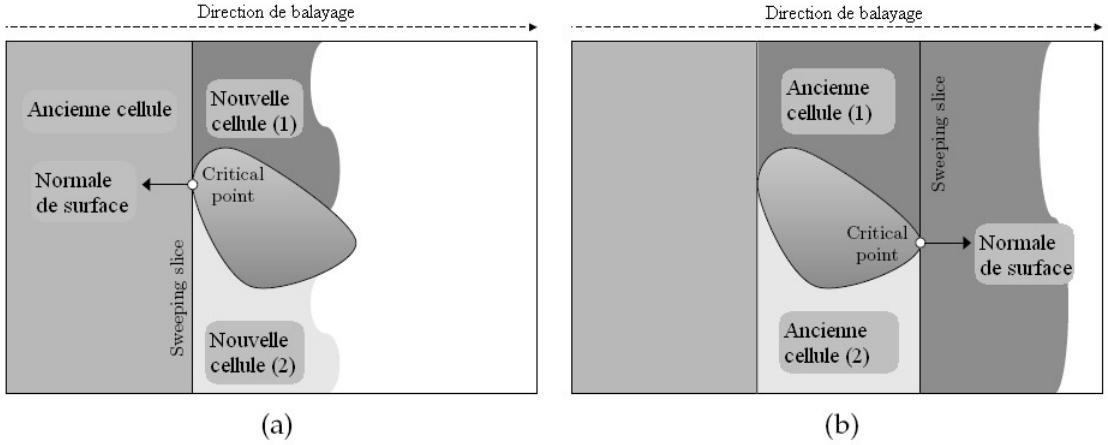


FIGURE *2.5* – Détermination cellulaire par la décomposition à base de Morse boustrophedonne.

perpendiculaire à la tranche et le mouvement le long de la frontière de la cellule, comme le montre la Figure 2.6. Tout d'abord, le robot tourne le long de la tranche W_λ. Ensuite, il se déplace à l'extérieur de la tranche en passant orthogonalement vers lui par une distance inter-tour, typiquement à une distance d'un rayon du capteur de robot ; λ est également augmentée par cette distance pour former une nouvelle tranche. Si le robot rencontre un obstacle (par exemple, la limite de la cellule) tout en se déplaçant le long de la tranche, le planificateur dirige le robot pour suivre la limite d'obstacle jusqu'à ce qu'il soit déplacé une distance de inter-tour ; puis, un nouveau tour le long d'une nouvelle tranche est commencé. Le processus se répète jusqu'à ce que la cellule soit complètement recouverte.

La figure 2.7 représente la décomposition cellulaire boustrophédon à base Morse d'un exemple espace de travail avec son graphe d'adjacence associé. Un point clé de décompositions Morse est que, en choisissant différentes fonctions de Morse pour définir la tranche qui est balayée à travers l'espace ; différents modèles de décomposition et de chemin de couverture peuvent être générés, tel que le motif en spirale [18]. La figure 2.8 montre un motif en spirale obtenue en utilisant la fonction-morse $h(x, y) = \sqrt{x^2 + y^2}$. Différents modèles de couverture est utile pour les véhicules avec des contraintes cinématiques. Par exemple, une trajectoire en spirale peut être facilement suivie par un véhicule de voiture-comme sous-actionné incapable de faire des virages serrés [29].
La méthode de décomposition-Morse a une limitation qu'empêche de gérer les environnements rectilignes. En effet, il est impossible de déterminer les points critiques dans ces environnements qui correspondent à un changement dans la topologie de l'espace (les points critiques sont dites dans ce cas dégénérés [23]).

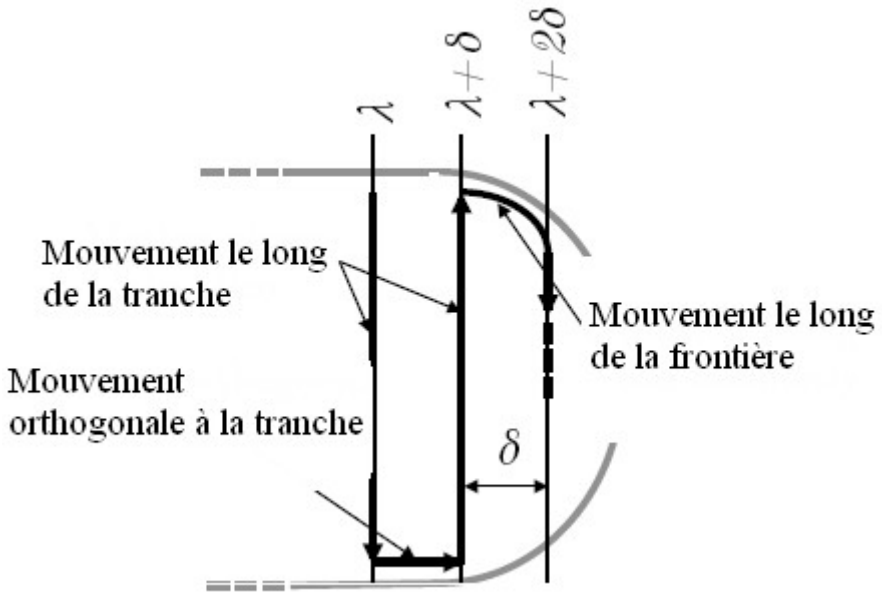


FIGURE *2.6* – Processus de Boustrophedon pour la construction du chemin, où δ est l'espacement inter-tour et λ est le paramètre de tranche.

2.4.1 Décomposition boustrophédonne basée Morse en ligne

Pour faire face au problème de la couverture à base de capteurs, les auteurs de [25, 30] ont donné une méthode pour détecter les points critiques d'une décomposition de boustrophédon base-Morse en ligne en utilisant les informations du capteur. En outre, ils ont présenté un algorithme qui assure de rencontrer tous les points critiques tout en effectuant la couverture. Pour détecter les points critiques, ils utilisent un capteur omnidirectionnel pour trouver les points où les droites normales de la surface $\nabla m(x)$ des obstacles et de la direction de balayage sont parallèles. Compte tenu d'un robot situé au point x, il faut laisser c_o être le point le plus proche de x sur la surface d'obstacle C_i :

$$c_o = argmin||x - c||, c \in Ci \tag{2.1}$$

Tout comme il faut laisser $d_i(x)$ être la distance entre le point x et l'obstacle C_i. Après cela, le gradient de $d_i(x)$, $\nabla d_i(x)$ peut être calculé comme :

$$\nabla d_i(x) = \frac{x - co}{||x - c_0||} \tag{2.2}$$

Rappelons que, par définition, un gradient est un vecteur unitaire d'une droite normal d'une surface à un moment donné. Dans l'équation 2, c_o est un point sur la surface de l'obstacle C_i, $x - c_o$ est un vecteur pointu vers l'extérieur à partir de c_o vers x. Étant donné que c_o est le point le plus proche de x sur la surface des obstacles, le vecteur $x - co$

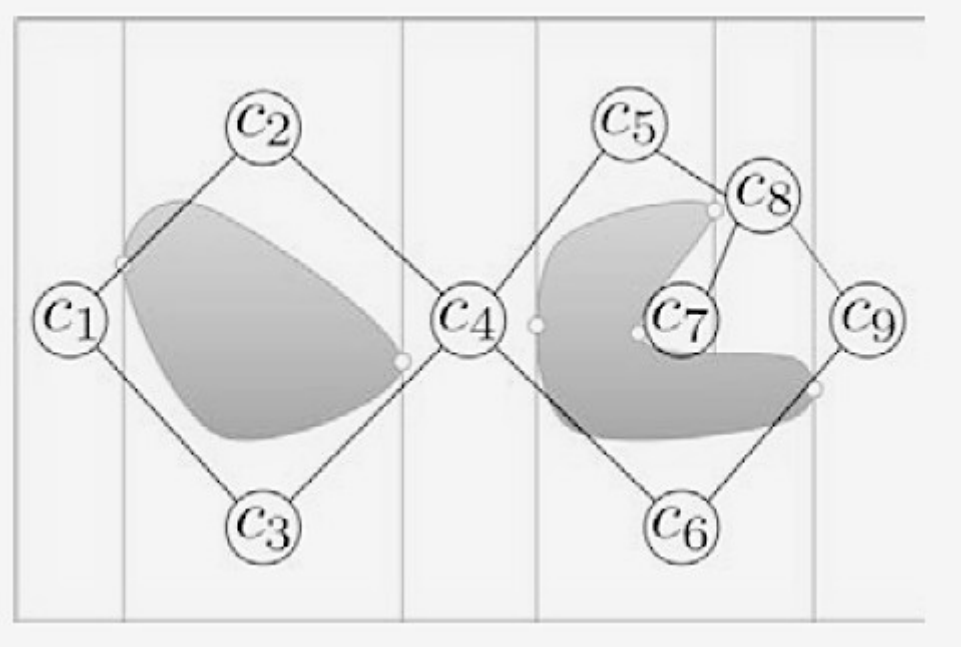


FIGURE *2.7* – Exemple de décomposition à base de Morse de graphe d'adjacence.

est donc perpendiculaire à la surface d'obstacle. En divisant par sa norme, $||x - c_o||$, le résultat sera transformé en un vecteur d'unité.
La détection d'un point critique se produit lorsque $\nabla d_i(x)$ est parallèle avec la direction de balayage ou, en d'autres termes, lorsque la direction de balayage et la droite normal de la surface d'obstacle sont parallèles. La figure 2.9 illustre cette situation.

Il faut remarquer que les points critiques ne peuvent être détectés lorsqu'ils sont proches du robot par rapport à tous les autres points de la surface d'obstacle. Cela implique qu'ils ne peuvent être détectés lorsque le robot effectue mur suivant. Par conséquent, l'utilisation d'un zigzag simple peut manquer certains points critiques, comme ceux de la figure 2.10.

Pour résoudre ce problème, dans [18] les auteurs ont présenté un algorithme qui utilise des cycles répétés de mouvement rectangulaire, avec mur suivant sur les deux extrémités d'un tour, comme le montre la figure 2.11. L'algorithme est appelé "algorithme de cycle». Il faut noter que cette trajectoire cyclique sera donc plus longue qu'une trajectoire en zigzag simple.

Le processus de leur algorithme proposé, pour la génération d'une trajectoire cyclables est illustré dans la figure 2.12. Initialement (figure 2.12.a), le robot commence à avancer à partir de point S_i et se déplace vers le bas. Quand il rencontre un obstacle, il effectue mur suivant jusqu'à ce qu'il atteigne la tranche suivante ou jusqu'à un point critique soit

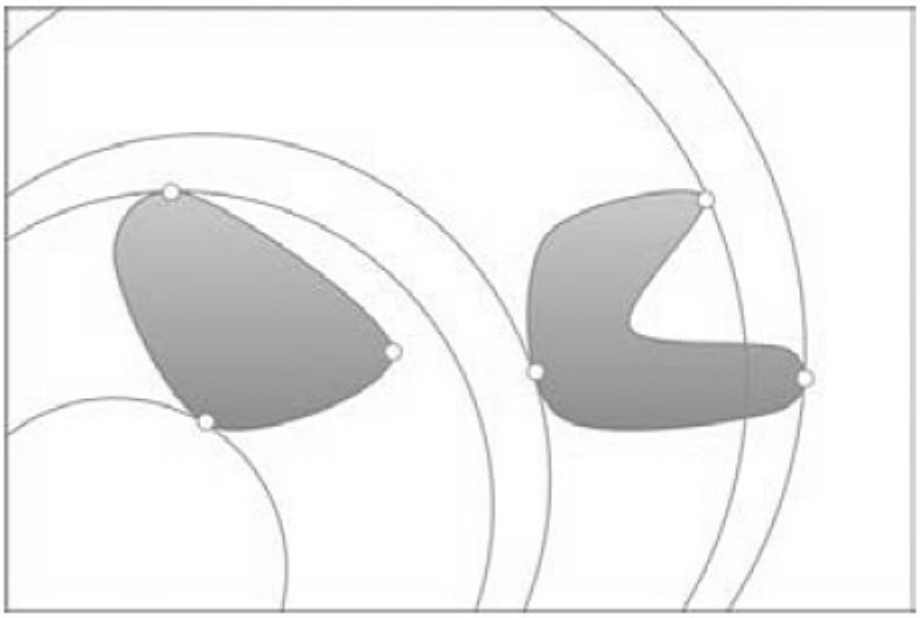


FIGURE *2.8* – Décomposition Spirale à base Morse obtenue en utilisant la fonction de Morse.

détectée. Si un point critique est détectée, le robot entre alors dans une phase inverse où il se déplace vers le haut (voir la figure 2.12.b). Dans cette phase, si un obstacle est détecté, le robot suit ce mur. Si un point critique est détecté sur la limite d'un obstacle, la tranche suivante est déplacée à l'endroit où le point critique a été détecté. Le mouvement se poursuit jusqu'à S_i est atteint à nouveau la tranche courante. Ensuite, le robot commence un nouveau cycle au point S_{i+1}. L'algorithme a prouvé la détection de tous les points critiques.

Pour stocker et construire progressivement la décomposition Morse en la ligne, il est stocké comme un graphe de Reeb [31]. Le graphe de Reeb est un dual du graphe d'adjacence en ce que les noeuds du graphe de Reeb sont les points critiques et les bords relient des points critiques voisins, correspondent à des cellules. Un exemple de décomposition Morse avec son graphe de Reeb associé est représenté sur la figure 2.13.

Chaque fois qu'un point critique est rencontré, le robot met à jour le graphe de Reeb. Quand une trajectoire cyclable où les points critiques ont été trouvés est terminée, le robot cherche des cellules non encore couvertes à partir de dernier point critique rencontré. Si le point critique est associé à deux cellules non encore couvertes, le robot prend l'une des cellules associées comme la prochaine cellule à couvrir. Si il n'y a pas de cellules non couvertes associés au dernier point critique rencontré, une recherche au profondeur est effectuée sur le graphe de Reeb. Pour se déplacer vers une cellule sélectionnée non encore couverte, le robot suit le graphe de Reeb et planifie un chemin qui passe à travers les cellules et les points critiques. Lorsqu'il ne reste pas des cellules non couvertes dans le graphe de Reeb, l'environnement sera complètement couvert.

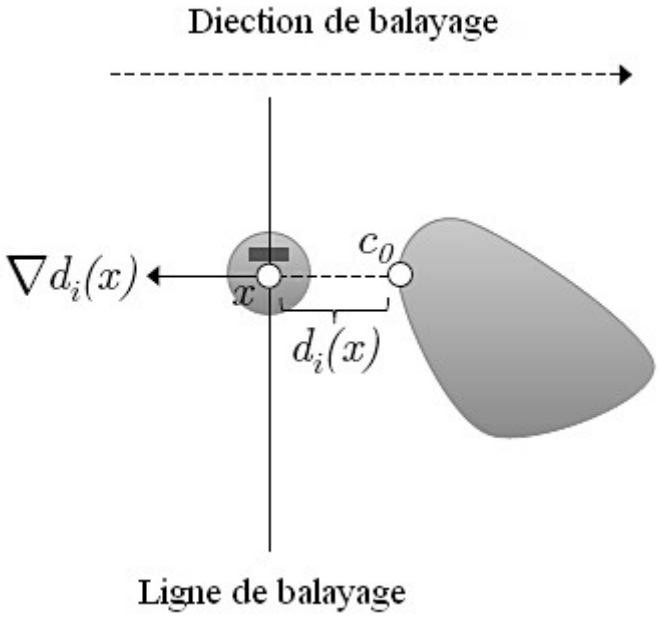


FIGURE *2.9* – Détection de point critique sur le rayon du robot.

L'algorithme de cycle qui a été décrit, peut ne pas détecter les points critiques sur certains obstacles non convexes. En particulier, les points critiques tels que Cp_2 concaves sur la figure 2.14 ne peuvent pas être détectés par des données d'intervalle lorsque la courbure de la limite est plus petite que la périphérie du robot [32]. Cependant, le point critique convexe le plus proche (Cp_3 dans l'exemple illustré à la figure 2.14) à un point critique comme Cp_2 dans la figure 2.14 sera effectivement détecté. Cela conduit à l'addition d'un faux bord du graphe de Reeb qui ne correspond pas à une cellule existante. Par conséquent, l'algorithme ne réussit pas à détecter le point critique pour la fermeture du bord nouvellement ajouté. Une solution à ce problème qui consiste à associer à chaque point critique avec son obstacle et définir une entrée et sortie unique des points critiques pour chaque obstacle est proposée dans [32]. En outre, ce papier aborde plusieurs détails de l'algorithme du cycle de mise en œuvre.

2.4.2 Décomposition cellulaire basée-Morse Combinée avec le diagramme de Voronoi généralisé

Les auteurs de [33,34] ont présenté une approche de couverture à base de capteurs avec extension. Ils soulignent que, « avant de faire la couverture à extension vous allez tomber dans l'un des deux extrêmes : la couverture avec un effecteur de la même taille du robot, et l'autre avec un effecteur qui a une portée infinie." ils considèrent les résultats de leur travail comme une couverture au milieu de ce spectre : la couverture avec une portée de détecteur qui va au-delà du robot, et pourtant qu'elle soit encore limitée dans cette portée. Ils

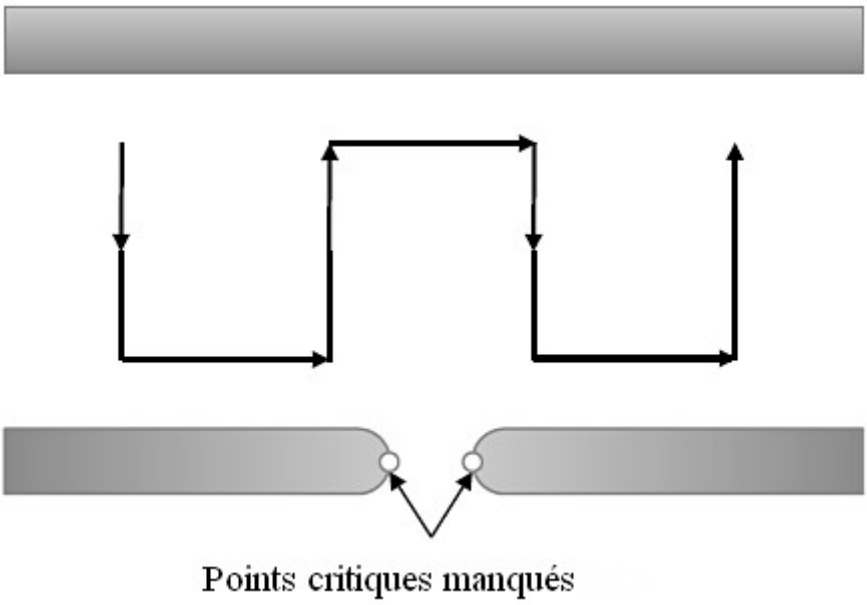


FIGURE *2.10* – Avec le décomposition à base Morse, une trajectoire en zigzag simple d'un robot, va manquer les points critiques dans la figure à moins qu'il effectue le mur suivant à la fois sur le haut et le bas d'une cellule.

appellent ces capteurs : des capteurs de portée étendue. Selon ses résultats, la couverture est réalisée en deux étapes : la première étape considère des espaces vastes et ouverts, où le robot peut utiliser toute la portée de son détecteur ; le robot couvre alors ces espaces comme si ils étaient aussi grands que sa portée de détecteur (voir Figure 2.15, à droite). Un travail précédent dans l'utilisation des décompositions cellulaires Morse [18] où ils ont utilisé pour couvrir les espaces inconnus. Comme expliqué ci-dessus, les cellules de cette décomposition peuvent être couvertes par de simples mouvements de va-et-vient, et la couverture du vaste espace se réduit pour assurer que le robot visite chaque cellule dans le vaste espace. La deuxième étape considère les espaces étroits ou encombrés où les obstacles se trouvent à portée de détection, et donc ; le détecteur remplit le pourtour de la région. Dans ce cas, le robot peut couvrir l'espace étroit en suivant simplement le diagramme de Voronoi généralisé (GVD) de cet espace qui sont des ensembles de points équidistants aux deux obstacles [35, 36](voir Figure 2.15, à gauche). Le GVD peut être construit en ligne à l'aide des informations de capteur et il a été déjà utilisé pour l'exploration à base de capteurs [35] et l'inspection des structures en 3D [37]. Une décomposition hiérarchique qui combine les décompositions Morse et les GVDs est introduite pour s'assurer que le robot en effet visite touts des espaces grands, ouverts, ainsi que étroits, encombrés. Dans leur résultats, ils montrent comment construire cette décomposition en ligne en utilisant les données des capteurs accumulées pendant que le robot couvre l'environnement.

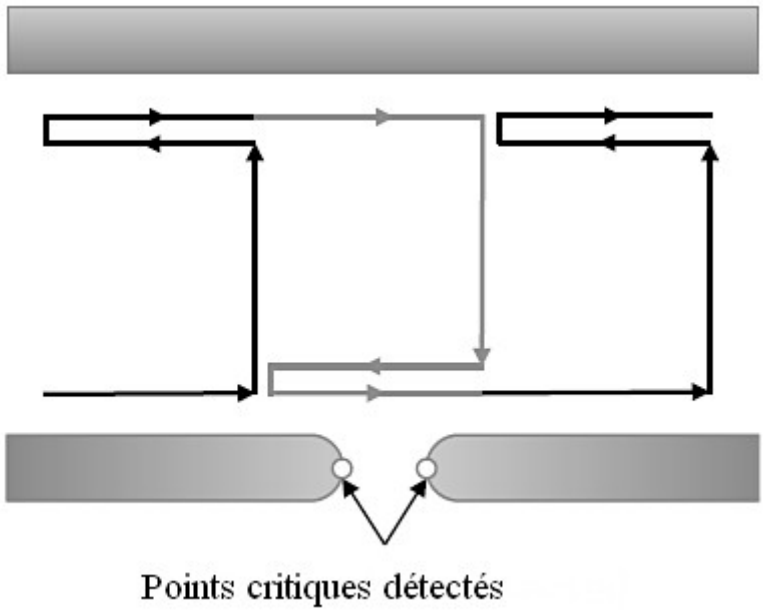


FIGURE *2.11* – Un chemin constitué de cycles rectangulaires permet la détection de tous les points critiques. Ce modèle est utilisé dans la décomposition à base Morse en ligne et l'algorithme de CCR.

2.5 Couverture topologique basée repère

Dans [38] Wong et al. ont présenté un algorithme de couverture topologique en ligne pour les robots mobiles basés sur la détection de repères naturels. Ce travail est destiné aux environnements de plans simples. Comme dans la décomposition Morse, leur méthode utilise également des concepts introduits par la décomposition de boustrophédon. Cependant, l'algorithme topologique proposé ici utilise différents événements pour déterminer les limites des cellules. La décomposition Morse place les limites d'une cellule sur les points critiques de la surface des obstacles. Cependant, comme commenté avant, les environnements rectilignes ne peuvent pas être manipulés par décomposition Morse car les points critiques dans de tels environnements sont dégénérés. D'autre part, comme les points critiques ne peuvent être découverts sur le côté du robot tout en effectuant mur suivant, un modèle de couverture rectangulaire, qui comprend une trajectoire est nécessaire. En revanche, l'approche topologique présentée ici utilise des repères simples pour déterminer une décomposition cellulaire exacte appelé «décomposition de tranche". Grâce à l'utilisation de points de repère simples, la décomposition de tranche peut traiter une plus grande variété d'environnements, y compris ceux avec des obstacles polygonaux, elliptiques et rectilignes. En outre, les obstacles peuvent être détectés de tous les côtés du robot, ce qui permet d'utiliser un modèle en zigzag simple sans revenir. En conséquence, le chemin de couverture généré est plus court avec cette méthode.

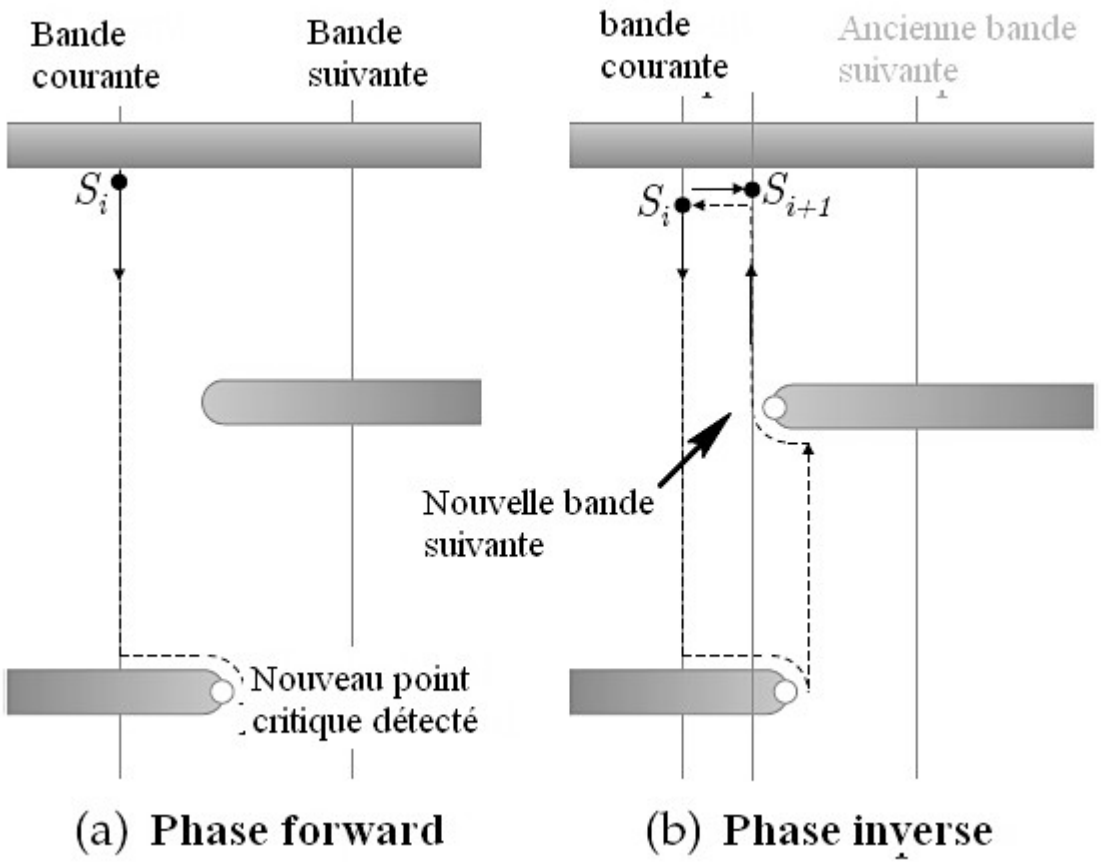


FIGURE *2.12* – Détection de point critique en utilisant l'algorithme de cycle.

2.5.1 Décomposition de Tranche

La décomposition de Tranche est construite par le balayage d'une ligne à travers l'espace. Elle utilise cinq événements différents pour déterminer les limites de la cellule :

1. *Événement de Division :* un segment libre d'un espace dans la bande précédente est divisé en deux par l'émergence d'un obstacle, comme illustré dans la figure 2.16.a
2. *Fusionnement :* deux segments libres d'un espace dans la bande précédente sont fusionnés en une seule par la disparition d'un obstacle, comme illustré dans la figure 2.16.b.
3. *Allongement :* la bande actuelle est beaucoup plus longue que la bande précédente, comme illustré dans la figure 2.16.c.
4. *Raccourci :* la bande actuelle est beaucoup plus courte que la bande précédente, comme illustré dans la figure 2.16.d.
5. *Fin :* le segment libre précédent d'un espace est le dernier dans la cellule courante, comme illustré dans la figure 2.16.e.

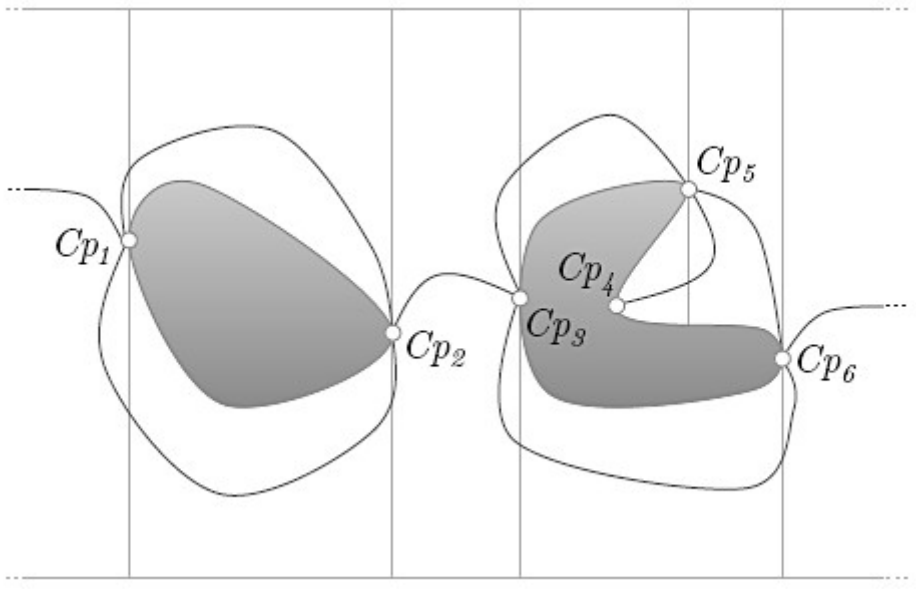


FIGURE *2.13* – Exemple de décomposition Morse avec le graphe de Reeb. $Cp_1 \ldots CP_6$ se sont les points critiques.

