REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : IVA 15/M2/2021

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Image et Vie Artificielle (IVA)**

Informed RRT* pour la planification de chemin en environnement virtuel

---

Par :

SEKRAF ABDELAZIZ

Soutenu le ../../…. devant le jury composé de :

Nom Prénom	grade	Président
Mr.Boucette Mebarek	MAA	Rapporteur
Nom Prénom	grade	Examinateur

Année universitaire 2020-2021

Dédicace

Je dédie ce travail à :

Aux âmes des chers grand-père et grand-mère, que
Dieu leur fasse miséricorde

Mes chers parents ,qui nulle dédicace ne peut exprimer mes sincères sentiments Pour leur patience illimitée leur encouragement contenu , leur aide , en témoignage de mon profond amour et respect pour ses grandes sacrifices .

A ma chère épouse qui m'a aidé dans ce travail

Et à la prunelle de mes yeux : Fatima Al-Zahra et Maria Jouri, et à toutes mes sœursEt à mon cher frère : Al -HaChemi

Et à tous ceux que j'aime

Remerciement

Nous tenons à remercier en premier lieu dieu puissant qui nous a donné la force pour dépasser toutes nos années d'étude .

Nous exprimons toute gratitude à notre encadreur

MR /Boucette Mebark pour ses conseils pendront notre travail de fin d'étude .

Toutes nos expressions de gratitude et de reconnaissance vont vers les membres du jury .nos remerciements vont également à l'ensemble des enseignants du département d'informatique pour la formation qu'ils nous ont donné durant notre cycle d'étude .

En fin nous remercions avec toute la suprême sincérité tout ceux qui nous ont idée de prés ou de loin à terminer et réaliser notre projet de fin d'étude.

Table de matière

Tableau de figure

LISTE DE TABLEAU

Introduction Générale

La modélisation d'environnement sous forme de bases géométrique en trois dimension devient de plus en plus courante .Toute personne a déjà vu et ce depuis de nombreuses années ,un bâtiment ,une ville ,une maisons ou toutes autres formes architecturales modélisées en 3d ,et ce dans les films d'animation (pixar,disney),dans les jeux vidéo ou tout simplement dans les effets spéciaux de films récents .cette représentation en 3D est une aide précieuse aux métiers du génie civil ou aux du septième art.

Ces environnement virtuels sont apparus dans un premier temps dépourvus d'entités mobiles et purement géométrique .Puis sont apparus les premiers personnages virtuels créant ainsi une pseudo-vie au sein-même de ces environnement .Les premiers personnage n'étaient qu'animation, ils suivaient une trajectoire déterminés au préalable par leurs créateurs respectifs.

Au fur et à mesure ces personnage de sont dotés d'une certaine autonomie ,ils étaient désormais capables d'analyser brièvement le monde extérieur afin de choisir la meilleure stratégie à adopter .Puis dans un souci d'automatisation des procédés d'animation ,le problème posé fut de doter le entités dynamique d'un comportement plausible .Cette automatisation de l'animation a ensuite permis de tendre vers la simulation d'humanoïdes virtuel.

la planification consiste à déterminer une trajectoire en générale la meilleure que doit suivre une entité virtuel pour aller d'un point à un autre dans un environnement complexe (nombreux obstacles ,terrain de nature différente).On rencontre ce problème surtout en robotique mais également dans les jeux vidéo .Dans le cadre de ce travail nous nous sommes intéressés à la navigation de tels agents virtuels au sein de ces environnement virtuels .La planification de chemin est une tache cruciale pour ces agents puisqu'elle va influencer directement sur le réalisme du mouvement et la capacité à agir au sein de ces mondes. [7]

1 -Contexte de recherche

La planification de chemin est une problématique de recherche qui a été largement abordée dans le domaine de la robotique. Que ce soit pour des robots se déplaçant dans des environnements réels ou pour des agents se déplaçant dans des environnements virtuels, le fond de cette problématique reste le même. Dans les deux cas, l'entité (robot ou agent virtuel) doit construire ou posséder une représentation de l'environnement l'entourant à fin de pouvoir naviguer de manière autonome au sein de cet environnement. Cette capacité de navigation est cruciale puisqu'elle va permettre à un agent d'allerà la découverte de son environnement, augmentant ainsi ses possibilités d'action avec cet environnement ainsi qu'avec les autres agents qui le peuplent.