Tous ces événements ou ces repères peuvent être détectés en utilisant une combinaison de mesures de seuillage de portée et des comparaisons de séquences temporelles (comparaison entre la lecture du capteur en cours avec la précédente) et l'odométrie (en comparant la longueur des bandes consécutives). Une solution alternative qui utilise un réseau neuronal pour détecter les événements est également présentée dans [39].
Chaque fois que l'un des événements déterminé se produit, une limite de cellule est placée le long de la ligne de balayage où se déroule l'événement. La décomposition de tranche peut être codée comme une carte topologique. Une carte topologique est représentée comme un graphe planaire où les nœuds représentent des repères (par exemple : événement de division, fusionnement, fin, allongement ou raccourci) et les bords indiquent les types de mouvements nécessaires pour se déplacer entre les nœuds .Ils stockent également les distances estimées séparant les deux nœuds qui connectent.

2.5.2 Algorithme de couverture topologique en ligne

Un algorithme qui construit la décomposition de la tranche en ligne est présenté dans le travail de Wong et al. L'algorithme garantit une couverture complète. Il construit d'une façon itérative la carte topologique associée à la décomposition de la tranche pour l'environnement en utilisant une machine à états finis avec trois états : limite, normale, et déplacement. La figure 2.17 montre son diagramme de transition d'état. L'algorithme commence par l'état limite, tout comme on suppose que le robot est initialement situé dans un coin de l'environnement. Cette hypothèse n'est pas un défaut car il est facile de

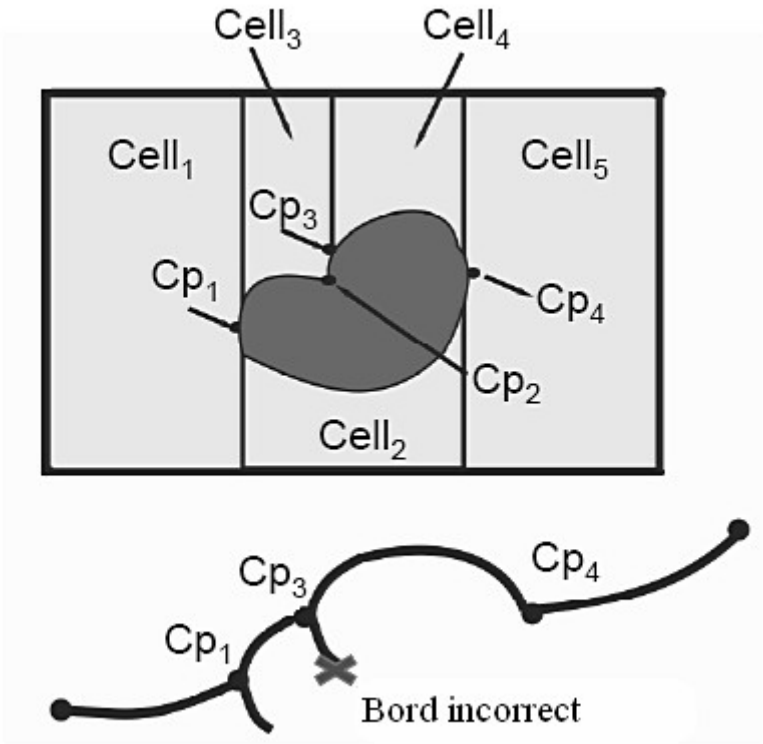


FIGURE *2.14* – Un point critique concave ne sera pas détecté si la courbure de la limite est inférieure à la périphérie du robot et conduira à un graphe de Reeb incohérent. Tel est le cas de cet exemple, Cp_2 dans l'environnement, ce qui ne se produit détecté et un bord incorrect émanant Cp_3.

programmer un robot pour trouver un coin en suivant de simples mouvements comme le mur suivant. Dans l'état-limite, le robot explore la limite actuelle de la cellule. Le but de l'exploration-frontière est d'exposer toutes les cellules voisines de la frontière actuelle. Chaque fois que le robot arrive à un point de repère ou à une extrémité de la frontière de la cellule, la carte topologique est mise à jour. Lorsque l'exploration limite est terminée, l'algorithme passe à l'état de déplacement. Dans l'état de déplacement, le robot recherche sur, la carte topologique, une cellule non encore couverte et il se dirige vers cette cellule. Ensuite, le robot entre dans l'état normal où il suit un modèle de zigzag afin de couvrir la cellule courante. De plus, chaque fois qu'un point de repère est trouvé, la carte topologique est mise à jour et l'algorithme passe à l'état limite. L'algorithme se termine lorsqu'il n'y a plus de cellules non couvertes dans la carte topologique.

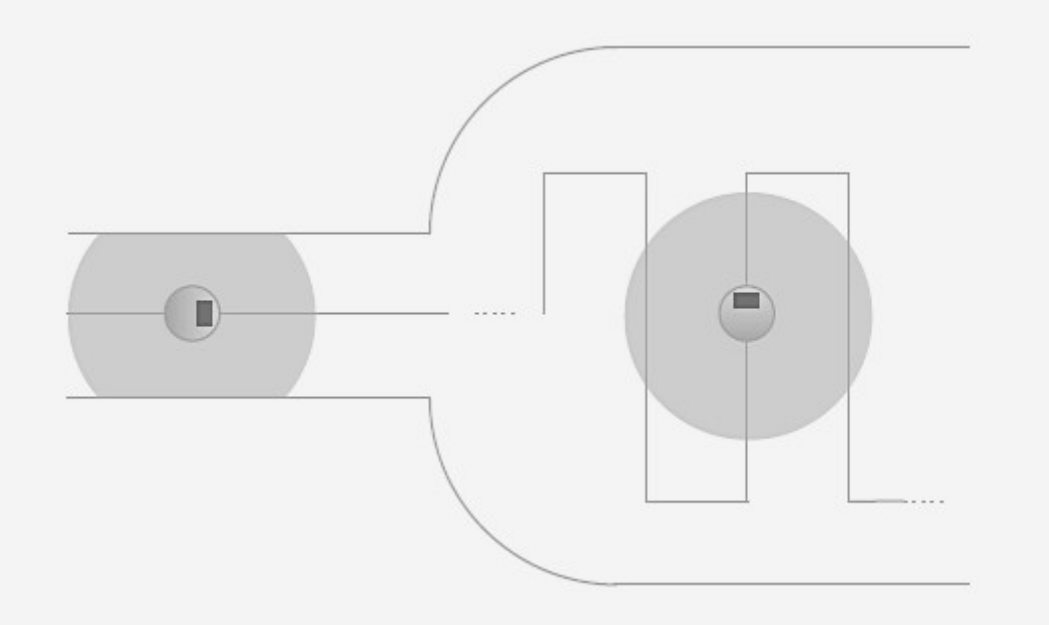


FIGURE *2.15* – Combinaison de décomposition Morse et GVD pour la couverture du capteur de rayon étendu.

2.6 Couverture des environnements rectilignes basée sur le contact du capteur (CC_R)

Dans [40] Butler et al. ont proposé (pour la couverture des environnements rectilignes basée sur le contact du capteur) CC_R, un algorithme de décomposition cellulaire exacte pour le contacte du capteur des robots (c à d- des robots sans portée des capteurs) couvre des environnements rectilignes inconnus en ligne. Son application travaille pour la couverture des environnements rectilignes sur un système de montage automatisé dans lequel le plan linéaire des moteurs fonctionne sur des surfaces de table dont le but de transférer les produits dans une usine.

Dans CC_R , le robot suit une trajectoire cyclique tracé comme représenté sur la Figure 2.11. En même temps, il construit d'une façon itérative une décomposition cellulaire de l'environnement. Un exemple d'une décomposition rectiligne produit par CC_R est représenté sur la figure 2.18. En effet, la décomposition construite par CC_R peut être considérée comme le cas de la décomposition Morse où tous les points critiques sont dégénérés, comme le cas des environnements rectilignes.

Normalement, CC_R suit le chemin cyclique. Un événement (et donc une limite de cellule) se produit chaque fois que le robot est empêché d'exécuter avec succès un cycle de chemin complet.Lorsque un tel événement se produit, CC_R choisit une nouvelle trajectoire basée uniquement sur l'environnement du robot et sa position actuelle. La prochaine trajectoire est déterminée par une liste de règles qui sont conçus pour continuer la couverture dans

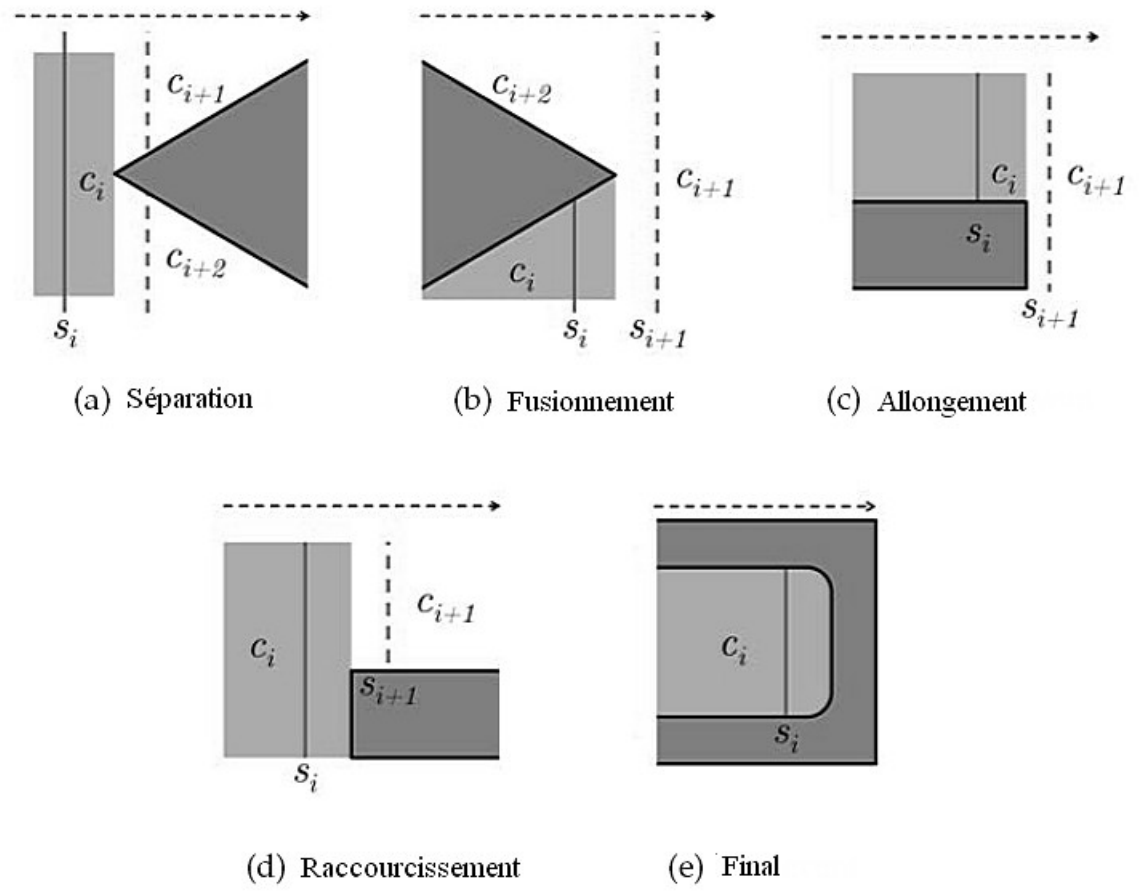


FIGURE *2.16* – Événements (Repères) dans la décomposition de la tranche.

tous les cas possibles.
Une preuve de complétude pour CC_R est donnée par la création d'une machine à états finis qui décrit tous les événements possibles rencontrés par le robot, et qui démontre que la machine à états finis n'a pas de boucles infinies et qu'il arrête lorsque la couverture est complète.

2.7 Méthodes basées-grille

Les méthodes par grille utilisent une représentation de l'environnement décomposée en un ensemble de cellules de grille uniformes. Cette représentation de la grille a été proposée dans [41] pour mapper un environnement intérieur à l'aide d'un anneau de sonar placé sur un robot mobile. Dans cette représentation, chaque cellule de la grille a une valeur associée indiquant si un obstacle est présent ou si elle a assez d'espace libre. La valeur peut être soit binaire ou une probabilité [42]. Typiquement, chaque cellule de la grille est un carré, mais également différentes formes de cellules de la grille peut être utilisé, tels que des triangles. Étant donné que les représentations de la grille ne se rapprochent pas

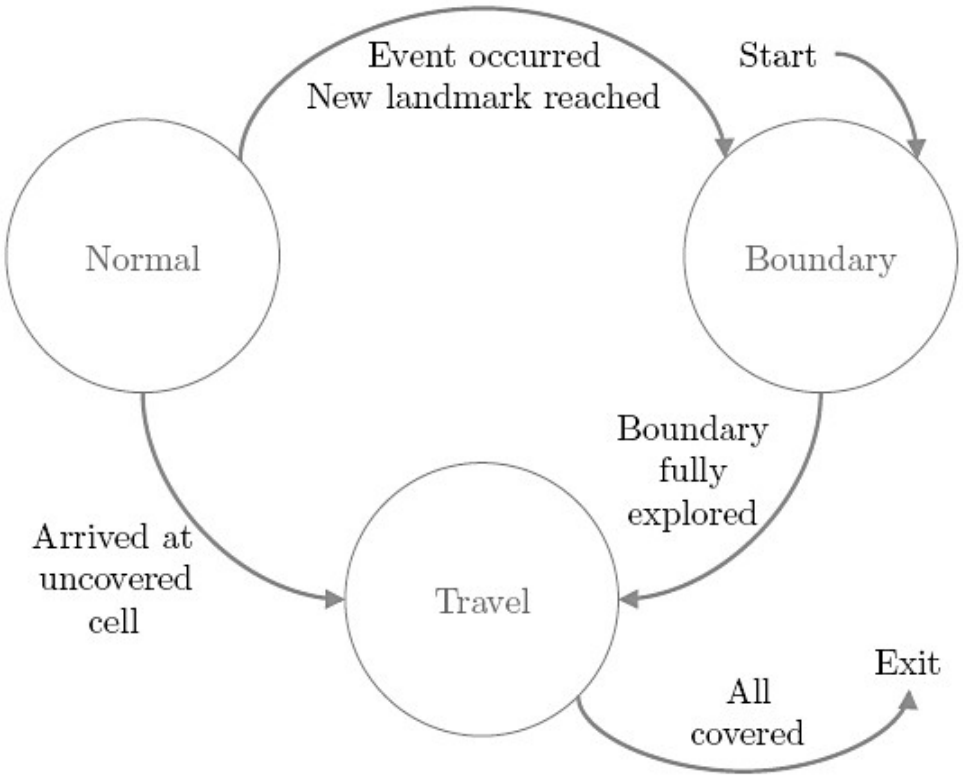


FIGURE *2.17* – État diagramme de transition de l'algorithme de couverture topologique.

de la forme de la région-cible et de ses obstacles, Choset a classé les méthodes basées sur une grille tels que des décompositions cellulaires approximatives [15]. Comme résultat de cette représentation approximative, la plupart des méthodes basées sur une grille ont une résolution-complète, dont, leur exhaustivité dépend de la résolution de la carte de la grille. La figure 2.19 montre un exemple de carte de grille.

Il est facile de créer une carte de grille, tout comme elle peut être représentée dans un tableau où chaque élément contient des informations d'occupation associé à une cellule. D'autre part, il est simple de marquer les zones couvertes dans une carte de grille. En conséquence, les représentations basées grille sont les plus largement utilisées pour les algorithmes de couverture. Néanmoins, les cartes de grille souffrent de la croissance exponentielle de l'utilisation de la mémoire parce que la résolution reste constante quelle que soit la complexité de l'environnement [43]. En outre, elles nécessitent une localisation précise pour maintenir la cohérence de la carte [44, 45]. Pour ces raisons, les méthodes de couverture basées sur une grille sont adaptées pour les opérations de robots mobiles d'intérieur, car la taille de la zone à couvrir est généralement relativement petite.
Habituellement, les cellules dans une carte de grille ont la même taille et la même forme carrée du robot . un algorithme de couverture est présenté dans [46] qui utilise une carte

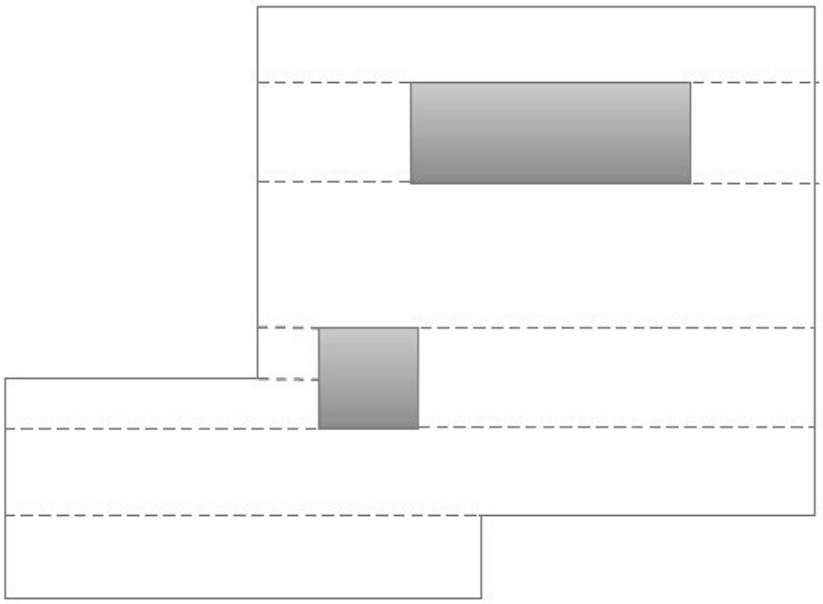


FIGURE *2.18* – CC_R utilise une décomposition cellulaire exacte pour les environnements rectilignes.

de grille dans laquelle les cellules sont des triangles. La raison derrière le choix des cellules triangulaires est qu'il offre une résolution plus élevée en comparaison avec les cellules rectangulaires de même taille. Toutefois, la résolution de la grille peut également être augmentée en utilisant des cellules carrées plus fine. Dans la robotique mobile, le domaine pour lequel l'algorithme mentionné est destiné, la plupart des robots mobiles ne sont pas capables de faire des ajustements de mouvement très fins ; et donc il n'y a pas besoin d'ultra-haute résolution dans la planification de chemin pour une couverture. Par conséquent, l'effort supplémentaire consacré à la mise en œuvre d'une grille triangulaire semble ne pas être valable.

2.7.1 Couverture basée-grille en utilisant l'algorithme de bloc d'ondes

Dans [47] Zelinsky et al ont présenté la première méthode basée sur les grilles pour la planification de chemin de couverture. Dans leur méthode hors ligne, ils utilisent une représentation grille et ils appliquent un algorithme complet de planification de chemin de couverture à la grille. La méthode nécessite une cellule de départ et une cellule d'arrivée. Une transformée de distance qui propage un bloc d'onde à partir de la cellule d'arrivée vers le départ, est utilisée pour attribuer un numéro spécifique à chaque élément de grille. Autrement dit, l'algorithme attribue un 0 à la cellule d'arrivée, puis un 1 à l'ensemble de ses voisins. Ensuite, toutes les cellules voisines de 1 non encore marquées seront marquées

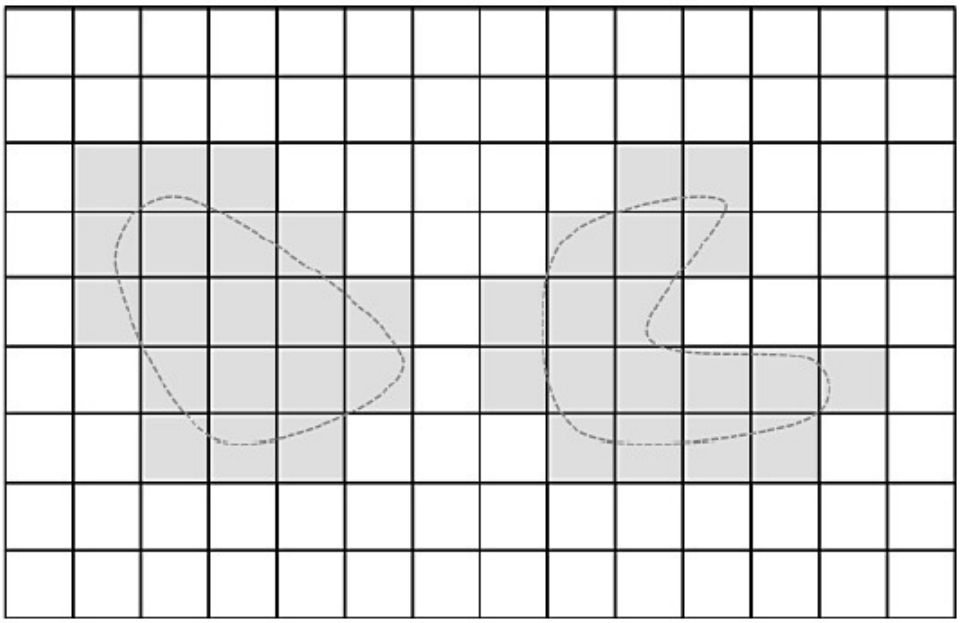
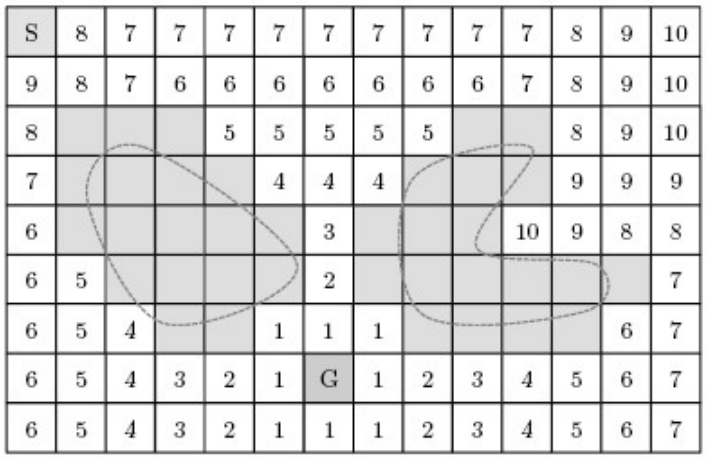


FIGURE *2.19* – Un exemple de carte de grille. les cellules de la grille avec des obstacles présents sont grisés.

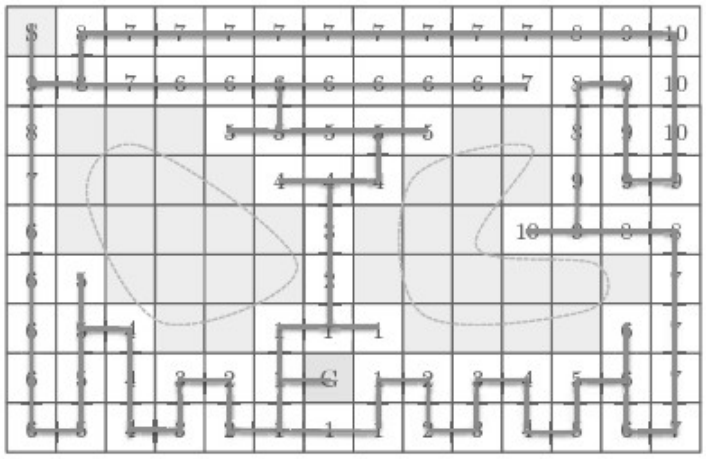
avec un 2. Le processus se répète incrémentiellement jusqu'à ce que la cellule de départ soit atteinte par le bloc d'onde. La figure 2.20.a illustre cette procédure sur un exemple d'environnement. Une fois que la transformée de distance est calculée, un chemin de couverture peut être trouvé à partir de la cellule de départ et la sélection de la cellule voisine qui a la plus haute étiquette non encore visitée. Si deux ou plus de cellules voisines non encore visitées partagent la même étiquette, l'une d'elles est sélectionnée de manière aléatoire. Ce processus de trouver un chemin de couverture est équivalent à l'utilisation de pseudo-descente de gradient à partir du point de départ sur la fonction de potentiel numérique, constitué par l'étiquetage, en suivant les courbes équipotentielles de haut en bas. La figure 2.20.b montre le parcours de la couverture générée pour un exemple d'environnement sur la figure 2.20.a. Une caractéristique unique de cet algorithme de couverture est qu'un point de départ et d'arrivé peuvent être spécifiés.
Dans [48] Shivashankar et al. ont introduit une généralisation de l'algorithme de bloc d'onde à des environnements inconnus pour atteindre la couverture en ligne avec un robot mobile.

2.7.2 Couverture basée-grille en utilisant la construction des arbres

Dans [49], Gabriely et al. ont proposé l'algorithme de couverture spirale par la construction des arbres "Spanning-Spiral tree" (STC), une approche en ligne pour les robots mobiles consiste à subdiviser l'espace de travail en une carte de grille et à suivre une tra-



(a) **Selection de point de départ (S) et point d'arrivée (G) par Transformé de disatnce de bloc d'onde**



(b) **Chemin de couverture généré par bloc d'ondes**

FIGURE *2.20* – Planification de chemin de couverture en utilisant l'algorithme de bloc d'ondes pour un exemple d'environnement.

jectoire en spirale systématique. Ce chemin spirale systématique est généré par le STC de la carte de grille partielle construite par le robot en utilisant ses capteurs embarqués. Le robot est capable de couvrir chaque cellule de la grille exactement une fois et de parcourir un chemin de couverture complète. Ils valident la proposition dans la simulation. L'algorithme Spiral-STC fonctionne comme suit : deux tailles de cellule de grille différentes sont utilisées. Les grandes cellules (appelées cellules méga) sont divisées en quatre cellules plus petites, qui sont de la même taille que le robot. Ceci est illustré sur la figure 2.21.a. Pour réaliser la couverture, le robot exécute la procédure suivante : à partir de la cellule courante, le robot choisit une nouvelle direction de parcourt en sélectionnant la première nouvelle méga cellule à partir de l'espace libre dans le sens inverse des aiguilles

d'une montre. Ensuite, un nouveau spanning-tree est déclenché à partir de la méga cellule actuelle vers la nouvelle. L'algorithme est appelé de manière récursive. La récursivité s'arrête uniquement lorsque la cellule actuelle n'a pas de nouveaux voisins (une méga cellule est considérée comme vieille, si au moins une de ses quatre petites cellules est couverte). A la suite de cette récursivité, le robot se déplace le long d'un côté de l'arbre de recouvrement jusqu'à ce qu'il atteigne l'extrémité de l'arbre. À ce point, le robot se retourne pour traverser l'autre côté de l'arbre. Il convient de noter que, lorsque la couverture est terminée, le robot retourne à la cellule de départ. D'autre part, la STC ne visite jamais toute petite cellule deux fois et minimise ainsi le temps moyen. La figure 2.21.b montre un exemple d'un chemin de couverture généré par l'algorithme Spiral-STC.
Une extension à la spirale-STC est l'algorithme Spiral Backtracking (BSA) [50], qui est aussi une approche en ligne destinée à des robots mobiles. Comme un avantage en ce qui concerne l'algorithme Spiral-STC, ils ont proposé une extension pour couvrir non seulement les cellules inoccupées, mais aussi les celles partiellement occupées. Cette extension est basée sur l'idée que les cellules partiellement occupées font partie de l'anneau externe de la trajectoire en spirale. Ces cellules sont couvertes par une procédure de mur suivant. L'extension proposée peut être appliquée à la plupart des algorithmes de couverture basés sur une grille. Des expérimentations de simulation valident l'algorithme proposé.
Dans [51] Choi et al. ont proposé une solution en ligne pour la planification de chemin de couverture complète sur la base des idées introduites par l'algorithme Spiral-STC et l'algorithme de BSA. Ils utilisent également des chemins spirales systématiques pour atteindre une couverture, sur la base de constatation murale active. Néanmoins, ils introduisent un système d'affectation carte de coordonnées basé sur l'historique des données de capteur pour améliorer le temps à la fin, et réduire le nombre de tours sur le chemin généré. Les chemins spirales générés sont ensuite reliés par une transformée de distance inverse qu'ils introduisent. Cette proposition est évaluée dans la simulation et aussi avec des expérimentations dans un monde réel avec un robot mobile.

2.7.3 Couverture basée-réseau de neurones sur cartes de grille

Les auteurs de [52,53] proposent une utilisation d'un réseau de neurones pour atteindre une planification de chemin pour une couverture en ligne destinée à des robots aspirateurs. Ils discrétisent un espace 2D en un plan de grille où la longueur de la diagonale de chaque cellule de la grille est égale au rayon de balayage du robot ; et ensuite ils associent un neurone à chaque cellule de la grille. Chaque neurone a des connexions avec ses 8 voisins directes. Ces concepts sont présentés sur la figure 2.22.
Une shunting équation basée sur l'équation de la membrane par Hodgkin et Huxley (1952) détermine la dynamique de chaque neurone dans le réseau. Le lansdscape de l'activité (par exemple : la valeur de sortie de tous les neurones à un instant donné) du modèle de shun-

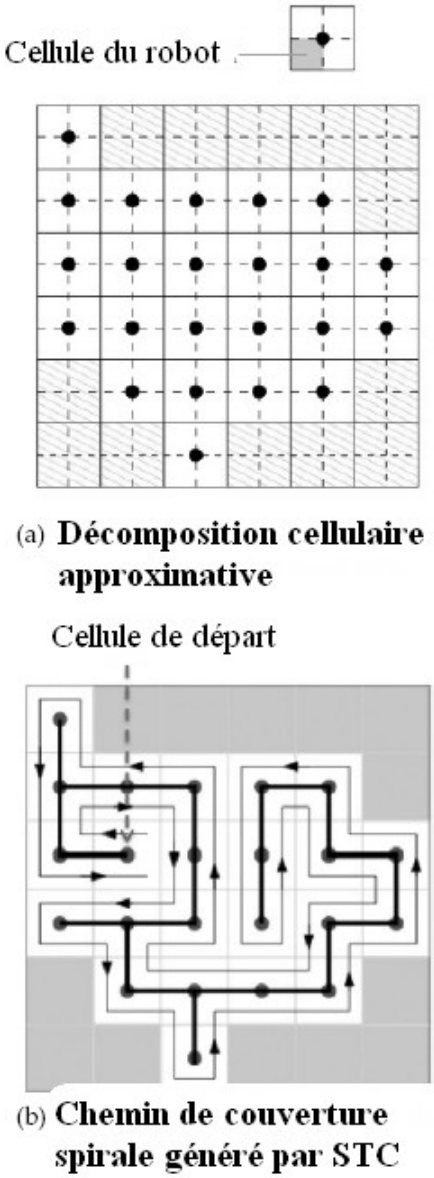


FIGURE *2.21* – Planification de chemin de couverture en utilisant l'algorithme Spiral-STC.

ting utilisé attire le robot aux zones non nettoyées, tandis que le robot est repoussé par les zones déjà nettoyées et les obstacles. La prochaine position du robot est déterminée par la position actuelle du robot et l'activité du neurone associé à sa position actuelle sans aucune connaissance préalable de l'environnement. Il est supposé que l'état actuel du robot (si il est dans une zone propre ou sale, ou devant un obstacle, et son emplacement) peut être déterminée par l'information du capteur. L'état du robot est une entrée pour le réseau neuronal. Le modèle utilisé a six paramètres qui peuvent être réglé dans un large intervalle de valeurs à la phase de conception du réseau de neurones, et donc la couverture est obtenue sans aucune procédure d'apprentissage. Un avantage de cette méthode est qu'elle peut gérer des environnements non stationnaires (i.e changement dynamique des obstacles). L'approche proposée de ce réseau neuronal est validée en simulation. Dans [53]

d'autres résultats de simulation sont présentés ainsi qu'une méthode pour effectuer la cartographie en ligne simultanément avec la navigation pour la couverture. Dans un autre travail, ils considèrent la carte typique basée sur une grille comme une représentation de maillage triangulaire de l'espace, tel que celui utilisé dans [46].
Une application de cette méthode basée sur le réseau de neurones pour un AUV qui couvre un espace de travail 2D dans le fond marin est proposée dans [54]. La proposition est validée dans un environnement de simulation simple. Néanmoins, cette approche souffre de problèmes d'évolutivité depuis la discrétisation en une carte en grille d'un vaste environnement tels que le fond marin qui présente un défi difficile à relever en termes de charge de calcul.
Dans [55] Qiu et al. ont ajouté une technique de planification de chemin local basée l'approche de réseau neuronal biologique inspiré discuté ci-dessus. Dans leur approche, la position suivante du robot ne dépend pas immédiatement mais plutôt une planification de trajectoire locale survient dans une fenêtre comprenant un voisinage déterminé du robot. Par l'utilisation de cette technique, ils réduisent la charge de calcul en comparaison avec l'approche de réseau de neurones toute seule.
Dans une approche similaire, dans [56] Guo et al. présentent une méthode basée sur un réseau de neurones pour générer la commande de direction continue pour un robot afin de couvrir complètement une région délimitée sur un temps fini. D'abord, ils discrétisent l'espace en une grille d'espacement régulier en forme de disque. Ensuite, un réseau neuronal basé sur la même équation de shuntage biologiquement inspiré comme dans les travaux décrits ci-dessus est utilisé pour fournir en continu de pilotage pour le robot. L'algorithme fonctionne pour les robots de voiture ont des contraintes de mouvement non holonomes. L'approche est validée dans la simulation.

2.7.4 Décomposition hexagonale de Grille pour les robots équipés de capteurs de vision latérale

Dans [57, 58] les auteurs ont présenté une méthode de couverture en ligne pour les robots équipés de capteurs latéraux prospectifs. Leur application est une opération MCM utilisant un AUV équipé d'un sonar à balayage latéral. Cette méthode de couverture dirige en permanence le cap du véhicule en utilisant l'optimisation multi-objectifs afin de maximiser le gain de l'information produite par les mesures des capteurs. La procédure d'optimisation utilise une décomposition de grille composée de cellules hexagonales uniformes. L'avantage d'utiliser une grille hexagonale est double : d'une part, la distance n'a pas besoin d'être prise en compte dans les fonctions objectives, parce que la distance à partir d'une cellule donnée à ses cellules voisines est la même. D'autre part, supposant que les hexagones sont assez petits pour visiter une cellule, cela garantis une couverture de deux cellules voisines par le capteur de vision latérale en minimisant la quantité de cellules

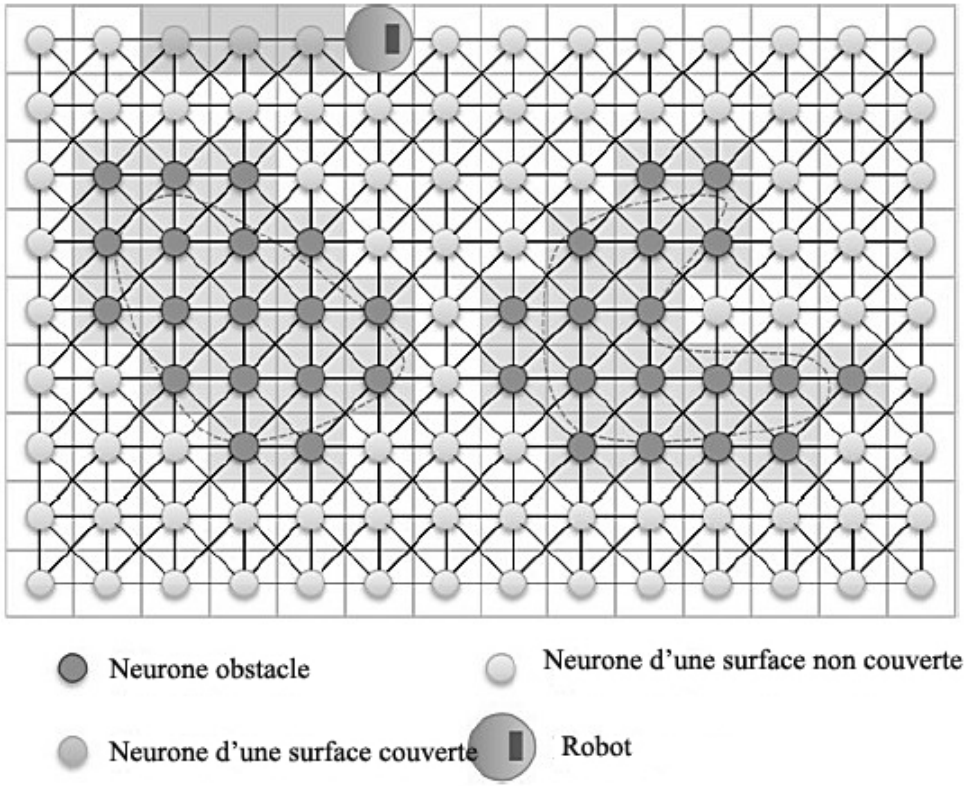


FIGURE *2.22* – Schéma du réseau de neurones utilisé par Luo, Yang et d'autres pour parvenir à une couverture.

partiellement couvertes. Bien que la méthode proposée est en mesure de couvrir les zones cibles avec des formes non convexes, les obstacles présents au milieu de la zone de travail ne sont pas considérés. L'efficacité de cette méthode est démontrée dans la simulation et l'expérimentation sur un AUV mené des opérations MCM.

2.8 Couverture basée-graphe

Dans [59] Xu et al. ont présenté des algorithmes de couverture pour les environnements qui peuvent être représentés sous forme de graphe, tel que un réseau de rue ou de route. En particulier, ce travail aborde les questions suivantes dans le problème de la couverture : premièrement, il prend en compte que les informations de la carte prioritaires qui sont fournies sous forme de graphe peuvent être incomplètes. Deuxièmement, il représente des contraintes d'environnement, telles que les restrictions dans certaines directions dans le graphe (correspondant à une rue à sens unique, par exemple). Troisièmement, il propose des stratégies pour une replanification en ligne lorsque des changements dans le graphe sont détectés par les capteurs du robot lors de l'exécution de couverture. Enfin, les stratégies de couverture qui utilisent plusieurs robots sont fournis.

2.9 Couverture optimale

Les travaux traitant de l'optimalité de la planification de chemin pour une couverture prévue, tels que la longueur du chemin et le temps d'achèvement, apparaissent dans la littérature de planification du chemin de couverture. Il faut noter qu'il y a une solution optimale pour un environnement à priori ou partiellement connu au moins depuis un exemple antagoniste qui peut toujours être trouvé par une approche à base de capteurs. Par conséquent, des méthodes de couverture optimale sont classés comme des méthodes hors ligne.

Dans [60] Huang a présenté une méthode optimale basée sur les lignes de balayage pour les algorithmes de décomposition dans des espaces cellulaires plats. Cette approche produit une longueur de chemin optimale pour une couverture en permettant différentes directions de balayage dans les chemins de tondeuses à gazon utilisés pour couvrir chaque cellule. L'idée principale est de minimiser le nombre de tours dans le chemin, car chaque tour implique généralement un coût supplémentaire du robot de décélération et d'accélération à nouveau après le virage. Ceci est réalisé pour faire différentes directions de balayage dans chaque cellule. Le nombre de tours est réduit au minimum par balayage dans chaque cellule parallèlement à son axe maximal de l'altitude. Autrement dit, la méthode vise à maximiser la longueur des tours dans le modèle en zigzag afin de minimiser le nombre de tours. Cependant, cette approche ne tient pas compte du coût de déplacement d'une cellule à une autre. La méthode est validée en simulation.

Dans [61, 62] Yakoubi et al. ont proposé d'utiliser un algorithme génétique pour obtenir une couverture optimale et aider un robot aspirateur de nettoyer toute une surface. Dans cette proposition, l'espace de travail et les obstacles sont supposés être circulaires et dynamique. Ensuite, l'espace libre est divisé en sous-régions en utilisant la méthode de décomposition cellulaire en grille. Tandis que, un autre algorithme génétique est utilisé pour planifier un chemin optimal qui couvre toutes les sous-régions dans [63] où l'espace de travail et les obstacles sont supposés être polygonaux et connus à l'avance. Ensuite, l'espace libre est divisé en sous-régions en utilisant la méthode de décomposition cellulaire trapézoïdale [19, 20]. Ces 2 propositions sont testées en simulation.

Un algorithme basé sur la décomposition cellulaire boustrophédon qui permet d'obtenir une couverture complète d'un espace connu, tout en minimisant la trajectoire du robot est proposé dans [64]. L'algorithme encode les cellules qui devraient être couvertes au bords du graphe de Reeb. Ensuite, la solution optimale au problème Postman chinoise est utilisée pour calculer une tournée Euler qui garantit une couverture complète de l'espace libre disponible tout en minimisant la longueur du chemin de couverture.

Dans [65] Xu et al. ont présenté une application de la méthode optimale de décomposition boustrophédon basée Morse pour les drones [64]. Tout d'abord, ils génèrent une trajectoire optimale exhaustive à travers le graphe d'adjacence de la décomposition des

cellules du terrain. Ensuite, ils couvrent chaque cellule avec des mouvements en zigzag en tenant compte des contraintes cinématiques du véhicule, comme le drone à voile fixe qu'ils utilisent a des contraintes non-holonomes. Les résultats expérimentaux étendus dans la simulation pour valider le système présenté, avec des données de plus de 100 kilomètres de vols de couverture ont été effectués à l'aide d'un aéronef à voilure fixe réel.

2.10 Couverture dans l'incertitude

Dans de nombreux scénarios, l'absence d'un système de localisation d'embarquement dans le robot tels que GPS entraine la dérive accumulée du robot ; ce qui engendre une incertitude croissante. Bien que les représentations topologiques tel que le graphe de contiguïté sont tolérantes à l'erreur de localisation, à la performance des algorithmes de couverture, même si l'utilisation de ces représentations, est affectée [15, 66]. En effet, le coût de la couverture à l'intérieur d'une cellule dépend de la direction du modèle en zigzag.

Les progrès récents dans le SLAM ont largement amélioré dans la localisation du robot. Le SLAM utilise des techniques statistiques pour corriger l'estimation de la position et l'orientation du robot. Cependant, le problème de la correction de la position du robot durant la couverture a été seulement abordée uniquement dans des petites recherches.

Dans [67] Acar et al. proposent de planifier les chemins de leur approche de décomposition Morse-basée-capteur en s'appuyant sur les limites de chaque cellule : donc dans le but de minimiser l'erreur de l'estimation.

Dans [68] Tully et al. ont utilisé une flotte de trois robots ; chacun d'eux est muni d'une bille rouge (facilement détectables en utilisant des techniques de vision par ordinateur standards) pour suivre une trajectoire en formation stratégique afin de minimiser l'erreur de localisation. Le chemin se compose d'une série d'étapes ou de sauts. Dans chaque saut, deux robots sont statiques et servent aux troisième comme des balises, ce qui constitue un progrès. Les robots échangent leurs rôles successivement. Des expériences réelles montrent une minimisation de l'erreur de localisation pour la couverture complète. Cependant, les obstacles ne sont pas pris en compte dans ce travail.

Dans [69] Kim propose une approche SLAM active au chemin de la couverture de la planification pour l'inspection de coque de navire dans un scénario 3D. L'algorithme proposé entraîne le robot le long d'une trajectoire de couverture pré-planifiées sur la coque du navire, et pendant l'exécution de la trajectoire, le robot sélectionne des emplacements de candidats qui, sont une fois revisités, et peuvent aider à réduire l'incertitude de position du robot. L'algorithme choisit de revisiter un emplacement de candidat une fois que l'incertitude de position dépasse un seuil fourni par l'utilisateur et suit le chemin pré-planifié autrement.

Dans [70] Bretl et al. suggèrent une façon pour planifier les chemins de couverture modifiés

pour un robot mobile dont la position et la vitesse sont sujettes à l'erreur bornée. Dans le pire des cas, en matière de modèle d'incertitude, les robots sont capables d'assurer une couverture complète. Cette garantie se fait au prix d'une trajectoire plus longue.

2.11 Discussion & Conclusion

Dans cette étude, nous avons ainsi vu que le problème de la planification du chemin de couverture a été traité à l'aide de différentes approches. Pour les espaces plates, la décomposition trapézoïdale garantit une couverture complète d'un environnement polygonal connu. Une amélioration de cette décomposition est la décomposition boustrophédon (classique) qui génère le plus court chemin de couverture complète pour la même classe d'environnement. La décomposition cellulaire basée-Morse fournit des chemins de couverture complète pour les environnements dont les limites des obstacles sont différentiables. La méthode pour détecter les points critiques détermine les limites des cellules en utilisant l'information du capteur qui permet de réaliser une couverture de décomposition cellulaire à base Morse en ligne. En outre, la décomposition Morse permet de générer différents modèles de couverture, tels que des modèles en spirale, qui peuvent simplifier le chemin suivant pour les véhicules avec des contraintes de mouvement.
Toutefois, la méthode de décomposition cellulaire à base-Morse ne gère pas les environnements rectilignes, ni les trajectoires rectangulaires cycliques utilisées pour détecter tous les points critiques ; ce qui les rend plus long qu'un chemin simple en zigzag. Ces limitations sont surmontées par l'approche de couverture topologique base historique. Cette méthode utilise une décomposition sur la base de points de repère cellulaire naturelles de l'environnement qui déterminent les limites des cellules. Un algorithme est donné pour effectuer la couverture en ligne sur des environnements inconnus en utilisant cette technique.
Pour le cas particulier des robots avec seulement des capteurs de contact (par exemple, sans capacités de rayon de détection) fonctionnant dans des environnements rectilignes, l'algorithme de CC_R garantit une couverture complète en ligne.
Les méthodes basées sur la grille, tels que l'algorithme de bloc d'onde, l'algorithme Spiral-STC et ses dérivés, le réseau neural et les approches de décomposition hexagonales, fournissent une couverture complète d'une représentation discrète de l'environnement cible. Cependant, la représentation de la grille utilisée d'un environnement est très sensible à l'erreur de localisation et entraîne une consommation de mémoire exponentielle. D'autre part, il est facile de créer une carte de réseau et de la faire fonctionner. Il est intéressant de remarquer, comme une capacité unique parmi les méthodes examinées, que les méthodes basées sur réseau neuronal discutées sont capables de gérer des environnements avec des obstacles mobiles.
Dans [59] Xu fournit des algorithmes de couverture pour les environnements qui peuvent

être représentés sous forme de graphe, comme un réseau de rue ou de route.
Nous avons discuté de plusieurs approches pour générer des chemins de couverture optimale dans les espaces plates et de réduire les erreurs de localisation tout en effectuant la couverture.
Les algorithmes à base d'échantillonnage probabiliste ont révolutionné dans les dernières années l'état de l'art dans la planification du chemin , et ils se sont avérés être extrêmement puissants comme démontré dans un ouvrage [69]. sur un navire (inspection de la coque). Par conséquent, l'exploitation de ces techniques ouvre la porte au développement d' algorithmes capables de réaliser des tâches de couverture de complexité sans précédent.
D'autre part, dans les applications du monde réel, un robot n'a souvent pas connaissance parfaite sur sa localisation ni de son environnement. Bien que plusieurs travaux de recherche ont exploré ce sujet en prenant en compte l'incertitude dans les problèmes de planification du chemin pour la couverture complète d'un environnement d'une façon optimale. Par conséquent, cela reste un sujet ouvert important pour la poursuite des recherches.

Chapitre 3
PCNN : Un réseau neuronal à impulsions-couplées

3.1 Introduction

Dans ce chapitre, nous allons décrire le modèle de réseau neuronal à impulsions-couplées (PCNN) qui constitue notre propre approche et notre contribution qui sera développée à travers les *chapitres* 4 et 5. A la suite de quoi, nous allons traiter la présentation de ce modèle (PCNN) et ses domaines d'applications.

Le PCNN est inspiré d'un modèle de neurone biologique Eckhorn et al [71] qui ont découvert que le mésencéphale d'une manière oscillante a créé des images binaires qui pourrait extraire des caractéristiques différentes de l'impression visuelle quand ils ont étudié le cortex visuel du chat [72, 73]. Grâce à cette découverte, ils ont développé un réseau de neurones, appelé le modèle de Eckhorn, pour simuler ce comportement.
Dans les années 1990, Rybak et al. ont également trouvé le comportement neural similaire basé sur l'étude du cortex visuel du cochon de Guinée et ont développé aussi un réseau de neurones, appelé modèle Rybak [74, 75]. Parce que ce modèle a fourni un moyen simple et efficace pour étudier la dynamique des impulsions synchrones dans les réseaux, ils ont été reconnus comme étant très potentiel dans le traitement d'image [76].

Ces découvertes décrites précédemment ont ouvert la voie à la génération de réseau neuronal à impulsions- couplées. Ensuite, dans [77] Johnson et al. ont porté un certain nombre de modifications et de variations pour adapter ses performances tels que des algorithmes de traitement d'image. Ce modèle neuronal modifié est appelé *réseau neuronal* à $impulsions - couplées$ $(PCNN)$.

Avant d'expliquer quelques applications de ce modèle, nous allons rappeler quelques notions concernant le reseau PCNN.

3.2 Architecture du réseau

Le PCNN est un réseau neuronal qui a été construit par la simulation de l'activité des neurones du cortex visuel d'un mammifère ; la structure de base du modèle PCNN est représentée dans la figure3.1.

Le PCNN produit une sortie d'images d'impulsions binaires lorsqu'il est stimulé avec des images. [78] a énuméré l'origine du PCNN(modèle de base et la relation aux modèles biologiques dans leur grande recherche sur le PCNN). Le modèle compartimental biologique est représenté sur la figure3.2.

Le circuit équivalent du modèle respectif est représenté sur la figure3.3 où $Y(t)$ représente le générateur d'impulsions : E_m est le potentiel de repos à l'intérieur de la cel-

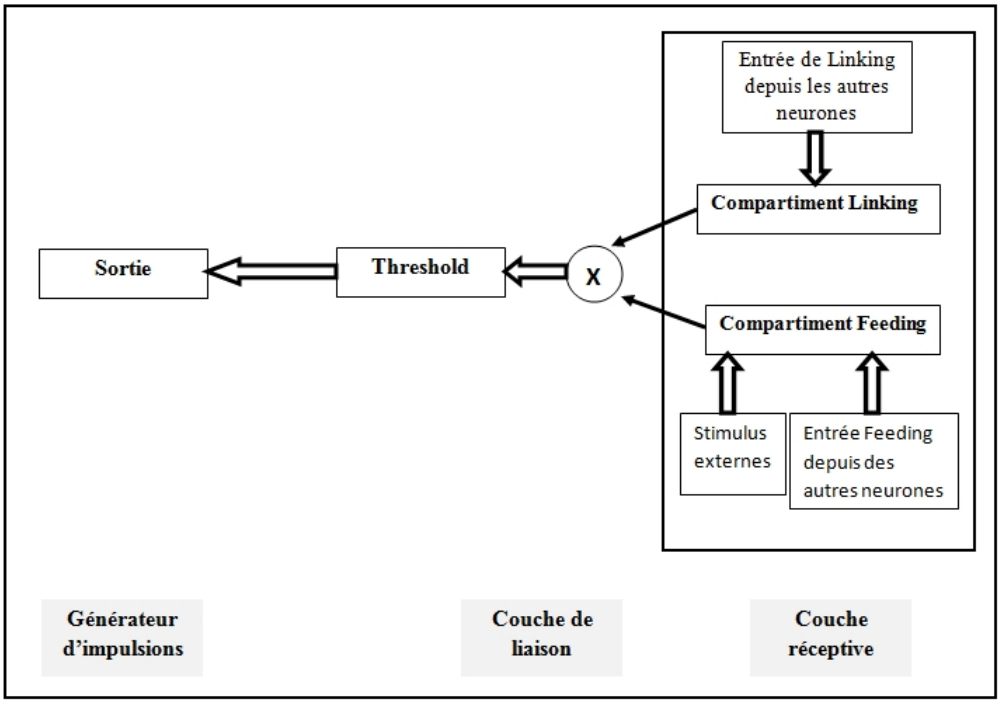


FIGURE *3.1* – structure de base du modèle PCNN

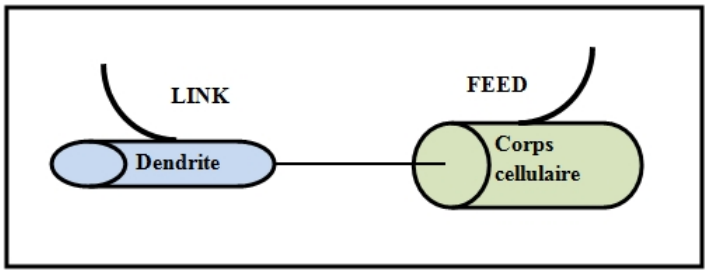


FIGURE *3.2* – Modèle compartmental biologique.

lule $(-70mV)$; E_s est le retour potentiel synaptique $(+20mV)$; C_1, C_2 sont les capacités compartmentales (ordre de nanofarads) ; g_{m1}, g_{m2} sont les conductances membranaires intrinsèques de fuite ; g_{12} est la conductance longitudinale ; V_1 et V_2 sont les tensions à compartiments.

Un réseau neuronal à impulsions-couplées est un réseau bidimensionnel constitué de 03 parties :

— Une couche d'entrée ou réceptive.

— Une couche de liaison.

— Un générateur d'impulsions.

La couche d'entrée ou réceptive est la partie primaire pour recevoir des signaux d'entré venant des neurones voisins et de sources externes, cette partie contient deux canaux internes appelés compartiment Feeding F et compartiment Linking L. Les entrées de

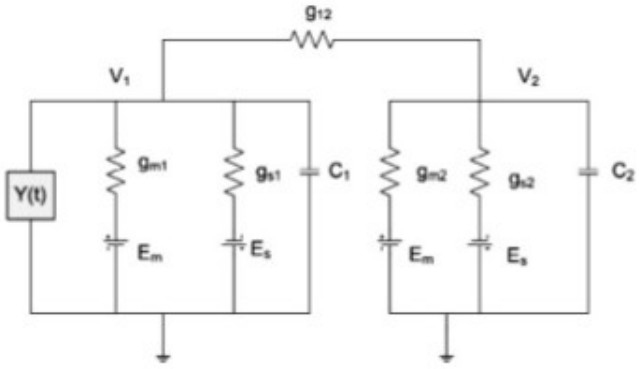


FIGURE *3.3* – Circuit équivalent du modèle compartmental.

Linking ont un temps de réponse rapide constant par rapport aux connexions de Feeding. Les entrées de Linking biaisées et multipliées sont multipliées avec l'entrée Feeding pour produire de l'activité interne totale U qui constitue la partie liaison ou modulation.
Enfin, le générateur d'impulsions du neurone est constitué d'un générateur de fonction en étape et un générateur de signal de seuil. Les neurones dans le réseau ont la capacité de répondre aux stimuli. Cette réponse est connue comme une activité. Cette activité se déclenche lorsque l'activité interne du neurone dépasse un certain seuil. La sortie de neurone Y devient 1. Maintenant, le seuil commence à diminuer jusqu'à la prochaine activation interne du neurone. La sortie du neurone est ensuite réinjectée d'une manière itérative avec un retard d'une seule itération. La sortie du neurone est donc remise à zéro (0) lorsque le seuil est plus grand que l'activité des neurones interne U. Le PCNN produit une série temporelle de sorties d'impulsions après n nombre d'itérations. La sortie d'impulsion carie les informations sur les données d'entrée. La décision sur le contenu des données est obtenue en examinant la sortie d'impulsions du réseau (PCNN).
L'entrée Feeding et Linking communiquent avec les neurones voisins à travers les poids synaptiques M et W. L'entrée de stimuli est donnée seulement dans le compartiment de Feeding. Le neurone reçoit l'entrée de stimuli S qui correspond à son information avec le stimuli des voisins dans les deux compartiments.

3.3 Modèle mathématique d'un PCNN

La figue.3.4 représente le modèle mathématique de PCNN tandis que la sortie des compartiments est déterminée par les équations suivantes :

$$F_{ij}[n] = e^{\alpha_F \delta_n} F_{ij}[n-1] + S_{ij} + V_F \sum_{kl} M_{ijkl} Y_{kl}[n-1] \tag{3.1}$$

$$L_{ij}[n] = e^{\alpha_L \delta_n} L_{ij}[n-1] + V_L \sum_{kl} W_{ijkl} Y_{kl}[n-1] \tag{3.2}$$

$F_{ij}[n]$: rétroaction du neurone dans la position (i, j) ;
$L_{ij}[n]$: point de liaison ;
S_{ij} : Entrée de stimuli à la position(i, j) ;
Y_{kl} : Sortie de neurones à partir d'une itération précédente $[n-1]$;
β : Coefficient de Liaison ;
M_{ijkl} et W_{ijkl} : Matrices synaptiques constantes de poids ;
V_F et V_L : potentiel inhérent de tension.