Ces deux domainesont donc, de manière similaire, cherchés à proposer des nouvelles solutions à la navigation autonome de l'entité. Toutefois, de nombreux points doivent être soulevés a fin de doter un agent de cette autonomie de déplacement. L'entité doit tout d'abord être capable d'appréhender sa présence et sa géométrie au sein de l'environnement et de pouvoir s'y localiser, répondant ainsi à la question "Où suis-je ?". La configuration de l'environnement doit ensuite être identifiée, a fin que l'entité déduise les accessibilités et obstructions existantes, répondant ainsi à la problématique "Où puis-je me rendreet comment ?".

Finalement l'agent va devoir être à même de trouver son chemin ausein de cet environnement, répondant à la question "Comment puis-je atteindre cette destination ?". Ces problématiques ont donc tout d'abord été traitées dans le domaine de la robotique depuis plusieurs décennies. Le développement récent, et de plus en plus important, des environnements virtuels a amené à adapter de nombreuses méthodes issues de la robotique à des agents virtuels et à rechercher de nouvelles solutions.[3]

Par leur utilisation dans des domaines très variés, les mondes virtuels vont présenter des propriétés très déférentes en fonction des applications visées. Cependant, le peuplement de ces environnements est un point crucial dans l'ensemble de ces applications puisqu'il va permettre de rendre ces mondes virtuels plus "vivants" et donc de créer par conséquent une meilleure immersion de l'utilisateur. Il est en effet très rare d'être en présence de lieux totalement vides dans nos environnements quotidiens et la création d'un monde virtuel peuplé renforce donc la crédibilité de celui-ci. Le peuplement de ces

environnements par des agents virtuels autonomes va appartenir au domaine de l'animation comportementale.

L'autonomie de ces agents va dépendre des diverses capacités qu'ils vont posséder. Parmi celles-ci, la capacité de naviguer de manière autonome est l'une des aptitudes les plus importantes. Cette aptitude va en effetinfluer directement sur l'autonomie de déplacement de ces agents au sein de l'environnement ainsi que sur leur capacité à aller agir avec celui-ci et avec les autres agents.

La navigation des agents virtuels est donc un centre d'intérêt commun à l'ensemble des domaines applicatifs utilisant des environnements virtuels. Toutefois, les besoins et les contraintes imposées ne sont pas les mêmes en fonctions des applications considérées. Ainsi, ces domaines applicatifs peuvent être divisés en deux parties : les applications interactives, telles que les jeux vidéo ou encore la réalité virtuelle, et les applications deproduction, telles que le cinéma d'animation ou la gestion de cycles de vie des produits. Dans le cadre des applications interactives, l'accent est mis sur les temps de réponse des méthodes proposées a fin qu'un utilisateur agissant avec l'environnement virtuel ait une réponse directe à ses actions.

À l'inverse, dans le cadre des méthodes de production, le résultat final sera en général produit hors-ligne à l'aide de méthodes coûteuses en temps de calcul. Toutefois, il est intéressant dans ce cadre applicatif de posséder des méthodes de prévisualisation rapides a fin d'obtenir à l'avance une idée du résultat final en économisant du temps de calcul.

2- Problématique :

Les méthodes de planification de chemin actuelles s'intéressent généralement à adresser le problème de la navigation pour une topologie particulière d'environnements. De nombreuses méthodes se sont ainsi intéressées à des agents au sein d'environnements La prise en compte d'environnements statiques va permettre de pré calculer de nombreux éléments nécessaires à la planification des agents puisque la topologie de l'environnement ne va pas être modifiée par la suite. Afin de s'ouvrir à un panel plus important d'environnements, de nombreuses méthodes de planification se sont ensuite focalisées sur la navigation au sein d'environnements changeants et dynamiques .