Les états de ces deux compartiments sont combinés pour créer l'état interne du neurone, et son activité $U_{ij}[n]$ qui est calculée suivant cette équation :

$$U_{ij}[n] = F_{ij}[n][1 + \beta L_{ij}[n]] \tag{3.3}$$

L'état interne du neurone est comparée à un seuil. h_{ij} pour produire la sortie de pulsation Y :

$$Y = \begin{cases} 1, & if U_{ij}[n] > \theta_{ij}[n] \\ 0, & ailleurs \end{cases} \tag{3.4}$$

Lorsque le neurone s'active ($Y > h$), le seuil augmente considérablement sa valeur. Cette valeur du seuil décroît ensuite jusqu'à ce que le nouveau neurone s'active. Donc, le seuil est dynamique. Le procédé est décrit par :

$$\theta_{ij}[n] = e^{\alpha_\theta \delta_n} \theta_{ij}[n-1] + V_\theta Y_{ij}[n] \tag{3.5}$$

où V_θ est une grande constante supérieure à la valeur moyenne de U_{ij}. Sur la base de l'application et des exigences, des modifications sont apportées au PCNN original et les modifications apportées sont discutées dans la section suivante.

3.4 Modifications au niveau du modèle PCNN

Pour réduire la complexité du réseau (PCNN) et augmenter la vitesse de calcul, certaines modifications ont été apportées dans le réseau. Ces modifications ont certes amélioré la performance globale du PCNN. Une version simplifiée du modèle PCNN est le modèle du cortex coupant (ICM). Un système a été développé en utilisant un minimum de deux oscillateurs couplés (fonction non linéaire et un petit nombre de connexions).
Les équations qui correspondent au réseau (ICM) sont illustrées suivant ces équations :

$$F_{ij}[n+1] = fF_{ij}[n] + S_{ij} + WY_{ij} \tag{3.6}$$

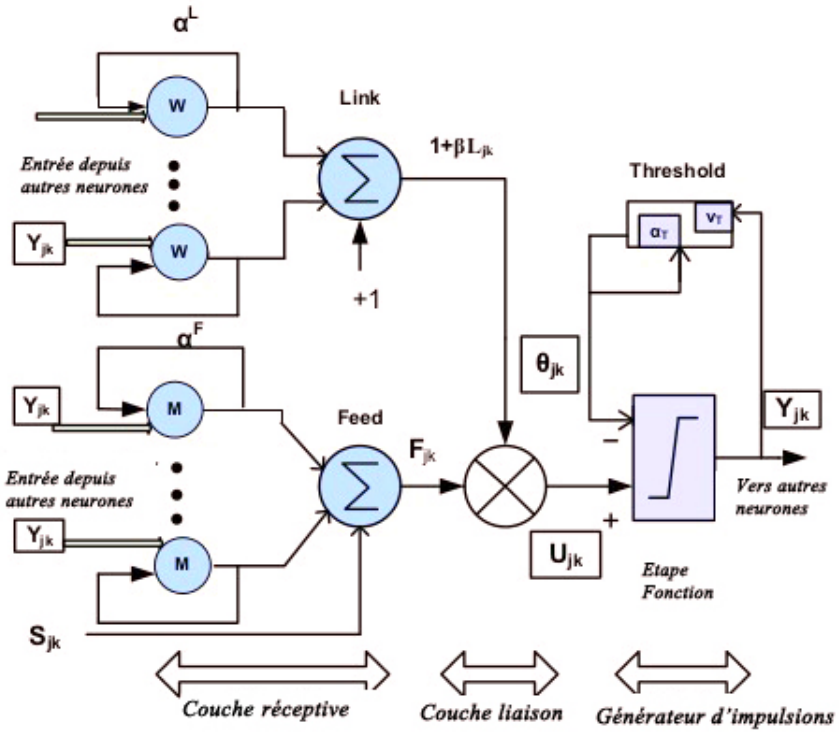


FIGURE *3.4* – Modèle mathématique d'un PCNN.

$$Y_{ij}[n+1] = \begin{cases} 1, & F_{ij}[n+1] > \theta_{ij} \\ 0, & ailleurs \end{cases} \tag{3.7}$$

$$\theta_{ij}[n+1] = g\theta_{ij}[n] + hY_{ij} + [n+1] \tag{3.8}$$

S_{ij} est le stimuli, h_{ij} est le seuil de neurone, Y_{ij} est la sortie et f, g et h sont des scalaires. Les paramètres ont été réduits et les entrées de liaison ont été faites d'une manière uniforme qui a abouti à un autre modèle appelé modèle Unité-Liaison. Ce modèle amélioré du PCNN a été utilisé dans de nombreuses applications par les chercheurs. Le PCNN est utilisé dans de nombreuses techniques de traitement d'image.

3.5 Applications du modèle PCNN

Le PCNN a été utilisé pour une variété d'applications de traitement d'images, incluant : segmentation d'images, extraction de visage, détection de mouvement, croissance de région, réduction de bruit, etc. [79]. Une revue complète sur les PCNN et ses applications depuis 1999 à 2015 est exposée dans l'article de Wang et al [80].
Le PCNN est un sujet de recherche intéressant dans le domaine de l'intelligence artificielle. Le PCNN est utilisé dans les algorithmes de construction pour diverses applications comme la segmentation, l'amélioration, la fusion, extraction de caractéristiques, détection

de bord, débruitage, la reconnaissance des formes, le décodage, l'amincissement d'image, etc. d'une façon générale dans les techniques de traitement d'image représentées sur la figure3.5. Les applications sont divisées en deux sections. La première section aborde l'utilisation du PCNN dans les procédures de traitement d'image, la segmentation, la fusion, la détection de caractéristiques, la suppression du bruit et la reconnaissance des formes des images ; tandis que la deuxième section examine les applications purement diverses. Le réseau trouve ses applications largement partout dans la médecine, la recherche, la surveillance, la reconnaissance des formes, etc.

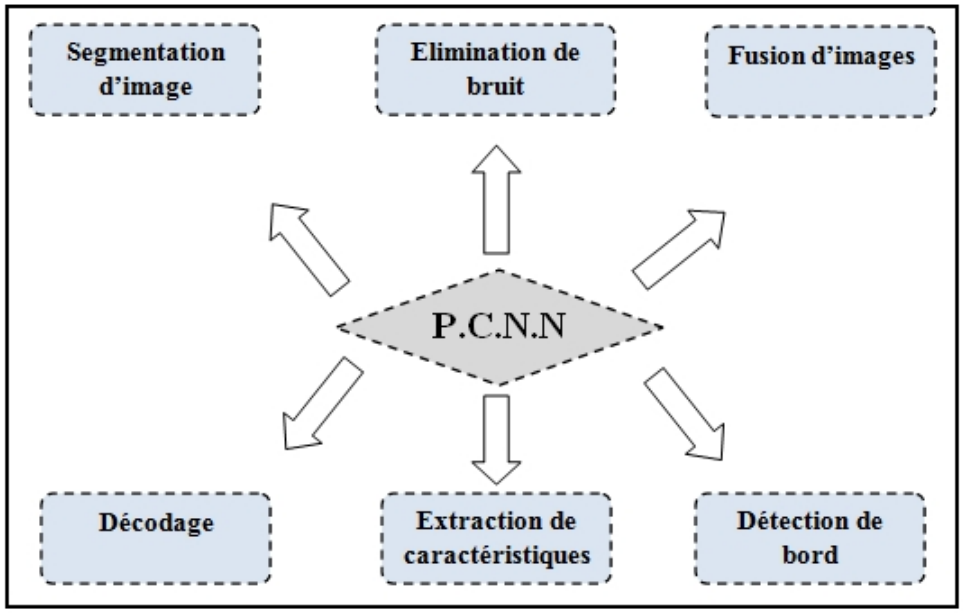


FIGURE *3.5* – Applications de PCNN.

3.5.1 La segmentation par le modèle PCNN

La segmentation d'images est une technique qui regroupe les pixels en régions, et définit donc les régions de l'objet figure3.6. La segmentation utilise les caractéristiques extraites de l'image ou des images. Par conséquent, une bonne sélection de caractéristique est importante dans la segmentation. L'inconvénient avec toutes les méthodes de segmentation est l'exigence de surveillance humaine qui va grandement influencer la précision. Donc, l'amélioration des performances et la diminution de temps d'exécution sont plus grandes préoccupations pour un bon système. Le PCNN traite de la plupart des inconvénients par des modifications selon les besoins.
Dans [81] Wang et al. ont utilisé un simple PCNN pour séparer un concombre d'un arrière plan complexe. Ils ont permis de réduire le nombre de paramètres en plus de l'ajustement du coefficient de poids à la connexion dans le modèle de réseau neuronal à impulsions couplées.

Un nouveau algorithme basé sur la recherche bidirectionnelle a été élaboré et mis en œuvre dans [82]. Le modèle a résolu le problème de la faible vitesse de calcul et donc a amélioré l'efficacité du travail et le modèle a également résolu les couleurs d'une image c-à-dire l'utilisation d'une information complète tout en segmentant des images en couleur. De même dans [83], Yin a fourni une direction multi-objet pour une segmentation d'image améliorée en utilisant le modèle PCNN.
Le problème difficile dans le modèle PCNN est le coefficient de liaison et le seuil d'activation primaire pour les neurones des statistiques d'images. Ce problème a été examiné par Rava et al dans [84] qui sont venus avec une solution de calcul d'un ensemble de paramètres globaux pour l'ensemble de l'image ou de paramètres locaux adaptatifs. Leur approche partage l'image en sous-images. Enfin, les paramètres globaux ont été calculés pour chaque sous-image.
Un nouveau modèle CMOS PCNN basé sur l'intégration et l'activation ont été mis au point [85].Le réseau (PCNN) a neurones IAF présente la dynamique électrochimiques de neurones biologiques naturels. Par ailleurs, le réseau peut ajuster les poids couplés de deux neurones (valeurs grises semblables) et de réaliser la fréquence et la synchronisation de phase. Dans le même domaine de l'amélioration de la performance du réseau neuronal à impulsions couplées de temps à autre, dans [86] Li et al. ont travaillé sur l'algorithme immunitaire qui a aidé les paramètres de PCNN à être ajustés de façon adaptative en prenant l'image d'entropie comme base d'évaluation. Ces auteurs ont prouvé par leurs résultats expérimentaux que la méthode proposée a donné une meilleure performance de la segmentation. La méthode de classification de Bayes a été incorporée avec PCNN [87] pour étendre la faisabilité du modèle développé pour l'extraction de cibles. C'est un algorithme itératif efficace pour la segmentation d'image automatique.
Un très bon résumé sur la stratégie de la segmentation itérative a été employé dans [88]. La précision de la segmentation dépend de paramètres du réseau. D'où le PCNN original a été simplifié en termes d'entrée et de seuil de neurones dynamique. Ils ont expérimenté avec des images infrarouges synthétiques et réelles et ont démontré la capacité de segmentation.

3.5.2 Planification du chemin entre deux positions par le modèle PCNN

Nous avons vu dans la section précédente l'utilisation de PCNN dans les procédures de traitement d'image et de segmentation. A présent, nous allons voir autres applications de ce modèle surtout dans le domaine de la navigation des robots mobiles.
A ce stade, la proposition de Yakoubi et al. dans [89], d'une approche évolutionniste pour la planification de chemin pour un robot mobile dans un environnement dynamique, a consommé beaucoup d'énergie (durée de déplacement) ; tandis que la durée de déplace-

FIGURE *3.6* – Ségmentation d'une image.

ment calculée par Qu et al. dans [90] était courte où ils ont proposé aussi un modèle PCNN pour la planification d'une trajectoire d'un robot mobile dans un environnement dynamique en temps réel. Pour cela, la carte topologique organisée de ce modèle est exprimée dans un espace d'état de dimension finie S qui représente l'espace cartésien de navigation des robots mobiles. Dans la figure 3.7 chaque noeud du graphe correspond à un neurone PCNN tandis que chaque segment associe une liaison entre les neurones. Chaque neurone n'a des connexions latérales locales uniquement qu'avec ses neurones voisins qui constituent un sous-ensemble R_i dans l'espace S.
Chaque $neurone_i$ représente soit un obstacle soit une position libre qu'occupe le robot dans l'espace de travail. Le poids de connexion W_{ij} représente la distance euclidienne entre le $neurone_i$ et le $neurone_j$ dans l'espace S, dont un exemple est illustré dans la figure 3.8 entre le $neurone_1$ et les $neurones_{2,3,4}$.

Dans ce modèle du PCNN, pour trouver le plus court chemin durant la planification entre une position de départ et une position d'arrivée, le modèle devrait produire une onde automatique par l'activation du neurone d'arrivée. A l'instant (t_0) où $t = t_0$ le neurone d'arrivée désigné en tant que P_t s'active et émet des impulsions. Ensuite, l'auto-ondes se propagent instantanément d'un neurone à un autre jusqu'au contacte avec le neurone de départ qui est désignée comme P_s. Cet évènement est représenté dans la figure 3.9.

Chaque $neurone_i$ possède une seule sortie Y_i tel que :

$$Y_i(t) = Step(U_i(t) - \theta_i(t)) = \begin{cases} 1, & \text{if } U_i(t) \geq \theta_i \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

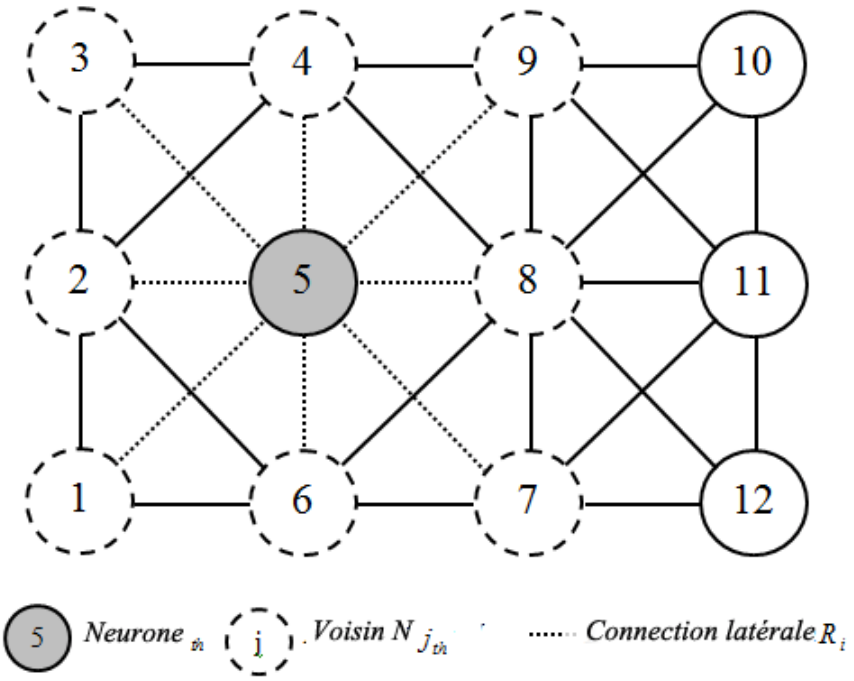


FIGURE *3.7* – Topologie de PCNN proposée.

Pour tout $i = 1, 2, 3, ...N$, où $U_i(t)$ et $\theta_i(t)$ sont respectivement la fonction d'activité interne et la fonction de seuil, N étant le nombre total de neurones et t représente le temps.
La fonction de seuil de neurone peut être exprimée comme suit :

$$\theta_i(t) = \begin{cases} A_{init}, & \text{when } t < t_{fire}^{R_i^F} \\ A_{i,j}, & \text{when } t_{fire}^{j} \leq t < t_{fire}^{i}, j \in \xi(i,t) \\ V_\theta, & \text{when } t \geq t_{fire}^{i} \end{cases} \tag{3.10}$$

où R_i^F et t_i^F sont respectivement le premier neurone à activer parmi ses voisins et la durée d'activation de ce neurone et où A_{init} et V_θ sont les constantes positives. La valeur $A_{i,j}$ est déterminée par :

$$A_{i,j} = \begin{cases} A^r, & \text{if } j \in R_i^r \\ A^l, & \text{if } j \in R_i^l \end{cases} \tag{3.11}$$

où R_i^l et R_i^r sont les deux sous-ensembles qui sont définis par :

$$\begin{aligned} R_i^l &= \{j \in R_i, w_{i,j} = 1\} \\ R_i^r &= \{j \in R_i, w_{i,j} = \sqrt{2}\} \end{aligned} \tag{3.12}$$

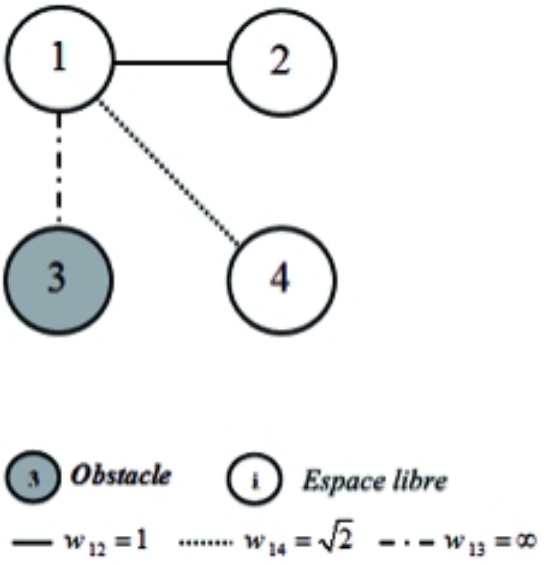


FIGURE *3.8* – Le poids de connexion de réseau.

L'activité interne $U_i(t)$ peut être exprimée comme suit :

$$\begin{cases} U_i(t) = 0, & \text{when } 0 \leq t < t_{fire}^{R_i^F} \\ U_i(t) = 0, & \text{when } t = t_{fire}^{j} \text{for any } j \in \xi(i,t) \\ \frac{dU_i(t)}{dt} = -\mu(w_{iR_i^P})U_i(t) + C, & \text{when } t \geq t_{fire}^{R_i^P} \end{cases} \tag{3.13}$$

où C est une constante positive donnée par :

$$\mu(w_{ij}) = \frac{B}{w_{ij}} \tag{3.14}$$

où B est une constante positive.

3.6 Conclusion

Nous remarquons que le modèle PCNN est puissant si les paramètres sont réglés correctement. Dans ce chapitre nous avons examiné la structure de base d'un réseau neuronal à impulsions couplées et les modifications consécutives faites par les chercheurs.

Le PCNN a été utilisé dans une variété d'applications de traitement d'images, incluant : la segmentation d'images, l'extraction de visage, la détection de mouvement, la croissance de région, la réduction de bruit, etc. Ce modèle a vraiment démontré son efficacité en dehors du domaine de traitement d'image surtout dans les applications robotiques où il a été utilisé pour la planification du chemin d'un robot mobile autonome en temps réel et qui a prouvé que les caractéristiques de transmission de ses impulsions peuvent

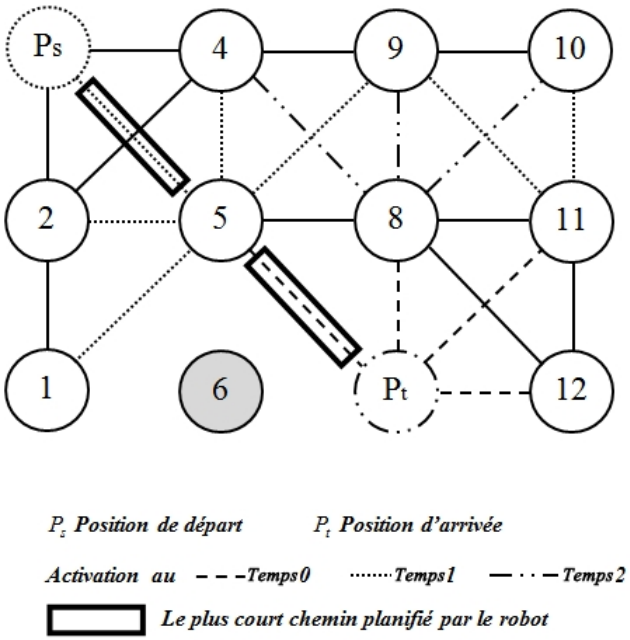


FIGURE *3.9* – Planification de plus court chemin.

rapidement trouver le chemin le plus court à partir d'un point d'arrivée vers le point de départ du robot.

De nombreux chercheurs utilisent le PCNN dont l'efficacité a été mise en évidence et prouvée, le PCNN est devenu dés lors un outil très utile dans notre vie quotidienne.

Grâce à la diversité des domaines d'applications, nous pouvons revenir sur la réflexion autour du problème de la planification du chemin pour la couverture complète d'un environnement ou de la planification d'un chemin pour un robot aspirateur (décrit dans le *Chapitre* 2). Donc, ce qui nous conduit à tester et concevoir le modèle du PCNN pour la résolution de ce problème.

Chapitre 4
Modélisation et simulation du modèle PCNN

4.1 Introduction

Dans le chapitre précédent, nous avons évoqué les différentes applications du modèle PCNN pour résoudre certains problèmes rencontrés dans notre vie quotidienne. La question est de savoir si le recours au modèle PCNN ne concerne et ne s'applique que dans le domaine de traitement d'image.

Cependant dans ce chapitre, nous allons proposer un modèle PCNN qui se base sur l'activation des neurones dont le but de traiter le problème cité précédemment dans le chapitre 2, où les auteurs, dans leurs approches, se sont heurtés à divers problèmes tels que "la planification de chemin pour un robot aspirateur nettoyeur de pièce " et à la difficulté de les intégrer dans un environnement dynamique.
Nous allons employer des capteurs (sonars et cameras) pour utiliser la planification en ligne au lieu de la planification hors ligne. Le processus de ce modèle s'effectue par étapes, où à chaque position, le robot évalue et choisit le meilleur de chemins proposés et se déplace selon celui-ci. Une fois le chemin parcouru, le robot aspirateur réitère le processus en nettoyant une autre zone libre, etc.

4.2 Modélisation cinématique

D'après les auteurs de [43,91–95], la cinématique est l'étude du mouvement des points, des objets ou système d'objets sans prendre en considération les forces en raison de laquelle le mouvement est provoqué. Elle est aussi appelée la géométrie du mouvement. En d'autres termes, c'est l'étude du comportement de systèmes mécaniques. Fondamentalement, les trajectoires des points, des lignes et d'autres propriétés géométriques, telles que la vitesse et l'accélération sont étudiés en vue de l'analyse cinématique. L'analyse cinématique est utilisée pour mesurer les quantités cinématiques utilisées pour décrire le mouvement d'un objet. En cas de robots mobiles, l'analyse cinématique nous aide à comprendre le comportement mécanique des robots afin de créer un logiciel de contrôle pour le matériel de robot mobile.

Tous les systèmes mécaniques nécessitent une telle analyse pour les aider à mieux travailler. Les manipulateurs robotiques ont également été soumis à une telle analyse depuis plus de trente ans. Ces manipulateurs étaient plus complexes que les robots ordinaires et donc une analyse plus profonde était nécessaire pour leur compréhension. Les questions sur la cinématique de la robotique mobile sont les mêmes que celles posées pour les manipulateurs mobiles.

L'estimation de la position du robot a également beaucoup de dissemblances par rap-

port à l'estimation de position du manipulateur robotisé dans le cas d'un manipulateur, dont une extrémité est fixée à l'environnement. Donc la position du manipulateur est considéré comme la position de l'effecteur d'extrémité. Cependant, dans le cas d'un robot mobile sur roues l'ensemble du robot est mobile et sa position n'est pas un facteur à mesurer instantanément. Sa position doit être estimée en intégrant son mouvement au fil du temps et en ajoutant des inexactitudes de l'estimation de mouvement dûe à la contrainte de glissement.

4.2.1 Contraintes cinématiques

La dérivation d'un modèle pour le mouvement de l'ensemble du robot est une approche de plus bas jusqu'au plus haut. Le fait que chaque roue contribue à la commande de mouvement du robot en même temps elle impose des contraintes au le robot. Cependant, les roues fonctionnent avec l'ensemble de la géométrie du châssis du robot de sorte que les contraintes sont également combinés entre elles pour former les contraintes de mouvement de cet ensemble, tandis que les contraintes et les forces de chaque roue doivent être exprimées par rapport à un cadre de référence lucide et fiable. Ceci est particulièrement important en robotique mobile en raison de son autonomie et sa nature de déplacement. Les contraintes imposées par les roues sont :

- ☐ Il devrait y avoir un mouvement de roulement.
- ☐ Il ne devrait y avoir aucun glissement latéral.

4.2.2 Conception des roues

La conception d'un robot mobile à roues est subordonnée aux types de roues et à leurs systèmes de configuration et de commande. La caractéristique de mobilité du robot est définie en utilisant ces paramètres.
Les roues utilisées dans l'architecture d'un robot affectent la cinématique globale du robot. La roue doit être guidée le long d'un axe vertical si on veut changer sa direction ; cet axe de rotation de la roue joue un rôle vital dans le mouvement de la roue :
Les roues peuvent être divisées en cinq catégories principales en fonction de leur cinématique.

- ☐ Roue standard fixe (Figure4.1.)
- ☐ Roue standard orientable (Figure4.2.)
- ☐ Roue de Caster (Figure4.3.)
- ☐ Roue suédoise (Figure4.4.)
- ☐ Roue sphérique (Figure4.5.)

La roue-standard fixe est rêvée au châssis du robot et elle se déplace uniquement en arrière et en avant :

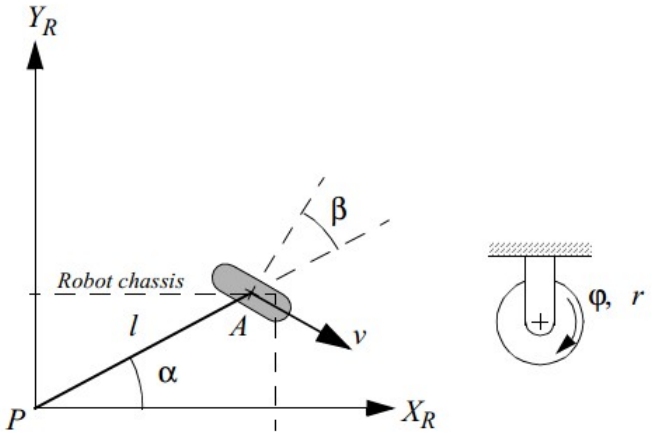


FIGURE *4.1* – Roue standard fixe.

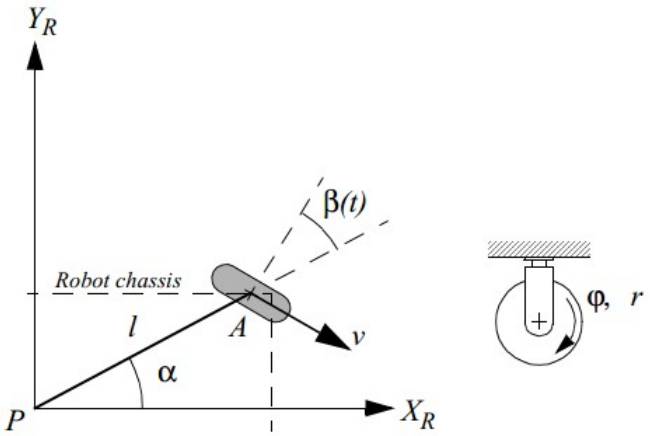


FIGURE *4.2* – Roue standard orientable.

- Il n'y a aucun axe de rotation vertical de direction pour la roue-standard fixe.
- L'angle entre le châssis et l'axe de roue est fixé.

Elle se limite à déplacer le robot au va-et-vient le long du plan de roue et à le faire tourner autour de son point avec le plan de masse de contact. Cette roue est utilisée pour des charrettes où la direction du véhicule est modifiée par le mouvement des Bœufs.

La roue standard orientable est fixée sur le châssis du robot, mais a un axe de rotation vertical. Elle est différente de la roue-standard-fixe pour avoir un degré de liberté supplémentaire :

- Il y a un axe de rotation vertical.
- L'angle entre le châssis et l'axe de roue est non fixé et il change à chaque direction.

De telles roues sont utilisées dans tous les véhicules de transport de sorte que leur orientation est variée selon le changement de direction des roues.

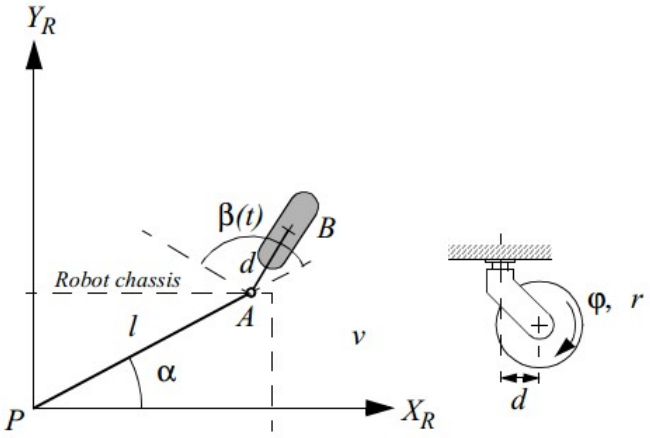


FIGURE *4.3* – Roue de Caster.