Toutefois les éléments dynamiques pris en compte par ces méthodes sont

généralement des obstacles que les agents doivent éviter au cours de leur navigation. Plus récemment, certaines méthodes proposé de gérer des environnements composés d'éléments dynamiques navigables mais elles font en contrepartie l'hypothèse que les mouvements des éléments dynamiques sont connus a priori a fin d'anticiper les accès créés. Ces simplifications limitent les possibilités d'applications de ces méthodes. En effet, l'évolution de la topologie d'un environnement n'est généralement pas connue àl'avance puisque cette évolution pourra être la conséquence de l'action d'un utilisateur ou le résultat d'une simulation physique. De plus, les objets dynamiques présents dansun environnement peuvent bien entendu se comporter comme des obstacles à la navigation de l'agent mais ces objets, tels que des plateformes mobiles, des planches ou encore des escabeaux, vont également pouvoir aider un agent au cours de sa navigation en lui proposant de nouvelles opportunités de navigation. Ensimplifiant le problème de navigation, les solutions actuelles ne permettent donc pas de gérer des environnements de ce type.

3-Organisation de ce mémoire

Dans Ce mémoire on va tout d'abord présenter une étude bibliographique du domaine de la planification de chemin. Cette étude va permettre d'introduire les bases de nos recherches mais également d'identifier les limitations des méthodes actuelles et de démontrer le besoin de nouvelles solutions.

Nous présenterons nos contributions en trois chapitres. Le premier s'intéressera à la représentation de l'environnement et la recherche de chemin et navigation . Dans cette partie nous définissons la planification de chemin et présenté les type de l'environnement et les éléments nécessaire pour chaque type de l'environnement.

Le chapitre suivant on fait la conception du système pour cela on détaille l'algorithme INFORMED RRT *.Le dernier chapitre est l'implémentation et résultats Nous expliquons les étapes de mise en œuvre de l'algorithme INFORMED RRT *RRT et les résultats obtenus.

Chapitre 1

Recherche de Chemin et Représentation de l'Environnement

1- Introduction à la planification de chemin

La problématique de la planification de chemin est définie ainsi : étant donné un robot possédant un nombre variable de degrés de liberté, et une description de l'environnement où ce robot est immergé, comment trouver un chemin libre de toutes collisions entre deux configurations du robot au sein de l'environnement. De nombreux exemples illustrant cette problématique peuvent être cités.

Le rôle de la planification va être par exemple de déterminer un chemin entre des positions dans un environnement. Il peut s'agir ainsi d'un robot virtuel cherchant son chemin dans un environnement .[6]

2-Qu'est-ce que la planification de chemin?

La planification de chemin est un problème classé dans le domaine de l' intelligence artificielle en informatique. Des nombreux ouvrages ont été écrit à ce sujet comme (Russell et Norvig, 20 10), (La Valle, 2006) et (Boulic, 201 0). Elle permet à un personnage, objet ou une unité de se déplacer dans un environnement en partant d' une configuration A (départ) et en se dirigeant vers une configuration B (destination) tout en évitant les obstacles présents dans cet environnement. L'objet doit rester dans une configuration valide durant toute la durée du trajet. L'environnement peut se retrouver modifié lors du déplacement de l'objet dépendamment du contexte.[5]

Le processus de la planification de chemin s'effectue en trois étapes :

- Discrétisation de l'espace continue en un espace discret.
- Recherche de chemin dans une espace discret trouvé
- Lissage d'un chemin obtenu.

3-les application de la recherche de chemin :

3-1jeu vidéo :

Les environnements virtuels sont depuis toujours très présents dans les jeux vidéo où le joueur doit généralement évoluer à l'aide d'un avatar le représentant.

Ces mondes virtuels sont généralement peuplés d'agents virtuels autonomes avec qui le joueur va pouvoir interagir et qui vont permettre, par leurs actions, d'enrichir le jeu. Grâce aux améliorations des performances des processeurs et des cartes graphiques, ces

environnements virtuels se complexifient afin d'être toujours plus riches, que ce soit par des rendus des décors toujours améliorés ou par l'intégration de plus en plus courante de moteurs de simulation physique.