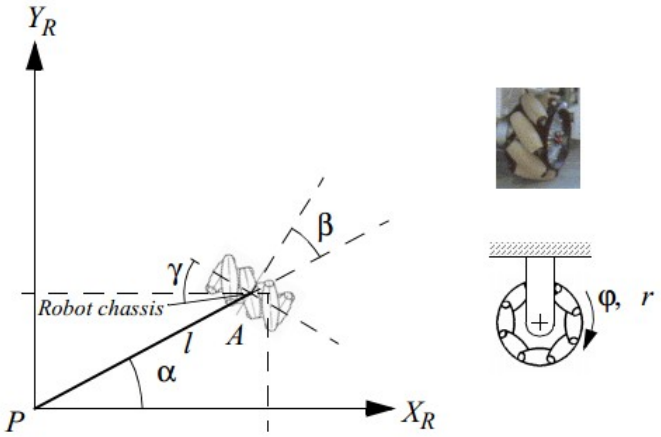


FIGURE *4.4* – Roue suédoise.

La roulette (Roue de Caster) est décalée par rapport au châssis. Sa roue de rotation ne passe pas à travers le point de contact avec le sol :

- Il existe deux axes, l'un à travers le châssis du robot et l'autre à travers la roue.
- Les deux axes ont un écart entre eux.

Ces roues sont utilisées pour des chariots, des chaises de bureau et de matériel de manutention industrielle.
La roue suédoise ou la roue omnidirectionnelle a de petits disques autour de sa circonférence qui sont perpendiculaires à la direction de laminage. La roue peut rouler à toute pleine vitesse ainsi qu'en coulissant latéralement facilement :

- Il n'y a pas d'axe de rotation vertical et le mouvement se fait en ajoutant un degré de liberté supplémentaire à la roue-standard-fixe .

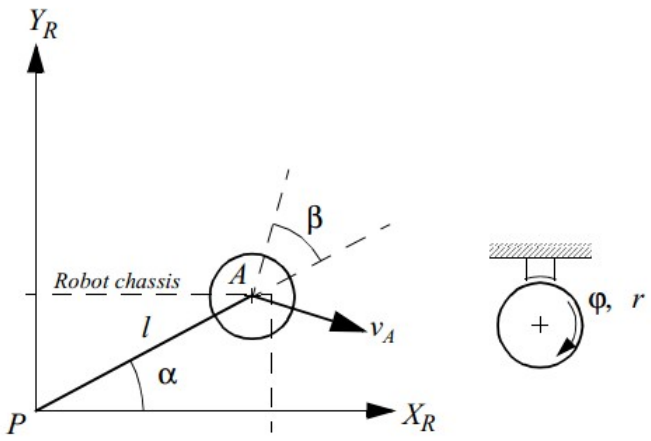


FIGURE *4.5* – Roue sphérique.

- Les rouleaux attachés à la circonférence de la roue ont des axes antiparallèles à l'axe principal de la roue.

Ces roues sont utilisées pour les systèmes holonomes d'entraînement et dans les robots qui ont besoin de se déplacer dans toutes les directions.
La roue sphérique n'a pas d'axe de rotation direct et n'a pas de contraintes de roulement ou de glissement. Donc elle est omnidirectionnelle :

- Il n'existe aucun axe principal de rotation.
- Elle peut tourner dans toutes les directions.

Une telles roues est utilisée pour une souris d'ordinateur par exemple.

4.2.3 Robots mobiles à roues

Les roues sont des appareils de locomotion les plus appropriés en cas de robots et autres véhicules fabriqués par l'homme car ils peuvent atteindre un niveau d'efficacité décent et ne pas avoir besoin d'une mise en œuvre mécanique compliquée. En règle générale, les robots mobiles à roues n'ont pas un problème d'instabilité en mouvement avec au moins trois roues.
Cependant, un robot à deux roues peut également se déplacer de façon constante alors que dans le cas de plus de trois roues, un système de suspension est nécessaire pour maintenir le contact du robot avec la surface du sol. D'où le principal sujet de préoccupation est la traction fournie par la roue afin de couvrir tous les types de terrain qu'il traverse avec la vitesse requise.

Selon l'orientation et les types de roues utilisées, les robots mobiles à roues peuvent être classés en

- ☞ Robot bicycle ou tricycle - une roue indépendante orientable et autre indépendante fixe.
- ☞ Robot à conduite différentiel - une roue fixe indépendante et autres roues omnidirectionnelles.
- ☞ Robot à conduite Synchronisée - une roue orientable indépendante et autres roues omnidirectionnelles
- ☞ Robot omnidirectionnel - seulement des roues de Castor et suédoise.
- ☞ Robot à conduite différentiel à deux roues - deux roues orientables indépendantes.

.
En fonction du nombre de roues dans le robot, les robots mobiles à roues peuvent être divisés en

- ➳ Robot mobile à deux roues
- ➳ Robot mobile à trois roues
- ➳ Robot mobile à quatre roues
- ➳ Robot mobile à six roues

4.2.4 Position du robot

La position du robot doit être exprimée en termes d'une relation entre le repère de la plate-forme globale du robot et le repère de la plate-forme locale du robot. Dans ce cas, X_I et Y_I définissent une base inertielle arbitraire sur le plan comme le repère de plateforme globale origine $O\{X_I, Y_I\}$. Afin de préciser la position du robot, nous choisissons un point P sur le châssis de robot comme un point de repère de position. $\{X_R, Y_R\}$ définissent deux axes par rapport à P sur le châssis du robot ; donc c'est le repère de la plateforme locale du robot. La position du robot peut être décrite en utilisant les coordonnées x et y et la différence angulaire entre les repères des plateformes locale et globale, donnée par θ. (Figure4.7). La position du robot peut être décrite à l'aide de ces facteurs sous la forme d'une matrice.

$$\xi_I = \begin{bmatrix} x & y & \theta \end{bmatrix}^T \tag{4.1}$$

La matrice appelée matrice orthogonale est utilisée pour cartographier le référence de la plateforme globale du robot dans son repère de plateforme locale.

$$R(\theta) = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

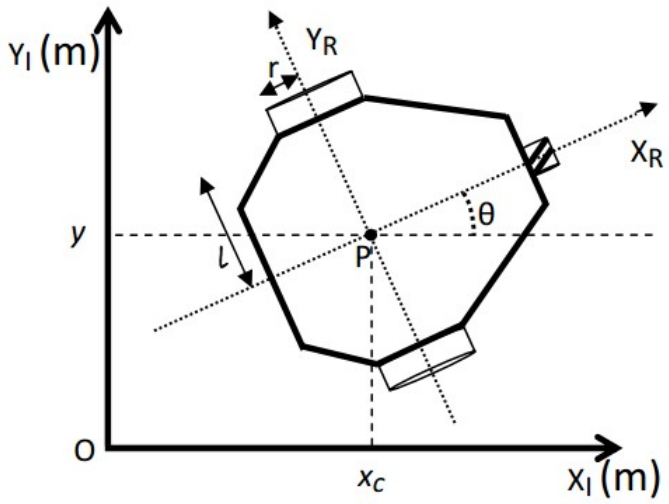


FIGURE *4.6* – Position du robot dans le repère de plateforme locale.

Le fonctionnement de la cartographie de la plateforme globale avec la plateforme locale est effectuée en utilisant la matrice de rotation orthogonale qui est exprimée par

$$R(\theta)\xi_I \tag{4.3}$$

4.2.5 Robot mobile de type tricycle

Ce type de robot est constitué de deux (02) roues fixes de même axe et d'une troisième roue orientable située au centre et placée sur l'axe longitudinal du robot. La vitesse longitudinale et l'orientation de la roue orientable sont les deux actions qui aident le robot à réaliser son mouvement.

4.2.5.1 Centre instantané de rotation (CIR)

Le CIR d'un robot mobile de type tricycle se situe à la rencontre des axes des roues et de la roue orientable, telle que représentée dans la figure 4.7. On peut déterminer ρ de manière géométrique à partir de l'angle d'orientation de la roue avant et à partir de la vitesse linéaire υ du robot (vitesse en O') et de ρ :

$$\rho = \frac{D}{tan\psi'} \tag{4.4}$$

$$= \frac{\upsilon}{D}\, tan\psi \tag{4.5}$$

Ce type de robot peut se diriger en ligne droite pour $\psi = 0$ et théoriquement tourner autour du point O' (on pourrait dire sur place) pour $\psi = \frac{\pi}{2}$. Néanmoins, le rayon de braquage de la roue orientable, généralement limité, impose le plus souvent des valeurs

de ψ telles que $-\frac{\pi}{2} < \psi < \frac{\pi}{2}$, interdisant cette rotation du robot sur lui-même.

L'écriture des contraintes sur chacune des roues et un raisonnement similaire a celui suivi dans le cas de l'unicycle, permettent de déterminer les modèles cinématiques des robots de type tricycle. Toutefois, par un simple raisonnement géométrique, on établit les équations suivantes :

$$\dot{x} = \upsilon \cos\theta,$$

$$\dot{y} = \upsilon \sin\theta,$$

$$\dot{\theta} = \frac{\upsilon}{D} tan\psi,$$

$$\dot{\psi} = \eta,$$

où, $(u = (\upsilon\, \eta)^T)$ est le vecteur de commande cinématique ; η représente la vitesse d'orientation imposée à la roue orientable. Ces équations sont celles du modèle cinématique en configuration simplifié ; la configuration associé $(q = (xy\theta\psi))$ ne décret pas les rotations propres des différentes roues.

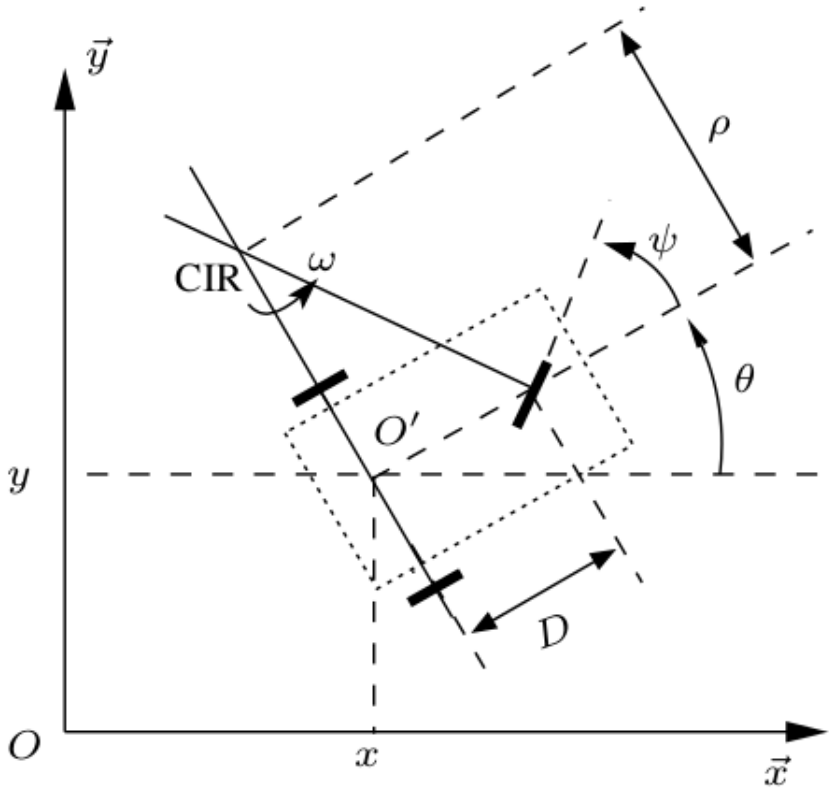


FIGURE *4.7* – Position du robot dans le repère de plateforme locale.

4.3 Architecture du robot

Le robot simulé dans notre étude est un robot aspirateur, qui, une fois dans une pièce, la nettoie en la parcourant plus ou moins aléatoirement et en évitant les obstacles et certains endroits et positions.
Le robot aspirateur est muni d'une (01) roue motrice et directrice commandée en fonction des obstacles détectés ainsi que deux (02) autres roues fixes de même axe. Ce robot est de forme circulaire, composé d'une camera au front et de 3 capteurs latéraux et arrières. La camera $S1$ et le capteur $S3$ assurent toutes collisions frontales et arrières ; quant aux capteurs $S2$ et $S4$ de 45° chacun, ils assurent l'évitement de toutes collisions latérales (Figure4.8)

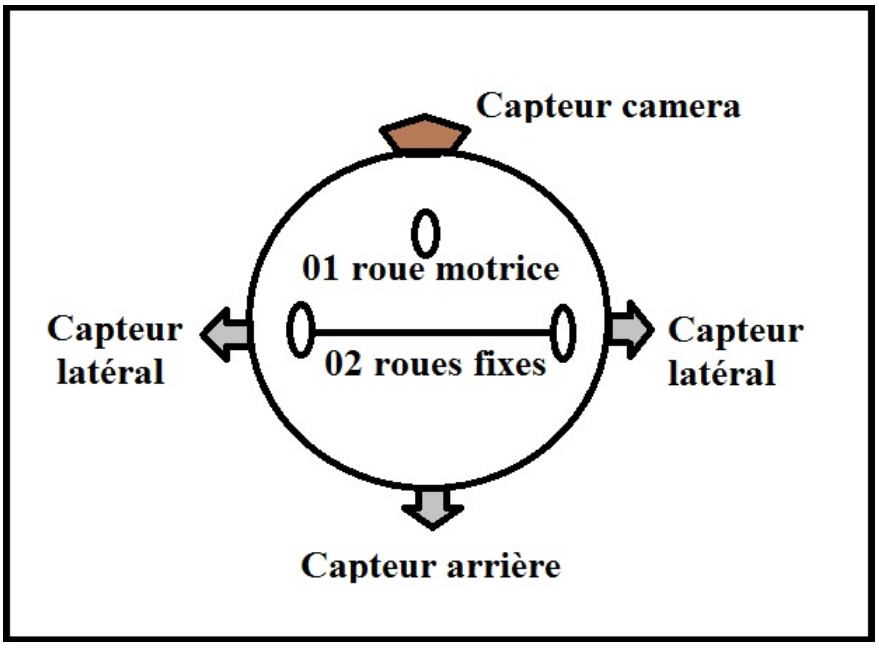


FIGURE *4.8* – Modélisation du robot.

4.4 Modélisation de l'environnement

Puisque nous utilisons une cartographie locale, l'environnement que perçoit le robot dépend directement de la portée de ses capteurs tels que la camera, capteur infrarouge etc.

Pour la simulation, nous avons modélisé l'environnement en disque de diamètre de 10 unités comme la forme circulaire du robot au lieu de la forme carrée ou rectangulaire. Cela nous permet d'envisager des obstacles aux formes plus extravagantes ainsi que la facilité de déplacement du robot dans cet environnement.
Nous considérons que le diamètre du disque doit être égal ou légèrement supérieur au

diamètre du robot.
Dans cette simulation le point de départ $S(x, y)$, la position du robot $P(x, y)$ à l'instant t_i, la position finale du robot $E(x, y)$ et les obstacles sont représentés respectivement par les couleurs rouge, bleu, vert et gris.(Figure4.9)

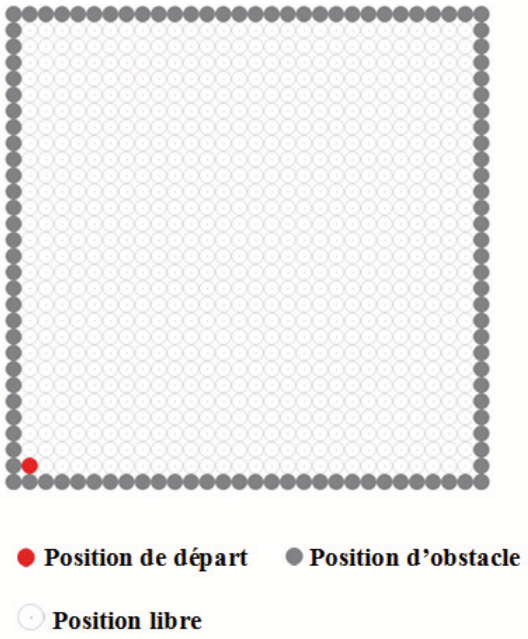


FIGURE *4.9* – Environnement du robot.

4.5 La couverture complète

Dans notre vie quotidienne, si nous voulons nettoyer une pièce, nous commençons généralement par un coin. Ensuite, nous choisissons la meilleure direction vers une zone à nettoyer dans le but d'obtenir le plus court chemin avec moins de zones revisitées et moins de tours de déplacement.
Cette intelligence humaine est applicable à un robot aspirateur pour qu'il puisse réaliser sa mission de balayage de tout l'espace de la pièce W pour chaque zone accessible avec évitement des obstacles.

Pour nettoyer une pièce simple vide, deux trajectoires-standards peuvent être parcourues en spirale [51] ou en zigzag [96]. (Figure4.10)
Par le balayage, nous entendons la couverture complète par le robot, de l'ensemble de l'espace à nettoyer W, tâche exécutable après la construction de la carte. Chaque cellule-disque de W peut être un obstacle ou une cellule libre dont le robot doit s'occuper. Quand une cellule libre est visitée par le robot elle sera nettoyée et si ce n'est pas le cas, elle reste non nettoyée.

Le problème de la planification du chemin pour la couverture complète d'un environnement est non seulement de passer sur chaque cellule libre mais aussi d'éviter les obstacles ; donc une planification d'un chemin efficace est fortement souhaitée.
De nombreuses approches de couverture complète ne sont pas, à ce jour, arrivées à obtenir un chemin efficace de courte distance, avec un nombre minimal d'angles de rotation et des cellules à revisiter. Ces approches ont par ailleurs, négligé le facteur de l'unité centrale de traitement (temps de CPU).

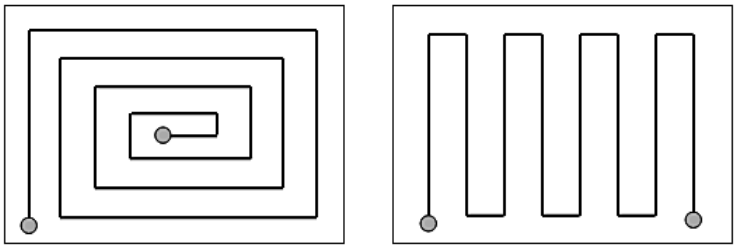


FIGURE *4.10* – Façons de couverture complète (a) trajectoire spirale (b)trajectoire zigzag.

4.5.1 Le modèle PCNN

Pour avoir une trajectoire efficace de la couverture complète d'un environnement par un robot aspirateur, nous appliquons le modèle PCNN et nous préservons la même structure et architecture illustrée dans la figure3.7.
La dynamique d'un neurone i sur le réseau neuronal est caractérisée par la même activité interne utilisée dans [90].

$$\frac{dU_i(t)}{dt} = -\mu(w_{iR_i^F})U_i(t) + C \tag{4.6}$$

Ce modèle est constitué de plusieurs neurones ; chaque neurone représente un obstacle ou une positon libre qu'occupe le robot dans l'environnement.
La tâche du robot aspirateur est de se déplacer d'un endroit libre vers un autre en évitant les obstacles. Ce déplacement est réalisé par le modèle PCNN.
L'évènement d'activation du modèle PCNN entre deux neurones représente le déplacement du robot entre deux positions. Si tous les neurones qui représentent les positions accessibles dans l'environnement par le robot sont activés, la mission de cet aspirateur est terminée.

Pour plus de précision, lorsque l'état d'activation se propage à partir d'une position P_i vers une autre position P_j, le robot aspirateur se déplace entre ces deux dernières

positons. Ce déplacement entre P_i et P_j ne peut pas être aléatoire mais soumis à plusieurs conditions.
Pour cela, cet évènement ne se propage pas à tous les voisins du neurone comme dans [90], mais il se propage vers un seul voisin ; ce dernier représente la position suivante que prend le robot aspirateur comme illustré dans la figure4.11. La nouvelle position que doit prendre le robot P_n est sélectionnée parmi l'une des positions voisines non encore nettoyées. Cette sélection est une tâche qui prend en considération la position précédente P_p du robot, la position actuelle P_c, la distance, la position x sur l'axe des abscisses et l'angle de braquage calculé entre la position actuelle P_c et la prochaine position proposée P_j du robot parmi ses voisins.
La position proposée P_j qui aura la quantité minimale Q_{p_n} sera la nouvelle position finale sélectionnée P_n vers laquelle le robot doit se déplacer.
Pour calculer la quantité Q_{pn} de chaque prochaine position proposée P_j, nous proposons une nouvelle fonction qui se base sur les paramètres décrit ci-dessus et qui peut être représentée par l'équation :

$$P_n \Leftarrow Q_{P_n} = min\{(x_j - x_c) + w_{cj} + D_j, j = 1, 2, ..., k\} \tag{4.7}$$

où k est le nombre des neurones voisins du P_c ; x_j et x_c sont respectivement des positions du voisin j et P_c sur l'axe des abscisses ; w_{cj} est la distance euclidienne entre la position actuelle P_c et la nouvelle position possible P_j ; D_j est une fonction monotone croissante de la différence entre la direction actuelle et l'autre proposée lors du mouvement du robot ; cette fonction peut être calculée avec la position précédente P_p, l'actuelle P_c et la prochaine possible P_j par la fonction suivante :

$$D_j = \frac{\Delta\alpha_j}{\pi} \tag{4.8}$$

où $\Delta\alpha_j \in [-\pi, \pi]$ est l'angle de braquage entre la direction actuelle et la prochaine que peut prendre le robot ; $\Delta\alpha_j$ est obtenue comme suit :

$$\alpha_j = atan2(y_{p_j} - y_{p_c}, x_{p_j} - x_{p_j}) \tag{4.9}$$

$$\alpha_c = atan2(y_{p_c} - y_{p_p}, x_{p_c} - x_{p_p}) \tag{4.10}$$

$$\Delta\alpha_j = \alpha_j - \alpha_c \tag{4.11}$$

Après que le robot ait sélectionné sa meilleure nouvelle position P_n par l'équation (4.8), l'état d'activation commence à propager des auto-ondes vers cette nouvelle position P_n ; celle-ci devient la nouvelle position actuelle du robot et la dernière position actuelle

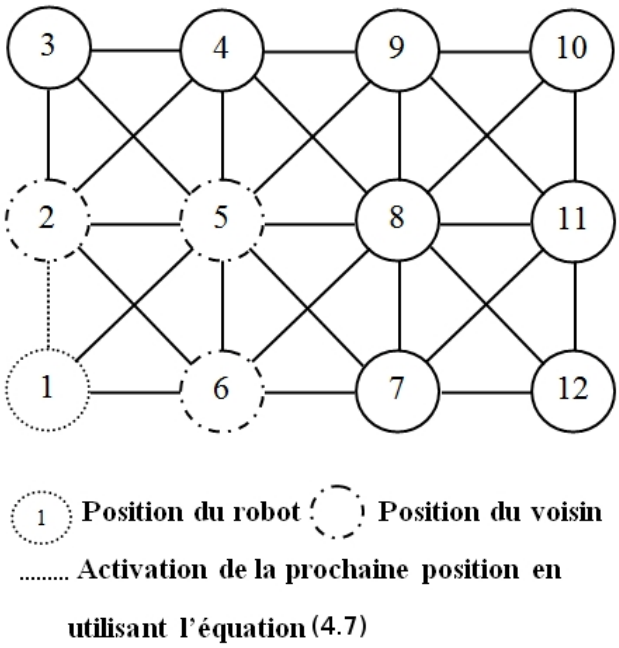


FIGURE *4.11* – Sélection de la nouvelle position du robot.

devient la nouvelle position précédente. La cartographie de l'environnement est mise à jour aprés cette opération.
Le chemin du robot est généré à partir de la fonction d'activation interne et la sélection de la meilleure prochaine position en utilisant les équations (4.6) et (4.7) pour minimiser le nombre de tours et la longueur de la trajectoire.

4.5.2 Algorithme de Couverture complète

Pour que robot accomplisse sa tâche qui est la couverture complète de toutes cellules non nettoyées dans un environnement avec évitement des obstacles, il doit passer par chaque cellule accessible et non encore nettoyée. Cette cellule représente un neurone dans l'architecture de réseau neuronal du modèle PCNN décrit dans les sections précédente.

L'événement d'activation est soumis à un algorithme qui permet au robot aspirateur d'accomplir sa tâche de couverture complète d'un environnement avec un chemin très efficace.

Nous pouvons diviser l'exécution de cet algorithme en deux (02) phases : *une phase initiale et une phase de couverture complète* .
La phase initiale est indispensable parce que le réseau neuronal devrait être stable au premier temps ($t = 0$) comme il sera expliqué dans les Algorithme1.1 et 1.2 ; la positon de départ du robot $P(x_0, y_0)$ devrait être aussi connue à l'avance.
Après l'achèvement de la phase initiale, la phase de couverture complète commence. Dans

cette phase, l'événement de l'activation du modèle PCNN se déclenche sous forme d'itérations.
Durant chaque itération, lorsque un neurone j s'active au temps, $t = t_{active}^{neurone}$, il sera considéré comme une nouvelle position sélectionnée où le robot aspirateur se déplace directement vers cette position $P_n(x_n, y_n)$. Après ce mouvement de déplacement et exactement lorsque le robot arrive à la position $P_n(x_n, y_n)$, la cartographie de l'environnement est mise à jour et dans laquelle la position actuelle $P_c(x_c, y_c)$ et la nouvelle position trouvée $P_n(x_n, y_n)$ seront respectivement transformées en une positon précédente $P_p(x_p, y_p)$ et une positon actuelle $P_c(x_c, y_c)$. Après tous ces événements, le robot va lancer une autre recherche d'une nouvelle et d'une meilleure prochaine position $P_n(x_n, y_n)$ en utilisant l'équation (4.7).
Toutes ces étapes seront répétées jusqu'au ce que toutes les neurones accessibles se soient activés. Dans ce cas, le réseau neuronale s'arrête de fonctionner ; cela signifie que toutes les cellules de l'environnement ont été nettoyées par le robot.

Pour plus de clarification, un exemple est illustré dans la Figure4.12 où nous remarquons que le robot exécute l'algorithme précédent pour résoudre le problème de couverture complète d'un environnement et aussi pour avoir un chemin très efficace entre la position de départ $P_s(x_s, y_s)$ et la position finale $P_e(x_e, y_e)$.

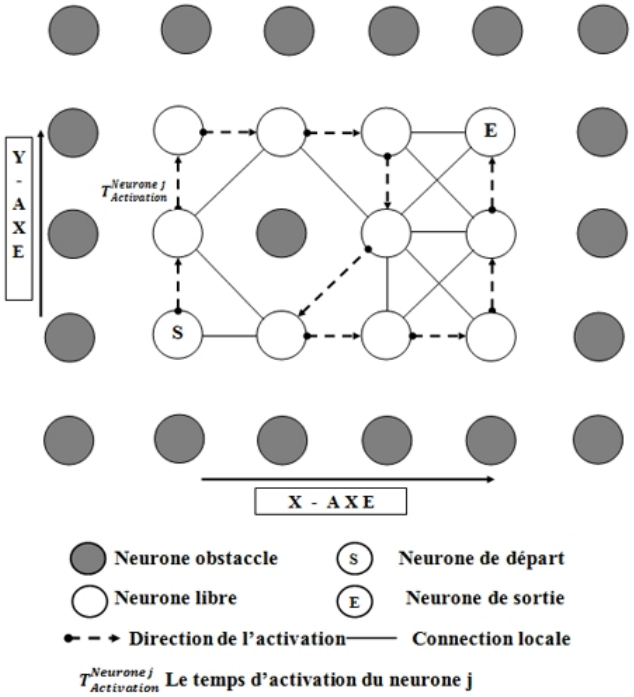


FIGURE *4.12* – La planification de chemin en utilisant l'algorithme de couverture complète.

Algorithm 1 Algorithme initial

Entrées : Le modèle du réseau neuronal de l'environnement et la position de départ du robot.
Sorties : Mise à jour du modèle de réseau neuronal de l'environnement.

~~1)~~ Initialisation de l'environnement : Déterminer A, B, C et V_θ selon les conditions dans [90].

2) Déterminer $\theta_i(0) = A_{init}$. Déterminer $U_i(0) = 0$ pour tous $i = 1, 2, ..., N$, $i \neq Start$ et $U_{Start}(0) = A_{init}$.

3) Déterminer $x_c = x_0, y_c = y_0$, où $P(x_0, y_0)$ est la position de départ.

Cela rend le neurone de départ prêt à s'activer au moment $t = 0$.

Algorithm 2 Algorithme de couverture complète du robot aspirateur

Entrées : Le modèle du réseau neuronal de l'environnement et la position actuelle du robot.
Sorties : Mise à jour du modèle de réseau neuronal de l'environnement ainsi que nouvelle prochaine position du robot.