Afin de planifier leur chemin au sein de tels environnement, les agents virtuels vont devoir être capables de réagir rapidement aux actions de l'utilisateur ou du moteur physique afin de prendre en compte les modifications topologiques créées et de proposer des interactions riches au joueur sans que celui-ci n'ait à attendre.[2]

3-2Robotique :

En robotique planifier un déplacement devient encore plus complexe. En effet, il s'agit de se déplacer dans un environnement réel. Tout d'abord, le robot ne dispose que d'une estimation de sa position, car ses capteurs ne sont pas parfaits.

De la même façon, il doit se déplacer au moyen d'effecteurs qui ne peuvent l'amener là où il décide qu'avec une certaine imprécision.[1]

3-3Le cinéma :

Le domaine cinématographique utilise depuis de nombreuses années les environnements virtuels afin de réaliser des films d'animation, des effets spéciaux s'intégrant au sein de scènes réelles ou encore à intégrer des acteurs virtuels (respectivement réels) dans des scènes réelles (respectivement virtuelles).

Certains films ont par exemple fait appel à ces techniques pour générer des armées d'acteurs virtuels s'affrontant sur des champs de bataille.[2]

4–Représentation de l'environnement :

Les environnements se caractérisent par leur structure qui n'est pas amenée à évoluer au cours du temps. Les méthodes proposées pour caractériser les environnements vont se diviser principalement en deux familles. D'un côté, les méthodes de décomposition en cellules vont représenter l'espace navigable à l'aide de zones interconnectées de formes prédéfinies permettant d'appréhender les limites de Obstacles.[5]

D'un autre côté, les méthodes utilisant les cartes de cheminement vont représenter un sous-ensemble de C-Free par un ensemble de chemins standardisés permettant la navigation de robot.

4-1Décomposition en cellules :

La décomposition en cellules propose de diviser l'espace des configurations en cellules de formes prédéfinies. Celles-ci sont ensuite caractérisées selon leur appartenance à Obstacle afin de construire une structure de données plus adaptée à la planification de chemin. Les méthodes de décomposition en cellules se répartissent en deux familles : les décompositions approximatives et les décompositions exactes. [5]

4-1-1 Décomposition approchées :

Les méthodes de décompositions approximatives proposent l'utilisation d'une forme prédéfinie de cellules afin de caractériser l'environnement . Ces cellules représentent des sous-ensembles de l'espace des configurations et sont interconnectées afin de capturer la connectivité de l'environnement. [4]

La représentation de l'environnement sous la forme d'une grille régulière 2D permet de projeter les obstacles de l'environnement dans une bitmap 2D et de pouvoir ainsi profiter de la carte graphique afin d'accélérer la planification de chemin lors des requêtes.

a) Grille uniforme :

La décomposition uniforme permet d'accéder rapidement à un point connaissant ses coordonnées. Cependant, elle souffre de certains défauts. Une cellule pouvant à la fois contenir un obstacle et une partie accessible, des informations utiles sont perdues. Ainsi, pour augmenter la précision, il faut diminuer la taille des cellules. En conséquence, la quantité de mémoire utilisée augmente de manière quadratique. Il est donc nécessaire de trouver un compromis sur la taille des cellules pour avoir une précision et une occupation mémoire satisfaisante.[7]

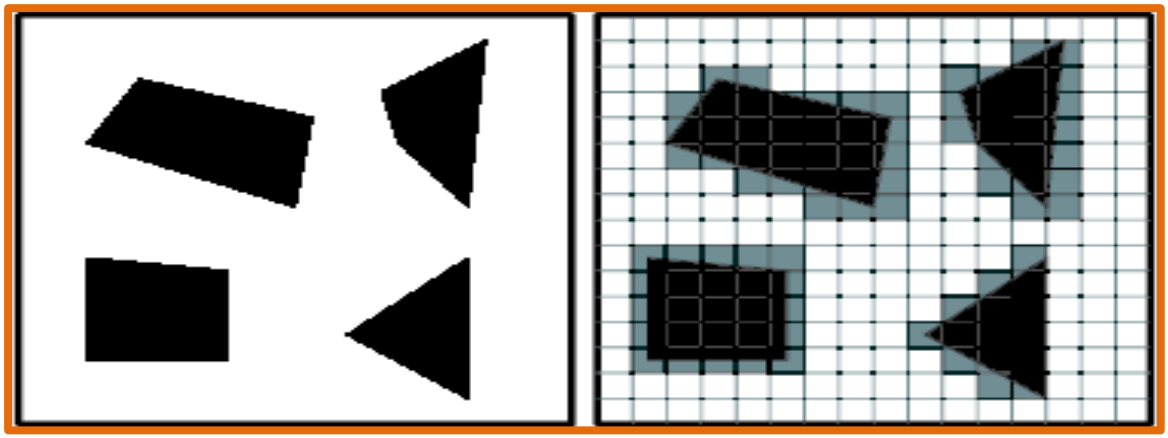


Figure1 :Grille uniforme

Pour réduire ce problème, une évolution de ce modèle est apparue sous la forme des grilles hiérarchiques.

b)*Grilles hiérarchiques :*

L'utilisation de grilles hiérarchiques consiste à découper l'espace grâce à un arbre de cellules. Les cellules est entièrement libres ou entièrement occupées sont conservée telle quelle. En revanche, les cellules mixtes sont de nouveau divisées. Le processus s'arrête quand les cellules ont atteintes une taille minimale. Cette technique permet donc une compression des données puisqu'elle ne rajoute de l'information qu'aux endroits où c'est utile.[7][14]

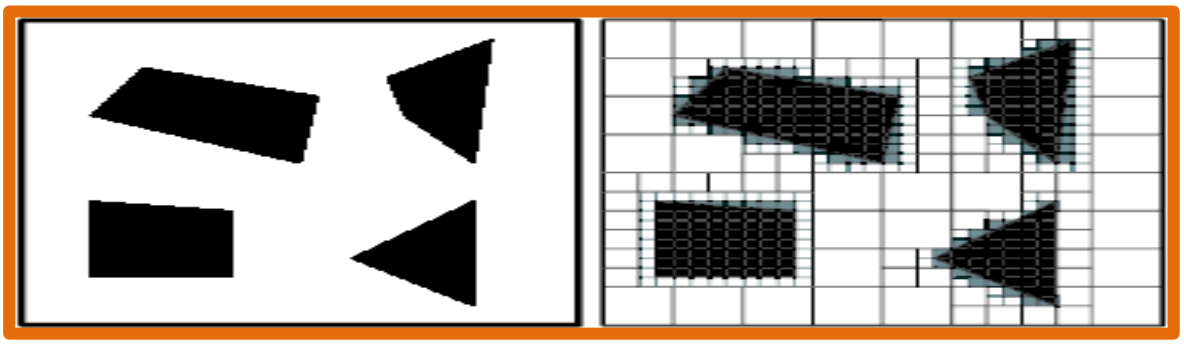


Figure 2 : grilles hiérarchique[7]

c)*Décomposition par rectangle :*

Cette méthode de décomposition consiste à représente l'espace sous la forme d'une collection de rectangle, ces rectangles alignés sur les axes, et recouvrent tout l'espace et ne partagent que des frontière. Cette décomposition (les rectangles) représente une zone de navigation libre c a d une zone contenue à l'intérieur d'un obstacle ou bien une zone comprenant à la fois une zone de navigation et une partie d'obstacle.

L'avantage de cette méthode de permettre de créer des cellules de taille variable enfonction de la complexité de la géométrique de l'environnement.