~~1)~~ Démarrage du réseau neuronal : Pour chaque neurone i dans le réseau :

a) Calculer $U_i(t)$, selon(3.13)

b) Calculer $Y_i(t)$, selon(3.9)

c) Lorsque $Y_i(t) = 1$:

i) Déterminer $\theta_i(t) = V_\theta$, selon(3.10)

ii) Trouver la prochaine position du neurone voisin $P_n(x_n, y_n)$ selon (4.7)

iii) Déterminer $\theta_n(t) = A_{ni}$ selon(3.11)

iv) Le robot commence son déplacement à partir de sa position actuelle P_c vers sa nouvelle prochaine position P_n

v) Déterminer $x_c = x_n$ et $y_c = y_n$ Mettre à jour la nouvelle actuelle position

2) *Si* tous les neurones accessible se sont activés

a) Arrêter le réseau.

(Tous les endroits de l'environnement sont nettoyés)

3) *Sinon* aller vers **1**

4.6 Planification du chemin pour la couverture complète

Comme précédemment mentionné sur le mécanisme de la couverture complète d'un environnement, nous présentons dans cette section un exemple qui illustre une application

du modèle PCNN par un robot aspirateur dans une petite pièce.

La figure4.13 montre un graphe qui représente la cartographie de l'environnement que doit nettoyer le robot. Cet environnement a été divisé en plusieurs disques (modélisation de l'environnement). Ces disques peuvent être des obstacles de couleur grise ou non encore nettoyés de couleur blanche ; par contre les positions initiale et finale du robot sont représentées respectivement par les couleurs rouge et verte.

Dans la figure4.13 l'architecture du réseau neuronal est constitue de 30x30 neurones ; chaque neurone représente un obstacle une position libre dont le robot doit s'occuper durant le nettoyage. Nous considérons que la position de départ est connue à l'avance.
Nous remarquons dans la figure4.13 que le robot aspirateur a commencé à partir de la position $S(1,1)$ et a terminé son nettoyage à la position $F(28,1)$. Durant ce déplacement, l'événement d'activation du modèle PCNN se déclenche et se propage à partir du neurone de départ vers un seul neurone parmi ses voisins en appliquant l'équation 4.7. Puis, le robot va se déplacer de sa positon actuelle vers la nouvelle position trouvée, et un mini chemin sera tracé en rouge. Après un certain moment, cette nouvelle position sera la position actuelle du robot qui s'activera pour continuer le nettoyage.
Le système de propagation des ondes va suivre les mêmes précédentes étapes effectuées par le neurone actuelle jusqu'à l'activation de dernier neurone qui représente la positon finale du robot non encore nettoyée $F(28,1)$.

La planification du chemin pour le nettoyage complet de la pièce par le robot s'est réalisé sous forme de zigzag, de haut-en-bas et de bas-en-haut avec évitement des obstacle. Cette trajectoire est représentée par une fine ligne rouge entre la position de départ $S(1,1)$ et la positon finale $F(28,1)$ du robot.
Ce déplacement ne se fait pas au hasard, il est planifié par le modèle PCNN avec ses deux $algorithmes$1.1 et1.2.

4.7 Conclusion

Nous avons présenté notre méthode portant sur la conception et la modélisation de notre modèle (PCNN). Cette conception est basée sur trois modélisation : la première est basée sur la modélisation du robot aspirateur de forme circulaire, comportant 4 capteurs et muni de trois roues (une roue motrice directrice et 02 autres roues fixes) ; la deuxième est basée sur la modélisation de l'environnement en disque de diamètre 10 unités en forme circulaire comme le robot. Cela nous permet d'envisager des obstacles aux formes plus extravagantes et d'obtenir la facilité du déplacement de robot dans cet environnement. La dernière et troisième modélisation est basée essentiellement sur l'architecture du modèle

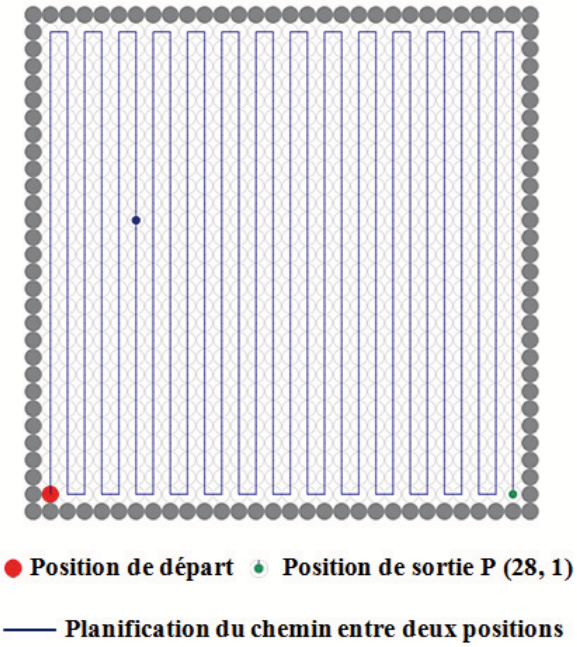


FIGURE *4.13* – une planification du chemin pour une couverture complète d'un environnement.

PCNN où ce modèle est constitué de plusieurs neurones, chaque neurone représentant soit un obstacle soit une positon libre occupée le robot dans l'environnement. Ainsi, ce modèle réalise la mission de nettoyage du robot aspirateur.

Chapitre 5
Tests et Résultats

5.1 Introduction

Le modèle PCNN proposé est basé sur l'évènement d'activation de neurones dont le but est de générer un chemin et d'aider le robot aspirateur à nettoyer toute une surface. Ce chemin parcouru par ce robot n'est pas aléatoire mais il doit être soumis à différents critères. Ces critères comme la longueur du chemin, le nombre de tours et le nombre de cellules revisitées pour le nettoyage de toute la surface jouent un rôle et influencent le bon fonctionnement de la tâche du robot qui est la couverture complète surtout au niveau du temps écoulé et la consommation de la batterie.

Le travail que nous proposons dans ce chapitre est de tester l'efficacité et la robustesse de notre algorithme et modèle proposés dans le chapitre 4 précédent. Pour ce faire, nous divisons le travail en deux :
Dans un premier lieu, nous validons le test dans un environnement de simulation, où la carte de l'environnement est modélisée avec des cellules en forme de disques. L'algorithme est testé dans différents états de l'environnement (une surface connue, complètement inconnue ou dynamique.)
Dans un second lieu, nous comparons notre méthode [97] avec d'autres techniques existantes dans la littérature (Algorithmes de CAC [98], BA* [99], LSSP [100] et le modèle de Luo et Yang [96]) .
Les simulations sont mises en œuvre en langage *C#*. Les limites de la surface sont supposées être connues durant la simulation.

5.2 Application développée

Nous avons développé une application qui reprend les idées proposées dans le chapitre précédent. Cette application a été écrite en Langage *C#* sous le système d'exploitation Windows 7 Professionnel (Tableau 5.1). Le choix de ce langage est dù principalement aux énormes qualités et au succès indiscutable qui le qualifient.

Traditionnellement, ce genre de méthodes sont développées sous le langage C voire même C++. Mais il est indéniable que travailler sous ces langage demandent un temps et un effort intellectuel importants. *C#* base sa philosophie sur la simplicité d'écriture de code. Écrire un code sous *C#* apporte un gain de temps. Donc, le développeur, plutôt que de réfléchir à la manière d'implémenter une solution va plutôt travailler à améliorer celle-ci sans contrainte de programmation.

C# étant Orienté-Objet, nous avons trouvé une grande facilité à développer le modèle

TABLE *5.1* – Configuration matérielle pour la simulation

Ordinateur	DELL VOSTRO 1015
Processeur	Intel Inside CORE Duo 2.2 GHz 2.2 GHz
RAM	3.0 Go DDR
Cache	1 Go
Disque Dur	320 Go
OS	Windows 7 Professionnel

PCNN.

Pour l'interface de l'application, nous avons opté pour GDI+ et OpenGL. GDI+ est la bibliothèque graphique utilisée dans le .NET Framework assez simple mais qui se prête parfaitement à nos besoins. Cette librairie nous offre les mécanismes nécessaires à l'affichage de l'environnement, à sa modification à la volée ; le tout pour une consommation de processeur et de mémoire négligeable.

5.3 Configuration du matériel

La configuration matérielle est nécessaire si nous voulons nous faire une idée de la masse de calcul nécessaire à la génération d'un chemin. Il est de notoriété publique que les réseaux de neurones classiques sont très gourmands en temps de calcul. A la base de notre configuration, chacun peut extrapoler le temps de calcul que cela prendrait sur des plateformes plus ou moins récentes que la notre.

Le tableau 5.1 résume l'environnement dans lequel a été effectuée la simulation et à partir duquel nous avons obtenu nos résultats lors de l'expérimentation.

5.4 Configuration paramétrique

Dans cette section sera relatée la sensibilité de notre modèle PCNN au différents paramètres. Par sensibilité de ce modèle, le réseau neural utilisé pour les tests expérimentaux est constitué de 30 x 30 neurones et chaque neurone représente un disque ou une position du robot sur l'environnement. Les paramètres du réseau neuronal sont initialisés : $A_{init} = 1, A^r = 1, A^l = \sqrt{2}$; pour la fonction de threshold et $B = 10, C = 100$, pour l'activité interne dynamique. Avant le test, nous supposons que la position initiale du robot est connu à l'avance.

5.5 Tests dans différents environnements

5.5.1 Dans un environnement connu

Le premier test est appliqué sur une surface connue où la cartographie est représentée par des disques illustrée dans la figure(5.1.a). Les obstacles sont représentés par la couleur grise tandis que la position initiale, actuelle et finale du robot aspirateur sont représentées respectivement par les couleurs rouge, bleu et vert. Le réseau neuronal contient 30 x 30 neurones et chaque neurone représente un disque sur l'environnement. Les paramètres du réseau neuronal sont initialisés : $A_{init} = 1, A^r = 1, A^l = \sqrt{2}$, pour la fonction de threshold et $B = 10, C = 100$, pour l'activité interne dynamique. Avant le test, nous supposons que la position initiale du robot et l'emplacement des obstacles dans l'environnement sont connus à l'avance.

La simulation des résultats est illustrée dans la figure5.1, où le robot commence son nettoyage à partir de sa position de départ $S(1, 1)$ et termine sa tâche à la position finale $F(28, 1)$. Durant son déplacement, le neurone de départ s'active en premier et envoie ses ondes à l'aide du système d'activation vers un seule neurone ; ce dernier est sélectionné parmi d'autres voisins par l'équation 4.7. Cependant, le robot quitte sa position actuelle et se déplace vers la nouvelle position trouvée précédemment. Nous avons représenté ce déplacement réalisé par le modèle PCNN entre ces deux positions par une fine ligne rouge dans la figure 5.1.
Après un certain moment, ce nouveau neurone deviendra le neurone qui représente la position actuelle du robot et s'active aussi en même temps. Le système d'activation continue la diffusion ou la propagation des ondes en suivant les mêmes étapes effectuées entre ces derniers neurones jusqu'à l'activation de dernier neurone qui représente la dernière position non encore nettoyée où l'événement s'arrête et le robot s'arrête aussi $F(28, 1)$.

A partir de la figure 5.1, nous remarquons que le robot a tracé un chemin en zigzag, du bas-en-haut et du haut-en-bas couvrant ainsi toutes les cellules qui n'ont pas été nettoyées de la surface avec évitement des obstacles. Ce chemin généré est illustré dans la figure5.1.b où nous remarquons aussi que le robot a fait des déplacements entre plusieurs positions dans différents temps d'activation. Pour confirmer nos résultats, nous avons représenté aussi sur la figure 5.1.c en $3D$ la fonction de threshold θ_i landscape de réseau neuronal à la position $D(6, 17)$; et cela ; à partir de la position de départ du robot $S(1, 1)$. Il faut noter que la fonction de threshold θ_i de chaque cellule nettoyée a une grande valeur et est représentée par un pique, contrairement au restes des cellules qui gardent une valeur constante.

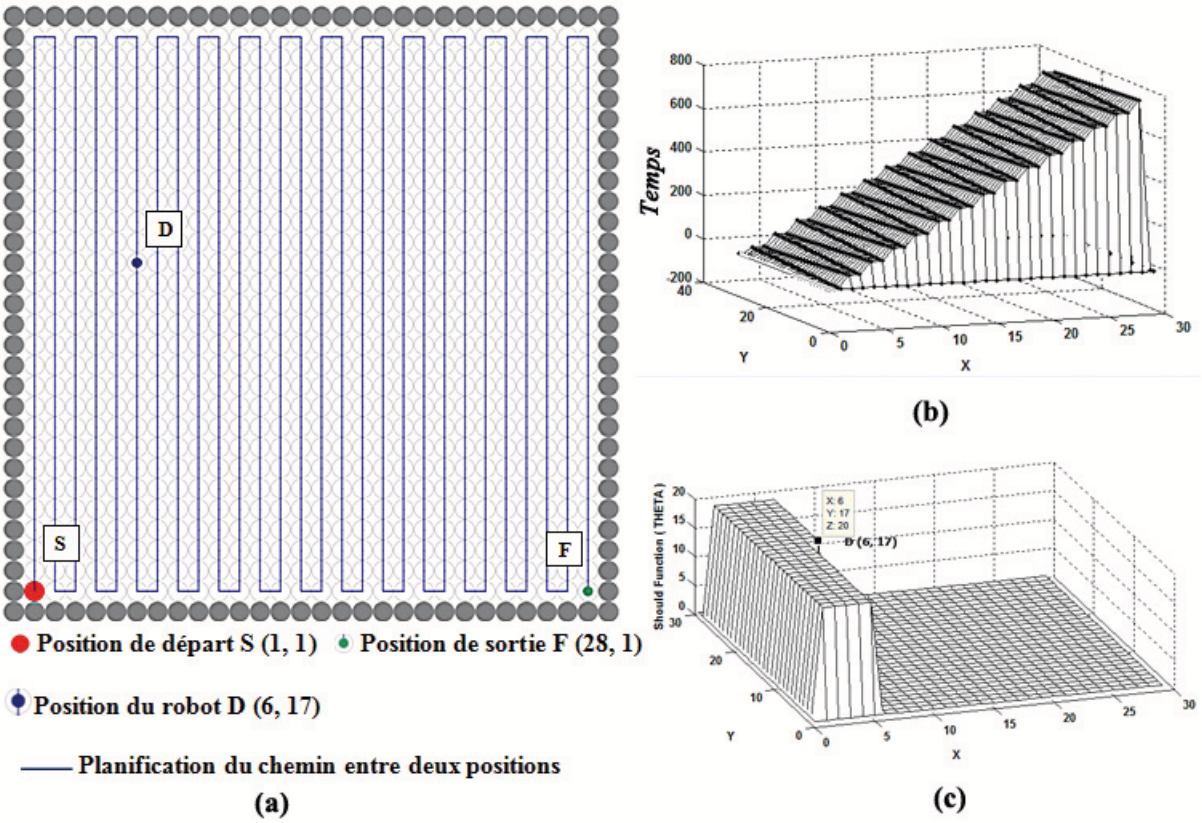


FIGURE *5.1* – Couverture complète d'une surface connue (a) Chemin planifié généré (b) Temps d'activation pour la planification du chemin par le robot (c) Landscape de la fonction neuronale should θ lorsque le robot atteint $D(6, 17)$.

5.5.2 Dans un environnement inconnu

Contrairement au premier test, avant de commencer, le robot ne dispose d'aucune information sur l'état de l'environnement qui est un environnement inconnu. L'information lui ait fournie par différents systèmes de détection des obstacles. La collection de ces informations obtenues à partir des différents capteurs utilisés et installés sur le robot aspirateur (comme cameras, infrarouges, ultrason ...etc), aident à construire une cartographie locale de la surface à nettoyer.

Dans ce test, le rayon du capteur installé sur le robot est limité à une portée de $R = 6$ comme illustré dans la figure5.2.a.

Le réseau neuronal contient 30 x 30 neurones avec les mêmes paramètres utilisés précédemment : $A_{init} = 1, A^r = 1, A^l = \sqrt{2}$, pour la fonction de threshold et $B = 10, C = 100$, pour l'activité interne dynamique.
Dans cette simulation, nous remarquons dans la figure5.2.a que lorsque le robot atteint la position $D(6, 17)$, ses capteurs peuvent obtenir un cercle qui contient un certain nombre

de cellules nettoyées, non nettoyées et des obstacles. Ce que confirme la figure 5.2.b qui représente le chemin généré par le robot depuis sa position initiale $S(1,1)$ jusqu'à sa position actuelle $D(6,17)$.
La fonction de threshold θ_i landscape du réseau neuronal (à la position $D(6,17)$ et cela à partir de la position de départ du robot aspirateur $S(1,1)$) est représentée dans la figure 5.2.c. Il faut noter que la fonction de threshold θ_i de chaque cellule nettoyée a une grande valeur et est représentée par un pique, tandis que le reste des cellules gardent une valeur constante.

Si nous faisons une comparaison avec le cas précédent de l'environnement connu (illustré dans la figure 5.1.a), nous remarquons que l'information sur toutes les cellules est connue à l'avance même lorsque le robot atteint la position $D(6,17)$ par contre dans la figure 5.2.a nous remarquons que le reste des cellules de la surface n'est pas encore connu parceque le robot dans ce cas n'a pas encore eu l'information ni exploré son environnent.

A partir du second test, nous constatons que le robot a appliqué le modèle PCNN où il a tracé ou planifié un chemin en zigzag entre une position de départ $S(1,1)$ et une position de sortie $F(28,1)$ avec évitement des obstacles détectés par ses capteurs. Ce système de détection des obstacles a vraiment aidé à créer une bonne cartographie de la surface et a aussi aidé le robot à visiter et nettoyer tout l'environnent en évitant ces obstacles ; ce que confirme la fonction de threshold θ_i landscape de réseau neuronal dans la figure 5.3.c où les quatre points de chute représentent les obstacles trouvés sur la surface tandis que les piques représentent les cellules nettoyées par le robot.

5.5.3 Dans un environnement dynamique

Dans cette section, nous testons la capacité du robot à naviguer et créer une très bonne cartographie d'une surface dynamique.
Dans ce cas, la situation est très compliquée par la simulation surtout au moment où le robot navigue et ce en lui modifiant la nature ou et la position de quelques obstacles.

Dans cette simulation, le réseau neuronal conserve la même topologie et paramètres utilisés précédemment. Nous remarquons que le robot démarre le nettoyage à partir de la position $S(1,1)$ en traçant un chemin en zigzag avec évitement de quatre obstacles statiques. Après un certain temps, le robot arrive à la position $J(23,15)$ où aucun déplacement ni modification d'obstacles n'ont été effectués comme illustré dans la figure 5.4.a et 5.4.b.
Le robot continue son déplacement normalement. Lorsque il atteint la position $H(23,15)$ des obstacles apparaissent soudainement sur sa trajectoire comme représenté dans la figure

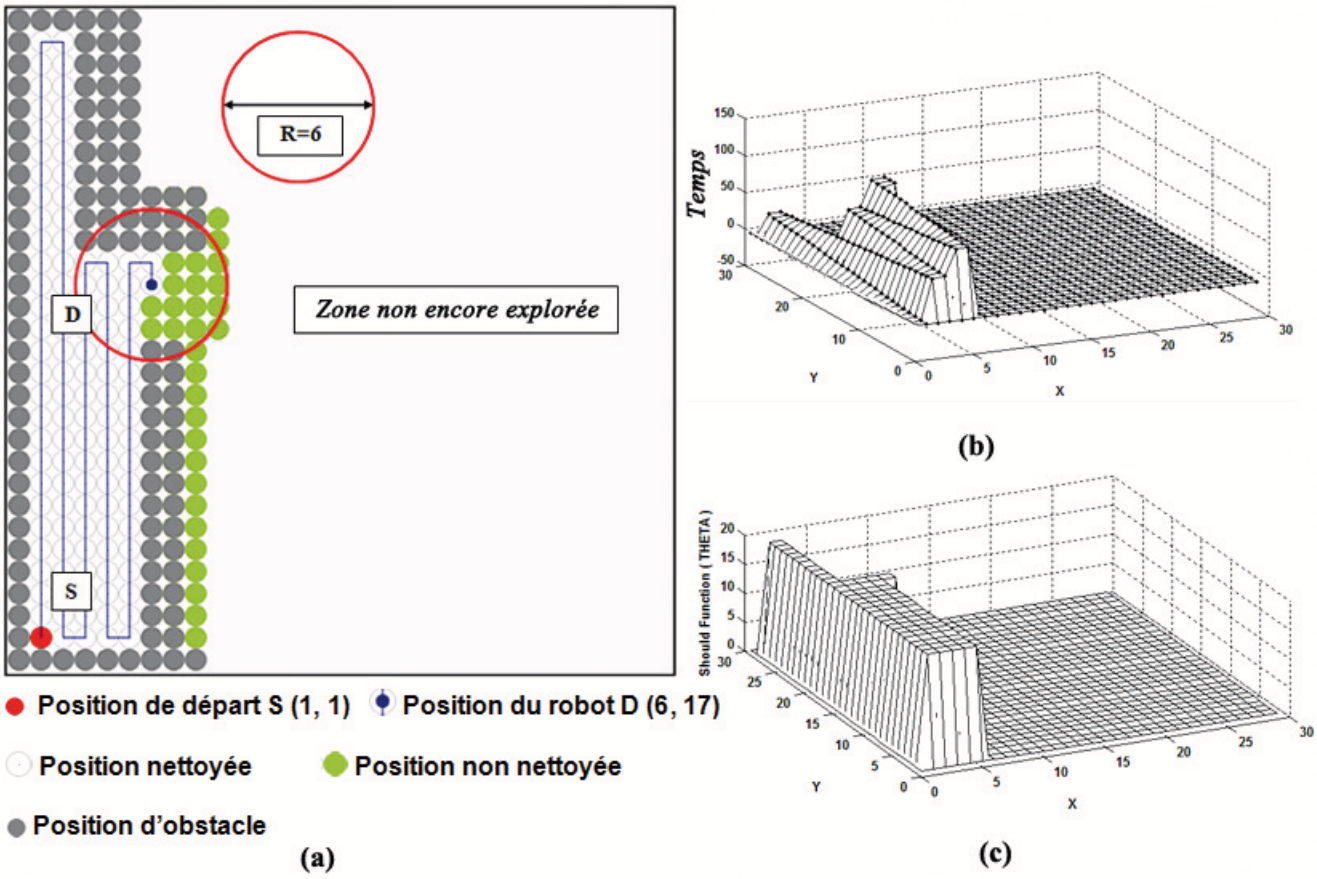


FIGURE 5.2 – Lorsque le robot atteint la position $D(6, 17)$ dans un environnement inconnu (a) Chemin planifié généré (b) Temps d'activation pour la planification du chemin par le robot (c) Landscape de la fonction neuronale should θ.

5.6.a. L'information qui met à jour la cartographie et révèle la présence de ces obstacles sur la surface est obtenue grâce aux capteurs du robot. Dans ce cas, le robot va générer un nouveau chemin qui lui permet de dévier à droite de sa trajectoire initiale au lieu de continuer tout droit jusqu'à la dernière position $F(28, 1)$ (figure 5.6.a).
Tout cela confirme la fiabilité de ce système d'activation du modèle PCNN lorsque les obstacles changent leur position durant le déplacement du robot. Nous pouvons également prouver cette fiabilité par le statut d'activité du neurone à la position libre $(24, 15)$ qui est en cadence contrairement au statut du neurone à la position $(23, 16)$ qui est statique (à zéro) comme illustré dans les figures 5.5.a et figure 5.5.b.

La fonction de threshold θ_i landscape du réseau neuronal dans la figure 5.6.b représente et confirme l'action qu'a pris le robot durant le nettoyage par le changement de la direction de son déplacement d'une position à une autre lorsque des obstacles surgissent devant lui.

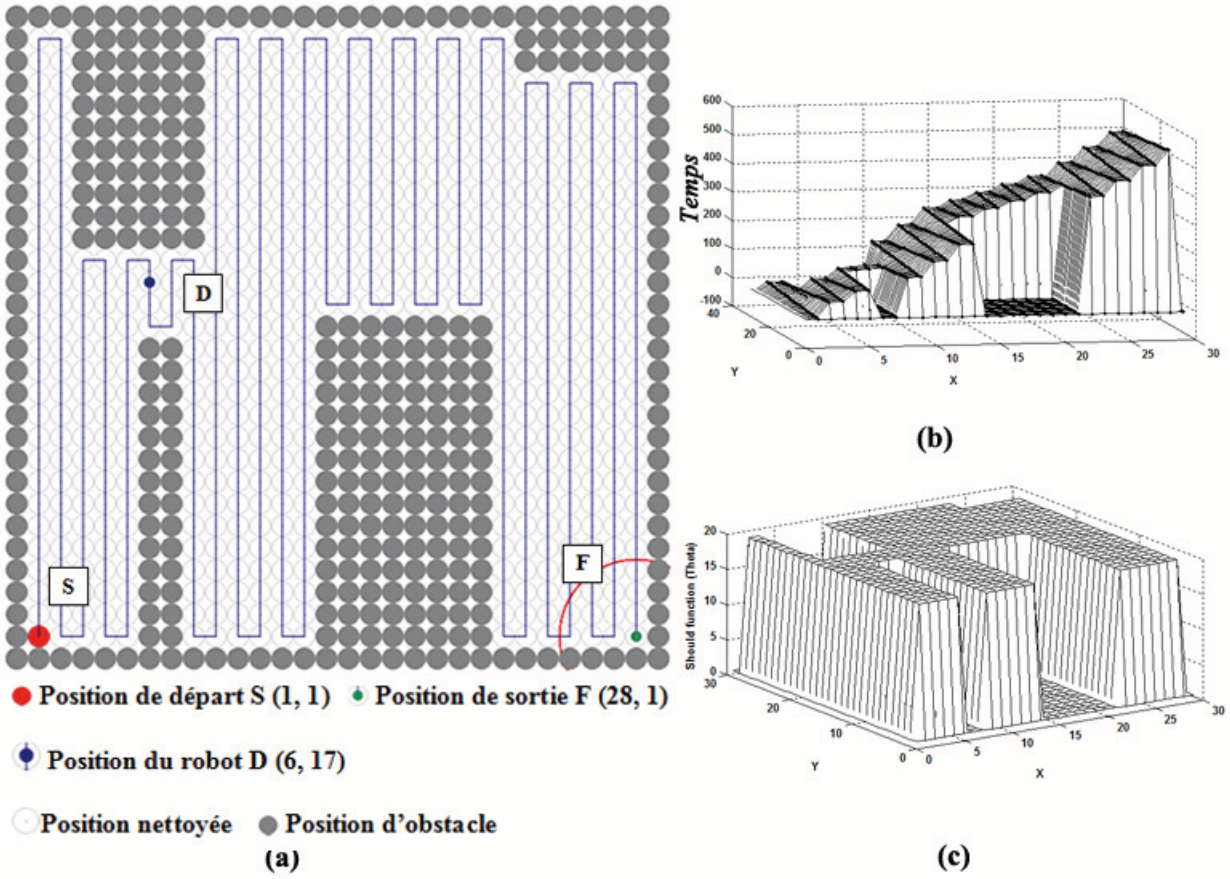


FIGURE 5.3 – Couverture complète d'un environnement inconnu (a) Chemin planifié généré (b) Temps d'activation pour la planification du chemin par le robot (c) Landscape de la fonction neuronale should θ.

5.6 Comparaison avec autres méthodes

Pour illustrer l'efficacité et la fiabilité de notre algorithme proposé et le modèle PCNN pour la couverture complète d'un environnement (dans le but de nettoyage par le robot aspirateur), nous nous intéressons à la comparaison de notre approche [97] avec différentes méthodes qui existent dans la littérature.

Pour cela, nous avons choisi quelques méthodes comme LSSP [100], BA* [99], CAC [98] et le modèle de réseau neuronal proposé par Luo et Yang [96].

Notre application ainsi que les autres méthodes sont codées en langage *C#* et testés dans un même environnement simulé et décomposé en plusieurs disques.

Pour chaque simulation, nous avons mesuré quatre paramètres : l'angle de braquage, les cellules revisitées, le temps de CPU et la longueur du chemin planifié par le robot pendant le nettoyage de la surface.