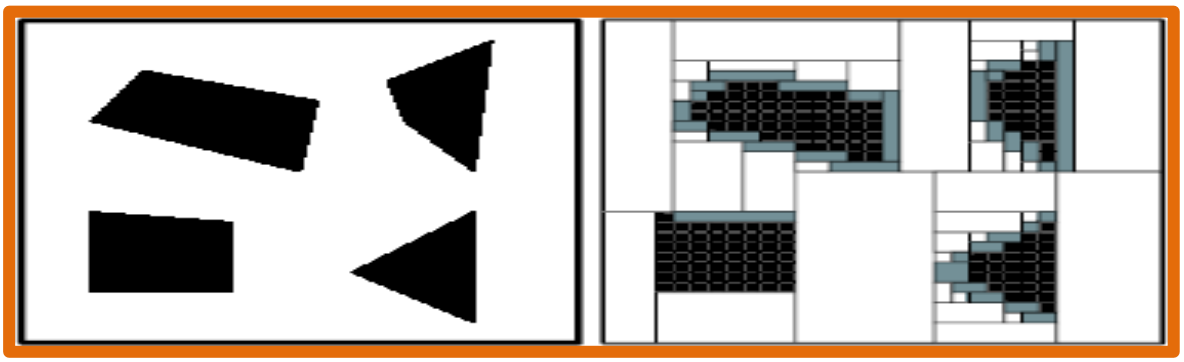


Figure3 :Décomposition par rectangle

4-1-2Décomposition exactes en cellules

À l'inverse des décompositions approchées donc Parmi les techniques utilisées on retrouve, par exemple des décompositions en cellules verticales ou des triangulations de Delaunay. Ces méthodes cherchent à caractériser précisément C-Free en utilisant les bordures des obstacles comme des contraintes formant les données d'en entrées.[10]

a)Triangulation de Delaunay :

La triangulation de Delaunay prend en entrée un ensemble de points du plan et fournit en sortie un ensemble de triangles dont les sommets sont formés par les points fournis en entrée. Ces triangles respectent la contrainte suivante :le cercle circonscrit au triangle ne contient pas de points de P autre que les sommets de ce même triangle .La propriété la plus intéressante de cette triangulation est que chaque point y est relié à son plus proche voisin par l'arrête d'un triangle .

Ainsi cette triangulation peut être utilisée pour représenter l'espace navigable ,les points d'entrée étant issus des obstacles[10][6]

b) Triangulation de Delaunay contrainte en 2d :

Cette triangulation peut être utilisé pour effectuer une subdivision spatiale en triangle les contraintes expriment alors les arêtes des polygones délimitant les obstacles peuplent les environnements, permettent d'effectuer une subdivision d'environnements sous forme de triangle.

Le nombre des arrêtes et de triangle générer la relation de Euler le nombre de triangle utilisé pour la subdivision spatiale est donc linéaire en fonction du nombre de points, indiquant que la discrétisation est donc directement proportionnelle à la complexité géométrique de l'environnement. Cette propriété présente un avantage certain comparativement aux méthodes approximatives, où la discrétisation est fonction de la précision désirée de la représentation.[6]

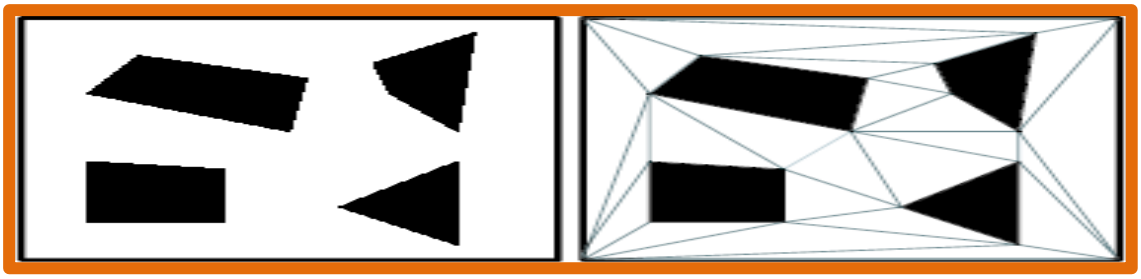


Figure 4 : *Triangulation de Delaunay contrainte[14]*

c)Triangulation de Delaunay filtrée :

La triangulation de Delaunay filtrée est une extension de la version contrainte. Deux types de filtrages sont proposés, l'un ayant pour effet d'augmenter le nombre de triangles produits afin d'affiner la représentation de l'environnement, l'autre de le diminuer afin de tenir compte de la visibilité pour l'élaboration d'un graphe de voisinage[5]. Premièrement, concernant son application à la subdivision spatiale, la triangulation.de Delaunay est filtrée par l'ajout progressif de contraintes représentant les goulets d'étranglement.