— **Angle de braquage :** Il représente le nombre de tours fais par le robot pendant

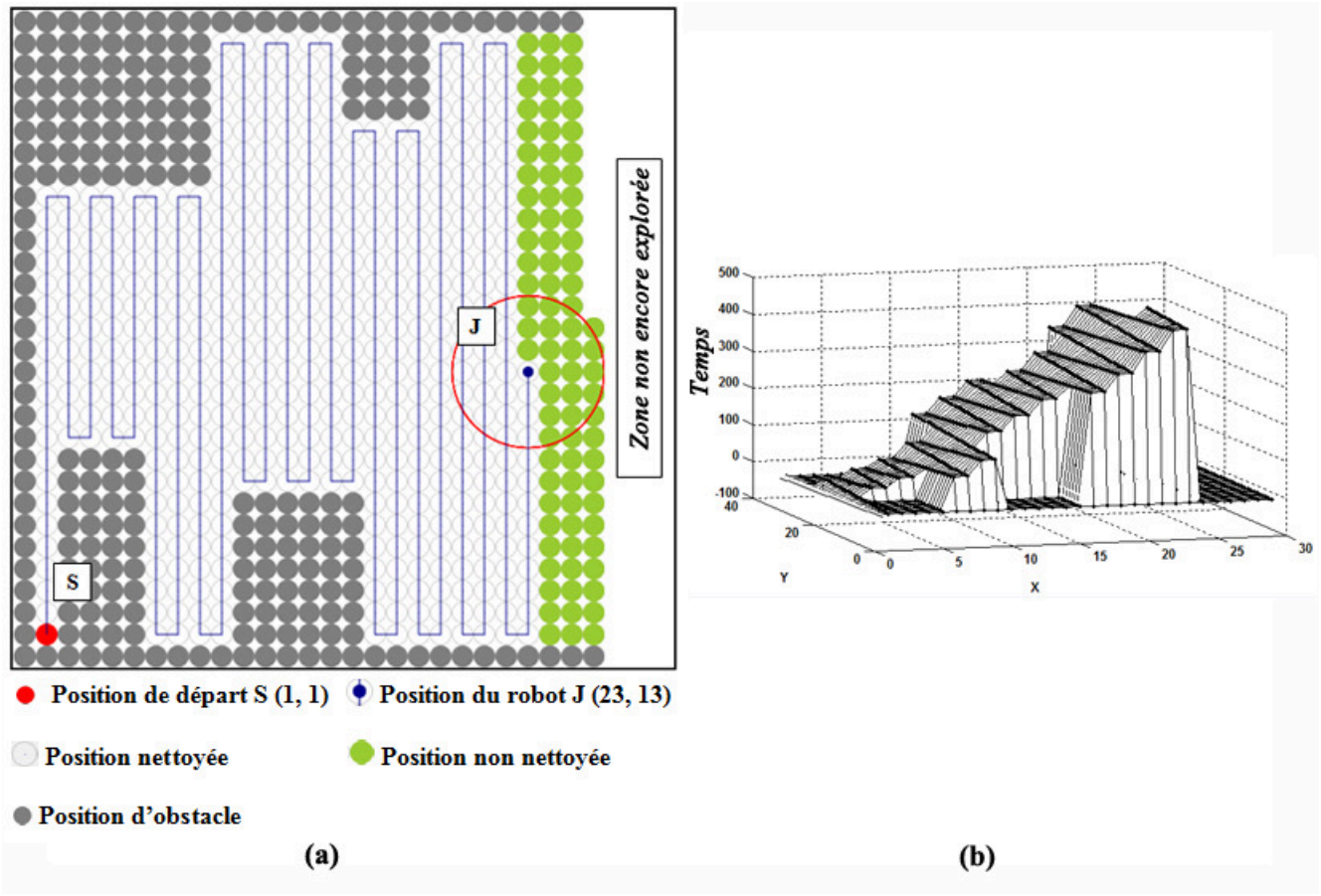


FIGURE 5.4 – Lorsque le robot atteint la position $J(23, 13)$ dans un environnement dynamique (a) Chemin planifié généré (b) Temps d'activation pour la planification du chemin par le robot.

son déplacement d'une position à une autre.

— **Cellules revisitées :** elles représentent le nombre de cellules déjà nettoyées et visitées plusieurs fois par le robot.

— **Temps de CPU :** Il représente le temps que met le programme du robot en microseconde pour accomplir sa tâche de nettoyage du robot d'une surface.

— **Longueur du chemin :** Elle représente la longueur de la trajectoire planifiée par le robot. Nous pouvons la calculer par la somme des différentes distances euclidiennes entre les cellules nettoyées.

5.6.1 Comparaison avec les algorithmes LSSP, BA* et CAC

Avant d'entamer la comparaison, nous allons clarifier et expliquer les déférentes méthodes que nous avons choisi pour la comparaison.

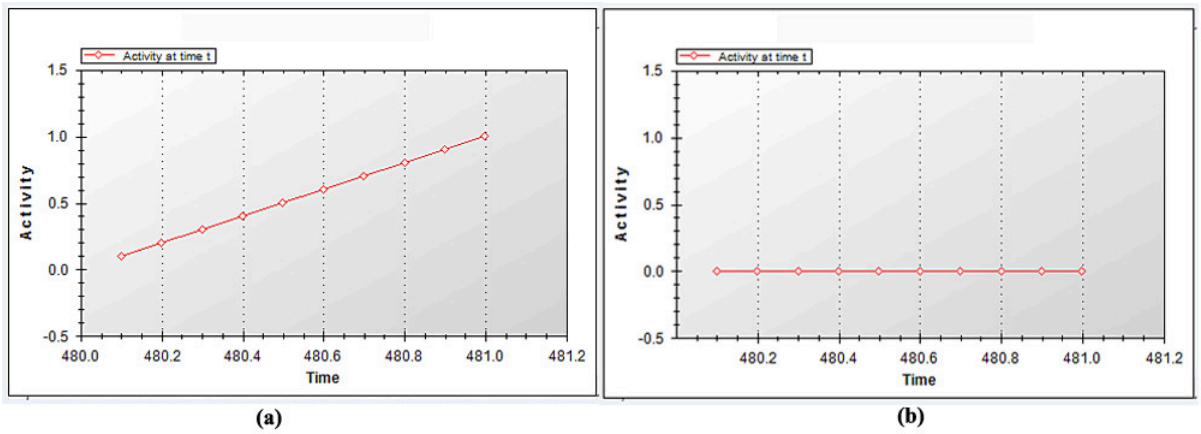


FIGURE 5.5 – L'activité neuronale lorsque le robot évite le nouveau obstacle dans. (a) position (24, 15) (b) position d'obstacle (23, 16).

5.6.1.1 Algorithme LSSP

Dans [100] Lee et al. ont proposé une méthode qui aide le robot à planifier son chemin de couverture en spirale. Cette méthode est basée sur une représentation cartographique grille de haute résolution et utilise un modèle de courbe cardinal spline pour générer un chemin spirale lisse et continu. Les chemins sont liés par la méthode "Constrained Inverse Distance Transform" (CIDT) [51] pour garantir la continuité de la couverture comme illustrée dans la Figure 5.7

5.6.1.2 Algorithme BA*

Dans [99] Viet et al. ont proposé aussi un algorithme pour une couverture complète d'une surface en zigzag. Cette méthode est basée sur deux procédés : un mouvement de boustrophédon unique et un algorithme A*.

Dans cette méthode, le robot effectue un mouvement de boustrophédon unique pour couvrir des cellules non encore visitées jusqu'à ce qu'il atteigne un point critique. Pour continuer à couvrir la prochaine cellule non encore visitée, le robot détecte les points de traçage en fonction de ses informations accumulées et détermine le meilleur point de backtracking comme le point de départ pour le prochain mouvement boustrophédon, et il applique un mécanisme de retour arrière (backtracking) basé sur l'algorithme A* de façon par venir sans collision au plus court chemin comme illustrée dans la Figure 5.8

5.6.1.3 Algorithme CAC

Dans [98] Jan et al. ont proposé un algorithme pour une couverture complète efficace d'un environnement en zigzag. C'est une méthode hybride basée sur la décomposition cellulaire verticale (VCD) et la méthode de coque convexe de cellule.

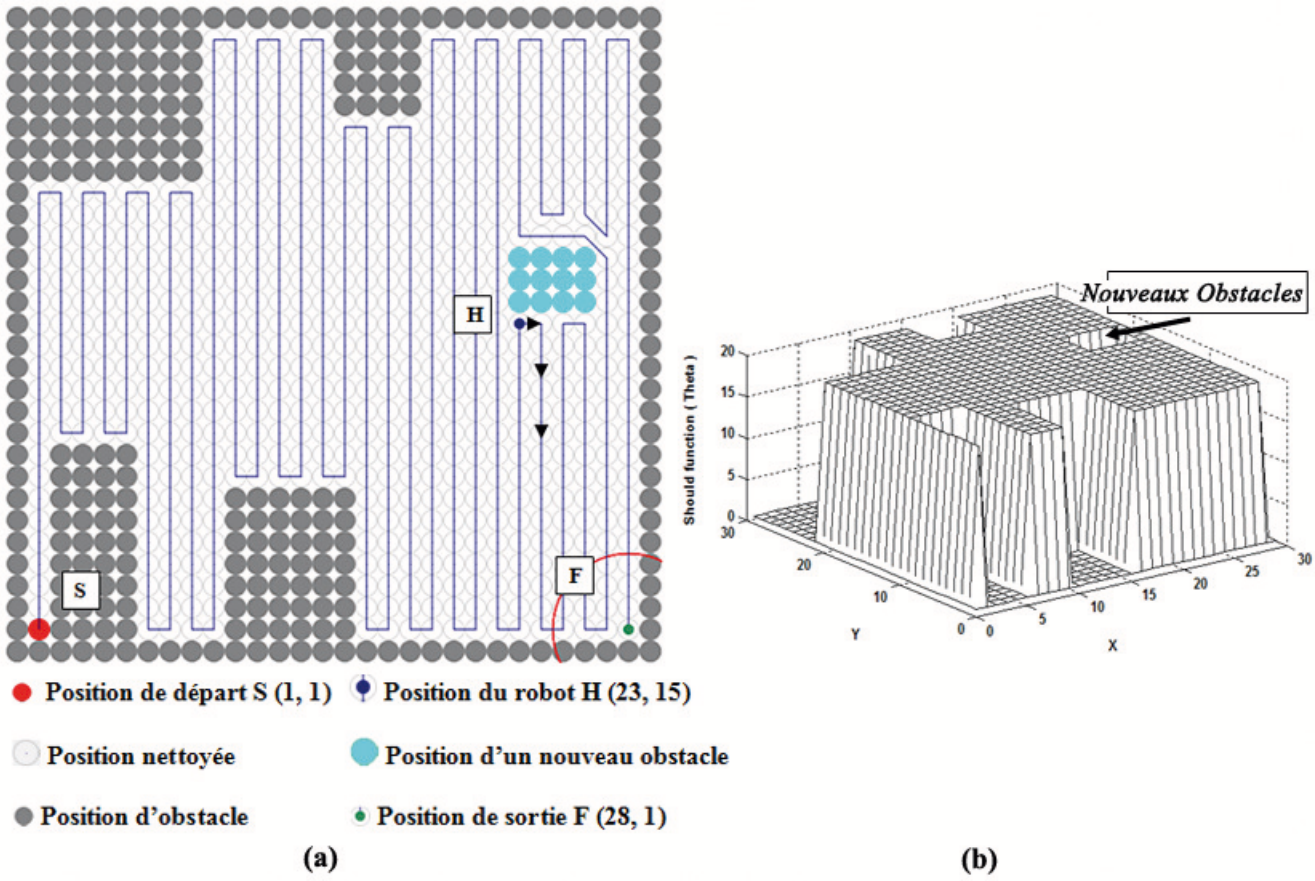


FIGURE 5.6 – Couverture complète d'un environnement dynamique (a) Chemin planifié généré (b) Landscape de la fonction neuronale should θ.

Le VCD s'applique pour décomposer la configuration initiale de l'environnement en sous régions, tandis que le coque convexe reconstruit le graphe adjacent.
Pour garantir le bon fonctionnement de l'algorithme CAC, Jan et al. ont émis des règles à suivre sous formes de 9 étapes commençant par l'initialisation de chaque point critique d'obstacles dans la configuration d'espace de travail et finissant par la génération de l'algorithme CAC détaillé comme illustrée dans la Figure5.9

5.6.1.4 Test de comparaison

Nous présentons à présent une comparaison de notre modèle proposé avec les algorithmes décrits précédemment LSSP, BA* et CAC.

Ces quatre méthodes sont appliquées pour balayer toutes les cellules non encore nettoyées d'une surface à condition qu'ils ne s'agissent qu'une seule surface qu'il y ait un même point de départ $S(1, 1)$ ainsi que les obstacles soient au même emplacement.
Dans la figure5.10.d, après que le robot ait utilisé l'algorithme LSSP et qu'il arrive à la position $A(27, 9)$ où il sera bloqué. Nous pouvons designer cette position : le point centrale

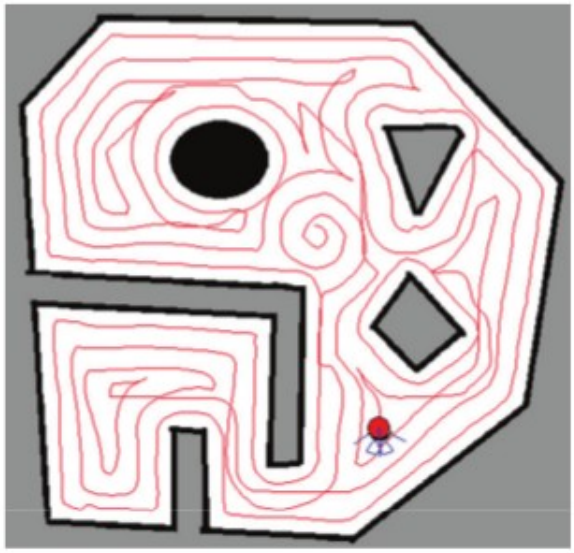


FIGURE 5.7 – Couverture complète par l'algorithme LSSP.

finale de la planification spirale. Dans ce cas, le robot doit trouver une sortie vers une cellule non encore nettoyée ; pour cela, il utilise une méthode qu'on appelle *CIDT* [51]. Cette méthode est utilisée pour que le robot se déplace vers une autre position non encore visitée ou nettoyée. Ce petit chemin est représenté par un tiret rouge.
Le même cas est arrivé au robot quand il a utilisé l'algorithme BA* sur la même position $A(27, 9)$; mais, il utilise une autre méthode ou mécanisme de retour en arrière pour sortir de son blocage vers une nouvelle position non encore nettoyée comme illustré dans la figure5.10.c.
Ensuite, la procédure de chaque méthodes précédentes est exécutée une autre foie pour compléter le nettoyage du reste de la surface.

Dans la figure5.10.b, le robot utilise l'algorithme CAC qui est basé sur la décomposition verticale des cellules pour accomplir aussi sa tâche de nettoyage. Notre approche qui a également été simulée aussi dans le même environnement avec la conservation de la topologie et les paramètres du réseau neuronal utilisés dans les sections précédentes. Le chemin généré par notre approche est illustré dans la figure 5.10.a.

A partir de deux figures 5.10.c et 5.10.d, comparées le chemin généré dans les deux autres figures 5.10.a et 5.10.b, nous remarquons que le robot est bloqué plusieurs fois aux positions A, B, C, G et K ; ceci a provoqué un chemin généré qui contient plusieurs cellules nettoyées et revisitées déjà par le robot .
Pour évaluer la performance de ces quatre méthodes dans la planification d'un chemin et résoudre le problème de couverture complète d'une surface, nous avons obtenu des résultats tels que groupés dans le tableau 5.2.

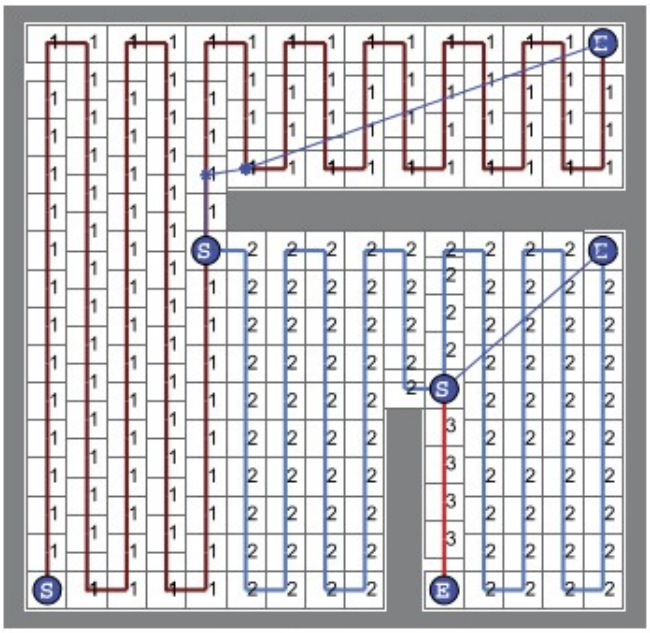


FIGURE *5.8* – Couverture complète par l'algorithme BA*.

Cette performance a été évaluée pour les quatre algorithmes où nous avons calculé la longueur complète de la trajectoire, le nombre des cellules revisitées ainsi que le nombre de tours.
Le tableau 5.2 montre que l'algorithme LSSP a eu de mauvais résultats parce qu'il avait une longue distance de 628.33*unités* par rapport aux autres algorithmes. Par contre l'algorithme BA* a effectué un petit nombre de tours d'angle (99) mais il a plusieurs fois revisité les cellules qui ont été déjà nettoyées ; ce qui a conduit automatiquement à augmenter la longueur de la trajectoire planifiée par rapport aux deux autres algorithmes CAC et notre approche proposée.
Il est vrai que l'algorithme CAC a effectué le cours chemin ; mais, il a trop dévié (à gauche et à droite) effectuant un très grand nombre de tours (222) par rapport à notre méthode qui a effectué moins nombre de tours (103 tours et 600.87*unités* de distance).
A partir de cette comparaison, nous ne pouvons pas conclure qu'un chemin ou un algorithme est efficace par rapport aux autres si nous ne les comparons que sur un seul paramètre. C'est pour cela que nous avons pris en considération tous les paramètres mentionnés dans le tableau 5.2 pour l'évaluation comparative.

Il faut noter qu'à partir des expérimentations et résultats du tableau 5.2, notre méthode *(modèle PCNN)* a montré une petite différence par rapport au meilleurs résultats des 03 autres méthodes si nous comparons chaque paramètre à part. Mais, si nous prenons en considération tous ces paramètres ensemble, notre méthode produit par rapport aux autres méthodes un chemin plus efficace pour la couverture complète d'une surface en

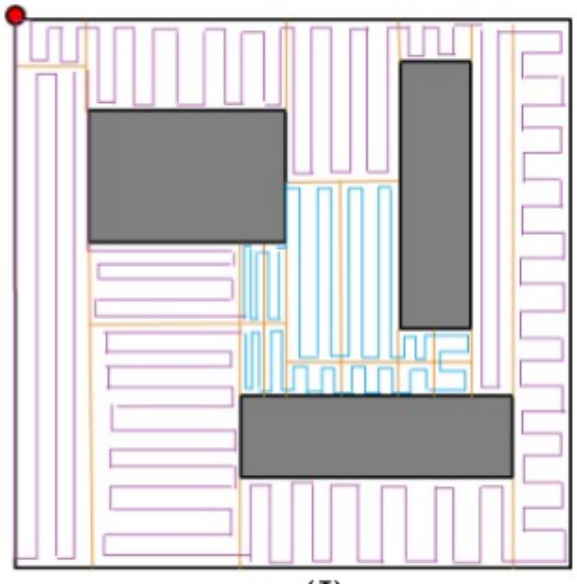


FIGURE *5.9* – Couverture complète par l'algorithme CAC.

TABLE *5.2* – Comparaison entre les algorithmes LSSP, BA*, CAC et la méthode proposée pour la couverture complète illustrée sur la figure 5.10

Modèle	**Cellules revisitées**	**Tours**	**Longueur**
Algorithme LSSP	25	108	628.33
Algorithme BA*	35	99	615.51
Algorithme CAC	0	222	597.00
Méthode proposée	0	103	600.87

minimisant la longueur de la trajectoire, le nombre de tours et les cellules revisitées. Nous pouvons déduire que ces paramètres ont un impact fort sur la couverture complète d'un environnement ; ce qui permet de réaliser cette tâche dans un court délai de temps et avec une petite consommation d'énergie (batteries du robot).

5.6.2 Comparaison avec la méthode Luo et Yang

Une autre méthode candidate à la comparaison présentée dans cette thèse est celle de *LuoetYang*. Pour rappel, le modèle de seuillage "shunting" de Luo Yang peut être décrit par une équation de shunting suivante :

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+\right) - (D + x_i)[I_i]^- \tag{5.1}$$

où tous ces paramètres sont définis dans [96].

La méthode proposée décrite dans les sections précédentes a les mêmes paramètres que dans les cas précédents. La simulation de ces deux méthodes (celle de Luo et la notre) est réalisée dans le même environnement i.e la même position de départ $S(1,1)$ et la même topologie du réseau de neurones qui a 30 x 30 neurones.

Les 02 robots prennent le même chemin en zigzag lors de l'application des deux méthodes comme représenté sur la figure 5.11.a et la figure 5.11.b. jusqu'à ce qu'ils arrivent à la position $D(18,1)$ où ils se ploquent parce qu'ils soient retrouvés des cellules voisines nettoyées soient devenues des obstacles.

Dans cette situation de blocage, les 02 robots doivent se déplacer à travers certaines cellules déjà visitées pour arriver à une nouvelle cellule qui n'est pas encore nettoyée. Pour cela, Luo Yang et ont proposé une solution qui est exprimée dans [96] où l'activité neuronale du neurone à la position $D(18,1)$ est plus grande que les autres activités neuronales de ses positions voisines et cela ne permet pas au robot de se déplacer vers une autre cellule. Pour sortir de cette situation, il faut attendre un certain temps pour que l'activité neuronale baisse à zéro dans la position $D(18,1)$. Ensuite, les activités des cellules non encore nettoyées se propagent dans toutes les directions. Après un certain temps, l'activité au niveau de la position $D(18,1)$ deviendra plus petite et le robot va recommencer à se déplacer vers le bas c-à-d vers la nouvelle position non encore nettoyée comme le montre la figure 5.11.b qui représente la trajectoire générée par le robot à partir de sa dernière position $D(18,1)$ vers la nouvelle cellule non encore nettoyée trouvée $H(9,9)$ avec des lignes en pointillés rouges.

Dans la figure 5.11.a nous avons appliqué notre méthode pour résoudre le même problème précédent où notre robot s'est bloqué. Le robot doit effectivement trouver une sortie vers une nouvelle cellule non encore nettoyée à partir de la position $D(18,1)$. Pour cela, le même modèle de PCNN avec ses algorithmes utilisés dans [90] sont appliquées.
Dans ce cas, pour trouver le chemin le plus court entre une position $D(18,1)$ et une nouvelle cellule non encore nettoyée, le modèle PCNN doit produire des auto-ondes à partir de la position $D(18,1)$. L'auto-ondes se propagent à travers les neurones voisins du robot jusqu'à ce qu'elle rencontre la première cellule non encore nettoyées dans l'environnement. La planification de cette courte trajectoire générée entre la position $D(18,1)$ et la première positon non encore nettoyée $H(9,9)$ trouvée depuis cet événement est représentée par des lignes en pointillés rouges dans la figure 5.11.a.

Après avoir trouvé la première cellule non encore nettoyée, notre robot va revenir à la

situation précédente où il continue sa trajectoire selon les algorithmes présentés au section 4.5.2 jusqu'à ce que toutes les cellules de l'ensemble de la surface soient nettoyées.

Si nous voulons faire une comparaison entre ces deux méthodes sur la manière avec lesquelles a été résolu le problème de la planification de chemin pour une couverture complète d'une surface, certains résultats tels que le nombre de cellules revisitées, le nombre de rotation, la longueur du trajectoire parcourue par les 02 robots et l'unité de traitement centrale (CPU) sont résumés dans le Tableau5.3.

Nous constatons qu'il n'y a pas une grande différence entre les deux méthodes sur le nombre de cellules revisitées, le nombre de rotation et la longueur de distance parcourue par les 02 robots. Par contre, le temps CPU nécessaire pour propager l'information de la position de départ vers la position de sortie selon Yang et al. est de 144.52 millisecondes alors qu' avec la méthode du modèle PCNN (appliquée par nous) est de 89,33 millisecondes ; ce qui signifie que le modèle PCNN par rapport au modèle Luo et Yang a certains avantages dans le temps de calcul pour propager l'événement d'activation et pour trouver rapidement le chemin le plus court. Ils faut signaler que les programmes de ces 02 méthodes ont été codées en *C#* et ont été effectués sur DELL Intel Core 2 Duo, 2,20 GHz, 2,20 GHz et 3 Go de RAM.

TABLE *5.3* – Comparaison entre le modèle Luo et Yang avec Approche proposée dans la couverture complète illustrée sur la figure 5.11

Modèle	**Cellules répétées**	**Tours**	**Longueur**	**Temps CPU**
Modèle Luo et Yang	23	107	644.15	144.52
Approche proposée	21	103	640.15	89.33

5.7 Conclusion

Dans ce chapitre, nous avons traité de la mise à l'épreuve du modèle PCNN que nous avons adopté. Nous avons présenté la configuration et la plateforme matérielle utilisée. Nous avons aussi exprimé les raisons de nos choix quant au choix du langage de programmation et le système d'exploitation.

Nous avons procédé à plusieurs tests distincts. Le premier de ces tests porté sur un environnement connu à l'avance c-à-d avant le début du nettoyage, le robot doit avoir toutes les informations sur les positons des obstacles situés dans l'environnement.
Le second test a porté sur un environnement inconnu, dans ce cas, le robot utilise ses différents capteurs installés autour de lui dont le but d'avoir des informations sur l'envi-

ronnement pour un nettoyage en temps réel.
le troisième test a porté sur un environnement dynamique où le robot aspirateur peut trouver un nouvel obstacle durant son nettoyage à n'importe quelle moment sur la surface. Ce test a démontré l'efficacité de modèle PCNN dans sa réaction d'évitement lors de sa confrontation avec un obstacle apparait soudainement.

A partir de ces 03 tests, nous pensons avoir démontré l'efficacité et la robustesse de ce modèle intégré dans notre robot pour atteindre une couverture complète dans un certain nombre d'environnements. De plus, nous avons comparé ce modèle avec certains méthodes existantes dans la littérature. Il mit en évidence certains avantages par la réduction de la longueur du chemin parcouru, le nombre de rotation et le nombre des cellules revisitées, ainsi que le temps calculé par le CPU.
Ces paramètres ont contribué à obtenir un temps plus court nécessaire à la planification du chemin pour la couverture complète d'une surface ainsi qu'une consommation plus faible d'énergie stockée (batteries) avec laquelle le robot aspirateur fonctionne [97].

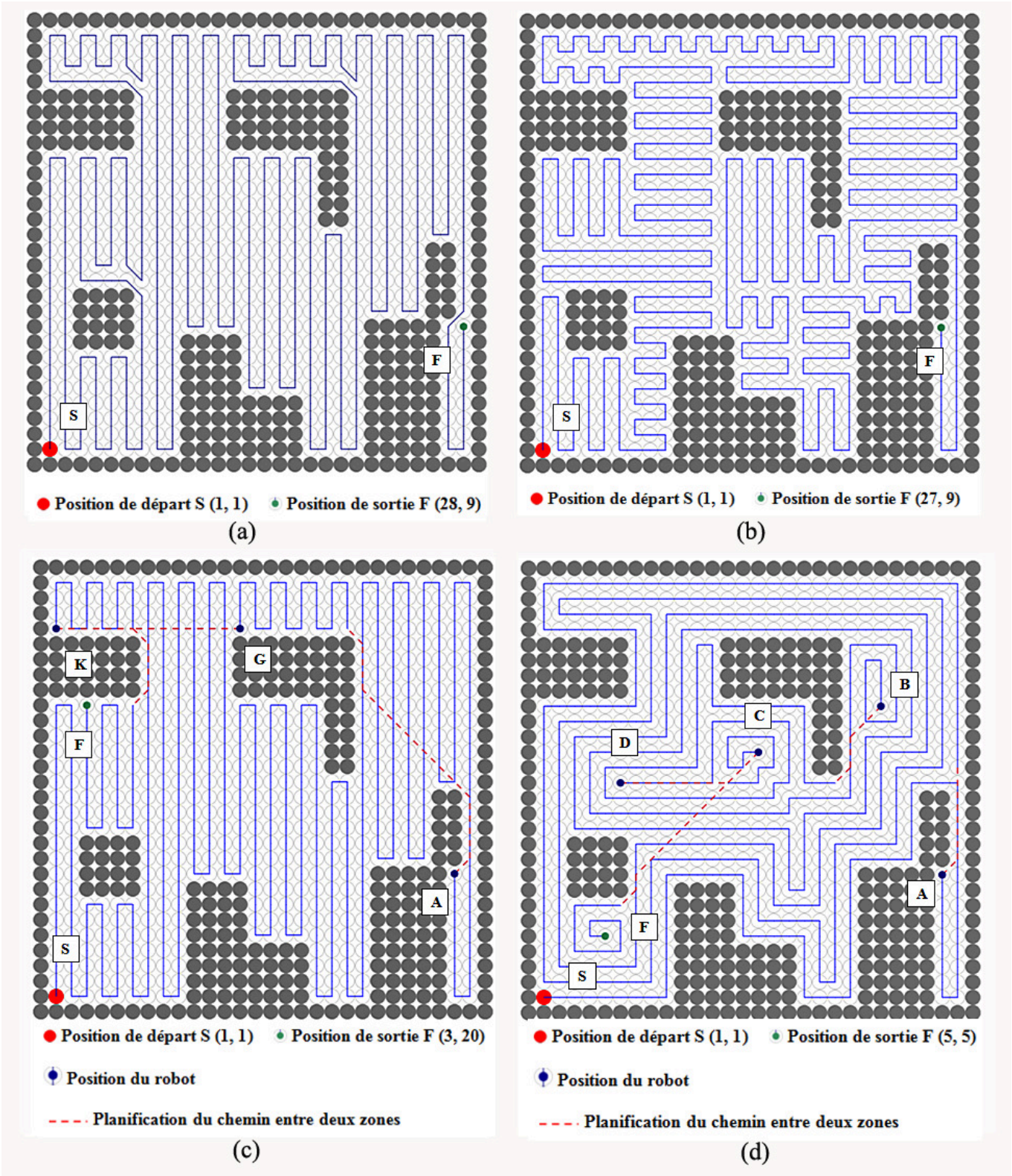


FIGURE *5.10* – Les cas de couverture complète d'un environnement inconnu en utilisant (a)Méthode proposée. (b)Algorithme CAC. (c)Algorithme BA*. (d)Algorithme LSSP.

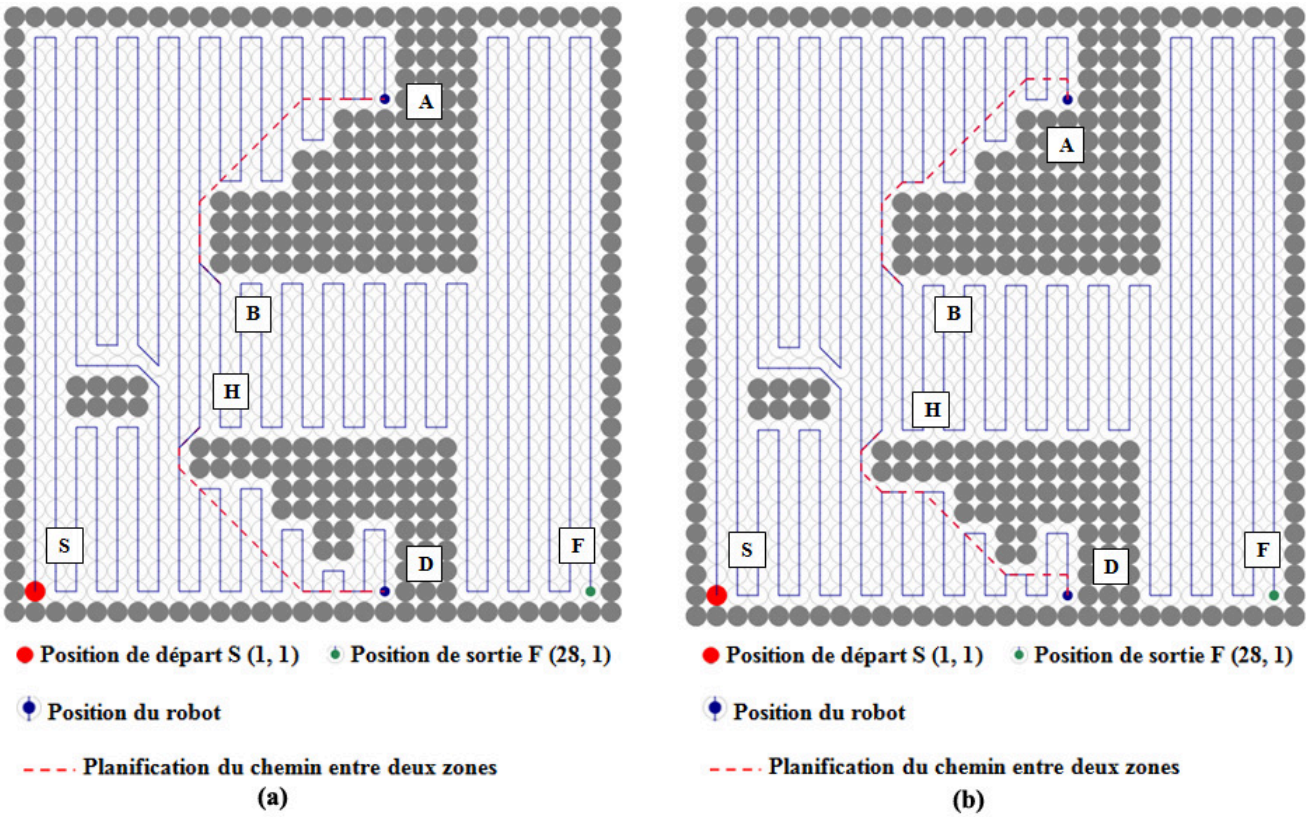


FIGURE *5.11* – Les cas de couverture complète d'un environnement inconnu en utilisant (a)Approche proposée. (b)Modèle Luo et Yang.

Chapitre 6
Conclusion Générale

6.1 Conclusion générale

L'évolution humaine et les exigences et les besoins qui en découlent ont permis à l'ensemble des champs de réflexion consacrés à l'amélioration constante de la qualité et du cadre de vie de l'homme. Depuis les temps les plus reculés, elles ont favorisé et boosté la recherche multidisciplinaire dans le sens d'alléger au maximum les tâches et actes de la vie multidimensionnelle quotidienne des individus et des sociétés. En un mot, l'homme a toujours cherché à faire faire et exécuter ses propres actes nécessaires aussi bien à sa survie qu'à son confort matériel voire moral. Cela a permis d'ouvrir, en grand et en continu, la recherche-développement dans tous les domaines et en particulier dans les domaines de l'automatisation, la robotique et la cybernétique.

Ce domaine est encore très actif, mais en ce moment, l'intelligence artificielle appliquée aux robots reste encore très loin derrière l'intelligence humaine. Encore une fois, il faut donc adopter des méthodes et des approches spéciales à ce contexte particulier et qui répondent aux exigences et spécificités de cette intelligence. La mise en œuvre réelle et pratique de cette intelligence, surtout dans des missions difficiles et coûteuses, fait de la simulation le moyen le plus largement utilisé pour la conception et l'évaluation des solutions proposées. La problématique traitée dans notre thèse concerne la modélisation et la simulation d'un robot aspirateur pour le nettoyage dans un environnement dynamique et ce par une méthode neuronale. Le concept de cette méthode est basée sur la planification du chemin de ce robot avec évitement des obstacles et raccourcissement de la durée de la mission et la longueur de la distance.
Dans cette conclusion générale, nous avons fait le point sur le cadre scientifique international et national des travaux de cette thèse. Nous avons dressé ainsi, le bilan des travaux effectués et des systèmes réalisés pour terminer par des perspectives qu'ouvrent ces réalisations.
Et pour contribuer modestement à cet immense mouvement ininterrompu en vue d'apporter quelques pistes théoriques pouvant ouvrir la capitalisation utile de ses résultats.

6.2 Résumé des parties traitées tout au long de cette thèse

Nous nous sommes investis dans cette thèse dans un domaine de recherche d'actualité, à savoir les robots aspirateurs et leur apport au nettoyage des environnements dynamiques. Les travaux de recherches portent sur la planification du chemin. La collision et le coût de ce chemin dans ce type de planification et spécifiquement dans la couverture complète d'un environnement dynamique, est une problématique très intéressante par sa

dimension humaine, économique et autres. De ce fait, nous avons introduit des modèles de simulations pour réaliser la mission de nettoyage.

Dans le cadre de cette thèse, nous nous sommes intéressés aux problématiques de planification de chemins liés à la couverture complète d'un environnement dynamique. L'objectif principal consiste à proposer une nouvelle méthode neuronale basée sur un modèle nommé réseau neuronal à impulsions couplées (PCCN) ; ce qui nous permet d'estimer un nombre important de situations de risques de collision. Cela signifie qu'elle va assurer un nettoyage complet de toutes les zones accessibles par le robot aspirateur d'un environnement *(Couverture Complète)*.

Dans un premier temps, nous avons abordé l'intelligence artificielle et ses applications d'une manière générale : le contexte, le principe de fonctionnement, les services et leurs domaines d'applications(soit des services associés aux systèmes de l'industrie - bras manipulateurs, chariots ...etc -, soit des services liés à la vie quotidienne d'un être humain). L'état de l'art réalisé s'est focalisé particulièrement sur la robotique mobile où l'intelligence artificielle diffère d'un robot à un autre et des divers domaines robotiques d'application existants.
Nous avons ainsi pu construire notre problématique qui est basée sur le nettoyage d'une pièce par un robot aspirateur mobile.

Nous avons également regroupé un état de l'art très important autour de la planification de chemins pour la couverture complète d'un environnement dynamique.
Cet état de l'art s'est étendu aux travaux et aux modèles utilisés pour traiter la problématique de notre thèse.
Nous avons, de cette façon, atteint l'objectif d'ouverture et de flexibilité recherché dans le cas de notre solution, en choisissant une méthode neuronale avec un modèle organisationnel nommé PCNN. Ce modèle nous offre la possibilité de mettre à jour le contrôle des robots selon la tâche qu'ils seront amenés à exécuter.

Nous avons présenté notre méthode portant sur la conception et la modélisation de notre modèle (PCNN). Cette conception est basée sur trois modélisation : la première est basée sur la modélisation du robot aspirateur de forme circulaire, comportant 4 capteurs et muni de trois roues (une roue motrice directrice et 02 autres roues fixes) ; la deuxième est basée sur la modélisation de l'environnement en disque de diamètre 10 unités en forme circulaire comme le robot. Cela nous permet d'envisager des obstacles aux formes plus extravagantes et d'obtenir la facilité du déplacement de robot dans cet environnement. La dernière et troisième modélisation est basée essentiellement sur l'architecture du modèle PCNN où ce modèle est constitué de plusieurs neurones, chaque neurone représentant soit

un obstacle soit une positon libre occupée le robot dans l'environnement. Ainsi, ce modèle réalise la mission de nettoyage du robot aspirateur.

La méthode proposée a été validée par plusieurs tests de simulations où ces tests ont été appliqués dans différents états d'environnement : un environnement sans obstacles, un environnement complètement connu avant le démarrage de l'opération de nettoyage, un environnement sans aucune information sur l'emplacement des obstacles avant le démarrage, (le robot a besoin de ses capteurs pour avoir cette information), et un autre type d'environnement rendu compliqué (dynamique). Dans ce dernier type, nous avons remarqué la capacité du robot d'éviter à n'importe quel moment les obstacles lors son déplacement.
Une autre analyse comparative avec d'autres travaux récents réalisées dans le même domaine a été effectuée dont les objectifs sont l'efficacité et la robustesse de notre algorithme et modèle proposés. Les résultats obtenus démontrent une capacité élevée de PCNN dans la planification du chemin d'un robot aspirateur.

6.3 Perspectives

Ces travaux de thèse ont permis d'ouvrir le champ à des perspectives variées et étendues à l'image du thème de recherche abordé. Dans la continuité du travail de recherche présenté, nous pourrions approfondir notre étude afin d'améliorer les résultats obtenus. Plusieurs perspectives futures peuvent être proposées à la suite de ce travail :

- ☐ L'extension de l'environnement dynamique présenté par le rendre plus complexe qu'avant (un labyrinthe par exemple).
- ☐ La seconde perspective concerne l'extension de méthode proposée à tous les cas. (Fusion de deux méthodes par exemple).
- ☐ La troisième perspective concerne l'implémentation de la méthode sur le terrain (traduction pratique et matérialisée de la simulation utilisée pour notre robot).

Bibliographie

[1] Nils J Nilsson. Shakey the robot. Technical report, DTIC Document, 1984.

[2] Lounis Adouane. *Architectures de Contrôle Comportementales et Réactives pour la Coopération d'un groupe de Robots Mobiles*. PhD thesis, Université de Franche-Comté, 2005.

[3] Nieves Pavón-Pulido, Juan Antonio López-Riquelme, Joaquín Ferruz-Melero, Miguel Ángel Vega-Rodríguez, and Antonio José Barrios-León. A service robot for monitoring elderly people in the context of ambient assisted living. *Journal of Ambient Intelligence and Smart Environments*, 6(6) :595–621, 2014.

[4] Mingzhong Yan, Daqi Zhu, and Simon X Yang. Complete coverage path planning in an unknown underwater environment based on ds data fusion real-time map building. *International Journal of Distributed Sensor Networks*, 2012, 2012.

[5] Anqi Xu, Chatavut Viriyasuthee, and Ioannis Rekleitis. Efficient complete coverage of a known arbitrary environment with applications to aerial operations. *Autonomous Robots*, 36(4) :365–381, 2014.

[6] IA Hameed. Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain. *Journal of Intelligent & Robotic Systems*, 74(3-4) :965–983, 2014.

[7] Nicola Doering, Sandra Poeschl, Horst-Michael Gross, Andreas Bley, Christian Martin, and Hans-Joachim Boehme. User-centered design and evaluation of a mobile shopping robot. *International Journal of Social Robotics*, 7(2) :203–225, 2015.

[8] Kazi Mahmud Hasan, Khondker Jahid Reza, et al. Path planning algorithm development for autonomous vacuum cleaner robots. In *Informatics, Electronics & Vision (ICIEV), 2014 International Conference on*, pages 1–6. IEEE, 2014.

[9] Zuo Llang Cao, Yuyu Huang, and Ernest L Hall. Region filling operations with random obstacle avoidance for mobile robots. *Journal of Robotic systems*, 5(2) :87–102, 1988.

[10] Esther M Arkin and Refael Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3) :197–218, 1994.

[11] Esther M Arkin, Sándor P Fekete, and Joseph SB Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1) :25–50, 2000.

[12] M LaValle. Chapter 4. planning algorithms, 2006.

[13] Thomas C Shermer. Recent results in art galleries [geometry]. *Proceedings of the IEEE*, 80(9) :1384–1399, 1992.

[14] Fajie Li and Reinhard Klette. *An approximate algorithm for solving the watchman route problem*. Springer, 2008.

[15] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4) :113–126, 2001.

[16] Vladimir J Lumelsky, Snehasis Mukhopadhyay, and Sun Kang. Dynamic path planning in sensor-based terrain acquisition. *IEEE Transactions on Robotics and Automation*, 6(4) :462–472, 1990.

[17] H Choset and P Pignon. Cover path planning : The boustrophedron decomposition. In *Proceedings International Conference on Field and Service Robotics, Canberra, Australia*, 1997.

[18] Ercan U Acar, Howie Choset, Alfred A Rizzi, Prasad N Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21(4) :331–344, 2002.

[19] Jean-Claude Latombe. Robot motion planning (the kluwer international series in engineering and computer science). 1990.

[20] Howie M Choset. *Principles of robot motion : theory, algorithms, and implementation*. MIT press, 2005.

[21] Timo Oksanen and Arto Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8) :651–668, 2009.

[22] Howie Choset, Ercan Acar, Alfred Rizzi, Jonathan Luntz, et al. Exact cellular decompositions in terms of critical points of morse functions. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2270–2277. IEEE, 2000.

[23] John Willard Milnor. *Morse theory*. Number 51. Princeton university press, 1963.

[24] Ercan U Acar and Howie Choset. Robust sensor-based coverage of unstructured environments. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 61–68. IEEE, 2001.

[25] Ercan U Acar and Howie Choset. Sensor-based coverage of unknown environments : Incremental construction of morse decompositions. *The International Journal of Robotics Research*, 21(4) :345–366, 2002.

[26] John Canny. Constructing roadmaps of semi-algebraic sets i : Completeness. *Artificial Intelligence*, 37(1) :203–222, 1988.

[27] John F Canny and Ming C Lin. An opportunistic global path planner. *Algorithmica*, 10(2-4) :102–120, 1993.

[28] Howie Choset. Coverage of known spaces : The boustrophedon cellular decomposition. *Autonomous Robots*, 9(3) :247–253, 2000.

[29] Michael Bosse, Navid Nourani-Vatani, and Jonathan Roberts. Coverage algorithms for an under-actuated car-like vehicle in an uncertain environment. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 698–703. IEEE, 2007.

[30] Ercan U Acar and Howie Choset. Critical point sensing in unknown environments. In *ICRA*, pages 3803–3810, 2000.

[31] Georges Reeb. Sur les points singuliers d'une forme de pfaff completement intégrable ou d'une fonction numérique. *CR Acad. Sci. Paris*, 222 :847–849, 1946.

[32] Elena Garcia and P Gonzalez De Santos. Mobile-robot navigation with complete coverage of unstructured environments. *Robotics and Autonomous Systems*, 46(4) :195–204, 2004.

[33] Ercan U Acar, Howie Choset, and Prasad N Atkar. Complete sensor-based coverage with extended-range detectors : A hierarchical decomposition in terms of critical points and voronoi diagrams. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 3, pages 1305–1311. IEEE, 2001.

[34] Ercan U Acar, Howie Choset, and Ji Yeong Lee. Sensor-based coverage with extended range detectors. *Robotics, IEEE Transactions on*, 22(1) :189–198, 2006.

[35] Howie Choset and Joel Burdick. Sensor-based exploration : The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, 19(2) :96–125, 2000.

[36] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. *Computational geometry*. Springer, 2000.

[37] Howie Choset and David Kortenkamp. Path planning and control for free-flying inspection robot in space. *Journal of Aerospace Engineering*, 12(2) :74–81, 1999.

[38] Sylvia C Wong, Bruce MacDonald, et al. A topological coverage algorithm for mobile robots. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1685–1690. IEEE, 2003.

[39] Sylvia Wong. *Qualitative topological coverage of unknown environments by mobile robots*. PhD thesis, ResearchSpace@ Auckland, 2006.

[40] Zack J Butler, Alfred Rizzi, Ralph L Hollis, et al. Contact sensor-based coverage of rectilinear environments. In *Intelligent Control/Intelligent Systems and Semiotics, 1999. Proceedings of the 1999 IEEE International Symposium on*, pages 266–271. IEEE, 1999.

[41] Hans P Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121. IEEE, 1985.

[42] Alberto Elfes. Sonar-based real-world mapping and navigation. *Robotics and Automation, IEEE Journal of*, 3(3) :249–265, 1987.

[43] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1) :21–71, 1998.

[44] José A Castellanos, Juan D Tardós, and Günther Schmidt. Building a global map of the environment of a mobile robot : The importance of correlations. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1053–1059. IEEE, 1997.

[45] Sebastian Thrun et al. Robotic mapping : A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

[46] Joon Seop Oh, Yoon Ho Choi, Jin Bae Park, and Yuan F Zheng. Complete coverage navigation of cleaning robots using triangular-cell-based map. *Industrial Electronics, IEEE Transactions on*, 51(3) :718–726, 2004.

[47] Alexander Zelinsky, Ray A Jarvis, JC Byrne, and Shin'ichi Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of international conference on advanced robotics*, volume 13, pages 533–538, 1993.

[48] Vikas Shivashankar, Rajiv Jain, Ugur Kuter, and Dana S Nau. Real-time planning for covering an initially-unknown spatial environment. In *FLAIRS Conference*, 2011.

[49] Yoav Gabriely and Elon Rimon. Spiral-stc : An on-line coverage algorithm of grid environments by a mobile robot. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 954–960. IEEE, 2002.

[50] Enrique Gonzalez, Oscar Alvarez, Yul Diaz, Carlos Parra, and Cesar Bustacara. Bsa : a complete coverage algorithm. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2040–2044. IEEE, 2005.

[51] Young-Ho Choi, Tae-Kyeong Lee, Sang-Hoon Baek, and Se-Young Oh. Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5788–5793. IEEE, 2009.

[52] Chaomin Luo, Simon X Yang, Deborah Stacey, Jan C Jofriet, et al. A solution to vicinity problem of obstacles in complete coverage path planning. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 612–617. IEEE, 2002.

[53] Simon X Yang and Chaomin Luo. A neural network approach to complete coverage path planning. *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, 34(1) :718–724, 2004.

[54] Ming Zhong Yan and Da Qi Zhu. An algorithm of complete coverage path planning for autonomous underwater vehicles. In *Key Engineering Materials*, volume 467, pages 1377–1385. Trans Tech Publ, 2011.

[55] Xuena Qiu, Jiatao Song, Xuejun Zhang, and Shirong Liu. A complete coverage path planning method for mobile robot in uncertain environments. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 8892–8896. IEEE, 2006.

[56] Yi Guo and Mohanakrishnan Balakrishnan. Complete coverage control for nonholonomic mobile robots in dynamic environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1704–1709. IEEE, 2006.

[57] Liam Paull, Sajad Saeedi, Howard Li, and Vincent Myers. An information gain based adaptive path planning method for an autonomous underwater vehicle using sidescan sonar. In *Automation Science and Engineering (CASE), 2010 IEEE Conference on*, pages 835–840. IEEE, 2010.

[58] Liam Paull, Saeed Saeedi, Mae Seto, and Huaqing Li. Sensor-driven online coverage planning for autonomous underwater vehicles. *Mechatronics, IEEE/ASME Transactions on*, 18(6) :1827–1838, 2013.

[59] Ling Xu. *Graph planning for environmental coverage*. PhD thesis, Microsoft Research, 2011.

[60] Wesley H Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 27–32. IEEE, 2001.

[61] Yakoubi Mohamed Amine and Laskri Mohamed Tayeb. The path planning of cleaner robot for coverage region using genetic algorithms. *Journal of Innovation in Digital Ecosystems Elseiver*, 3(1) :37–43, 2016.

[62] Mohamed Amine Yakoubi and Mohamed Tayeb Laskri. The path planning of cleaner robot for coverage region using genetic algorithms. In *International Conference on Pattern Analysis and Intelligent Systems PAIS*, pages 335–340. PAIS Tebessa, 2015.

[63] Paulo Jimenez, Bijan Shirinzadeh, Ann Nicholson, Gursel Alici, et al. Optimal area covering using genetic algorithms. In *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, pages 1–5. IEEE, 2007.

[64] Raphael Mannadiar and Ioannis Rekleitis. Optimal coverage of a known arbitrary environment. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5525–5530. IEEE, 2010.

[65] Anqi Xu, Chatavut Viriyasuthee, and Ioannis Rekleitis. Optimal complete terrain coverage using an unmanned aerial vehicle. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2513–2519. IEEE, 2011.

[66] Manuel Mazo Jr and Karl Henrik Johansson. Robust area coverage using hybrid control. *TELEC, Santiago de Cuba, Cuba*, 2004.

[67] Ercan U Acar and Howie Choset. Exploiting critical points to reduce positioning error for sensor-based navigation. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, pages 3831–3837. IEEE, 2002.

[68] Stephen Tully, George Kantor, and Howie Choset. Leap-frog path design for multi-robot cooperative localization. In *Field and service robotics*, pages 307–317. Springer, 2010.

[69] Ayoung Kim. Active visual slam with exploration for autonomous underwater navigation. Technical report, DTIC Document, 2012.

[70] Timothy Bretl and Seth Hutchinson. Robust coverage by a mobile robot of a planar workspace. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4582–4587. IEEE, 2013.

[71] R Eckhorn, HJ Reitboeck, Mt Arndt, and P Dicke. Feature linking via synchronization among distributed assemblies : Simulations of results from cat visual cortex. *Neural Computation*, 2(3) :293–307, 1990.

[72] HJ Reitboeck, R Eckhorn, M Arndt, and P Dicke. A model for feature linking via correlated neural activity. In *Synergetics of Cognition*, pages 112–125. Springer, 1990.

[73] Reinhard Eckhorn, HJ Reitbock, M Arndt, and P Dicke. A neural network for feature linking via synchronous activity : results from cat visual cortex and from simulations. 1989.

[74] Ilya A Rybak, Natalia A Shevtsova, Lubov N Podladchikova, and Alexander V Golovan. A visual cortex domain model and its use for visual information processing. *Neural Networks*, 4(1) :3–13, 1991.

[75] Ilya A Rybak, Natalia A Shevtsova, and Vladislav M Sandler. The model of a neural network visual preprocessor. *Neurocomputing*, 4(1) :93–102, 1992.

[76] Omid Omidvar and Judith Dayhoff. *Neural networks and pattern recognition*. Academic Press, 1997.

[77] John L Johnson. Pulse-coupled neural nets : translation, rotation, scale, distortion, and intensity signal invariance for images. *Applied Optics*, 33(26) :6239–6253, 1994.

[78] JL Johnson, ML Padgett, and O Omidvar. Overview of pulse coupled neural network (pcnn) special issue : Special issue on pulse coupled neural networks. *IEEE transactions on neural networks*, 10(3) :461–463, 1999.

[79] T Lindblad and JM Kinser. Image processing using pulse-coupled neural networks, secaucus, 2005.

[80] Zhaobin Wang, Shuai Wang, Ying Zhu, and Yide Ma. Review of image fusion based on pulse-coupled neural network. *Archives of Computational Methods in Engineering*, pages 1–13, 2015.

[81] Haiqing Wang, Changying Ji, Baoxing Gu, and Guangzhao Tian. A simplified pulse-coupled neural network for cucumber image segmentation. In *Computational and Information Sciences (ICCIS), 2010 International Conference on*, pages 1053–1057. IEEE, 2010.

[82] Liu Yang and Kang Lei. A new algorithm of image segmentation based on bidirectional search pulse-coupled neural network. In *Computational Aspects of Social Networks (CASoN), 2010 International Conference on*, pages 101–104. IEEE, 2010.

[83] Song Yin-Mao, Zhu Xiao-Hui, and Liu Guo-le. One segmentation algorithm of multi-target image based on improved pcnn. In *2010 2nd International Workshop on Intelligent Systems and Applications*, pages 1–4, 2010.

[84] TH Rava, Vineetha Bettaiah, and Heggere S Ranganath. Adaptive pulse coupled neural network parameters for image segmentation. *World Acad. Sci. Eng. Technol*, 73 :1046–1052, 2011.

[85] Ying Xiong, Wei-Hua Han, Kai Zhao, Yan-Bo Zhang, and Fu-Hua Yang. An analog cmos pulse coupled neural network for image segmentation. In *Solid-State and Integrated Circuit Technology (ICSICT), 2010 10th IEEE International Conference on*, pages 1883–1885. IEEE, 2010.

[86] Jianfeng Li, Beiji Zou, Lei Ding, and Xu Gao. Image segmentation with pcnn model and immune algorithm. *Journal of Computers*, 8(9) :2429–2436, 2013.

[87] Chao Gao, Dongguo Zhou, and Yongcai Guo. Automatic iterative algorithm for image segmentation using a modified pulse-coupled neural network. *Neurocomputing*, 119 :332–338, 2013.

[88] Dongguo Zhou, Chao Gao, and Yongcai Guo. A coarse-to-fine strategy for iterative segmentation using simplified pulse-coupled neural network. *Soft Computing*, 18(3) :557–570, 2014.

[89] Mohamed Amine Yakoubi and Mohamed Tayeb Laskri. Une approche evolutionniste pour la planification de chemin d'un robot dans un environnement dynamique. In *7th National proceeding of computer science Biskra. SNIB.*, pages 109–114. unknown, 2010.

[90] Hong Qu, Simon X Yang, Allan R Willms, and Zhang Yi. Real-time robot path planning based on a modified pulse-coupled neural network model. *Neural Networks, IEEE Transactions on*, 20(11) :1724–1739, 2009.

[91] Johann Borenstein and Yorem Koren. Real-time obstacle avoidance for fast mobile robots. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(5) :1179–1187, 1989.

[92] Robert Sim and James J Little. Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters. *Image and Vision Computing*, 27(1) :167–177, 2009.

[93] José Jesús Guerrero, Ruben Martinez-Cantin, and Carlos Sagüés. Visual map-less navigation based on homographies. *Journal of Robotic Systems*, 22(10) :569–581, 2005.

[94] Christophe Giovannangeli, Philippe Gaussier, and Gaël Désilles. Robust mapless outdoor vision-based navigation. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3293–3300. IEEE, 2006.

[95] Javier Ibañez-Guzman, Christian Laugier, John-David Yoder, and Sebastian Thrun. Autonomous driving : Context and state-of-the-art. In *Handbook of Intelligent Vehicles*, pages 1271–1310. Springer, 2012.

[96] Chaomin Luo and Simon X Yang. A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. *Neural Networks, IEEE Transactions on*, 19(7) :1279–1298, 2008.

[97] Mohamed Amine Yakoubi, Mohamed Tayeb Laskri, and Mohamed Nadjib Zennir. The complete coverage for the vacuum cleaner robot using pulse-coupled neural network in dynamic environments. *Journal of Ambient Intelligence and Smart Environments IOS Press*, not Published Yet, 2016.

[98] Gene Eu Jan, Chaomin Luo, Lun-Ping Hung, and Shao-Ting Shih. A computationally efficient complete area coverage algorithm for intelligent mobile robot navigation. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 961–966. IEEE, 2014.

[99] Hoang Huu Viet, Viet-Hung Dang, Md Nasir Uddin Laskar, and TaeChoong Chung. Ba* : an online complete coverage algorithm for cleaning robots. *Applied intelligence*, 39(2) :217–235, 2013.

[100] Tae-Kyeong Lee, Sang-Hoon Baek, Se-Young Oh, and Young-Ho Choi. Complete coverage algorithm based on linked smooth spiral paths for mobile robots. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 609–614. IEEE, 2010.