Ainsi, en considérant l'ensemble des arêtes produites, on est sûr de représenter tous les rétrécissements présents dans l'environnement. Deuxièmement, concernant son application aux graphes de voisinage, la triangulation de Delaunay est filtrée sur un principe équivalent à la triangulation contrainte, en supprimant les arêtes intersectées des obstacles de l'environnement Ainsi, cette triangulation peut servir à déterminer trivialement quels sont les voisins directs visibles d'une entité, et avec un simple parcours de graphe peut sélectionner les entités visibles à une certaine distance.

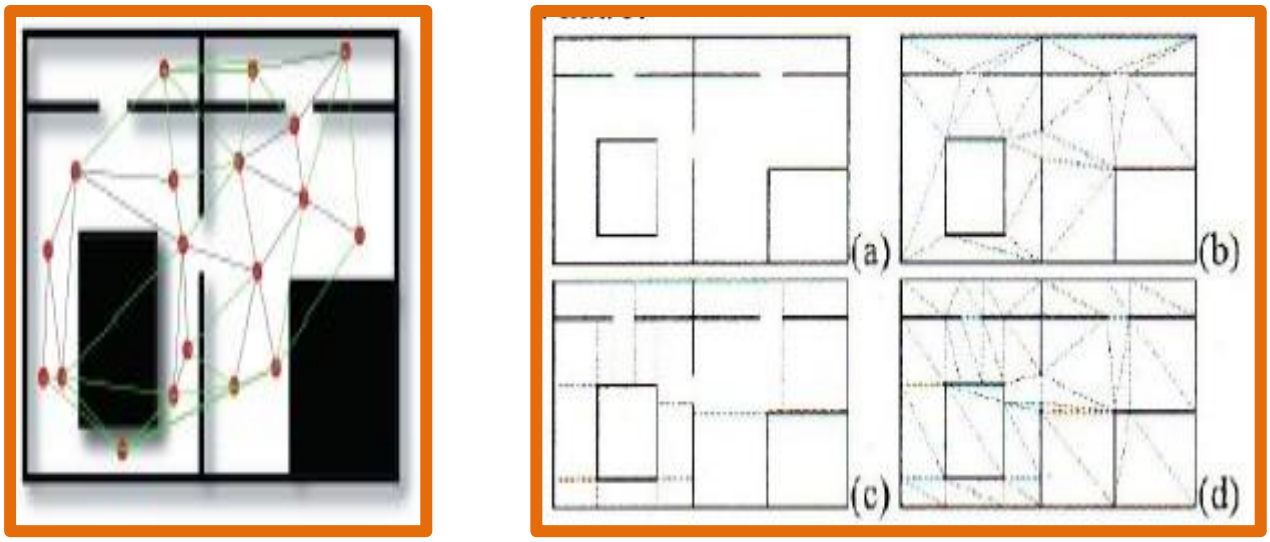


Figure5 :Triangulation de Delaunay filtrée[14]

(a)l'environnement au départ (b) après la triangulation de Delaunay

(c) distance minimum entre les angles et les murs

(d) triangulation de Delaunay en prenant en compte la distance minimum

d)Décomposition en trapèzes:

Permet de décomposer l'environnement sous forme de cellules trapézoïdales , elle est basée sur un algorithme de balayage. La géométrie de l'environnement délimité par des point tries suivant l'axe des cordonnées. Les segments généré (maximum 2 segment), ayant les point sélectionné comme origine et pour extrémité la prochaine intersection avec le bord d'un polygone délimitant un obstacle.[6]

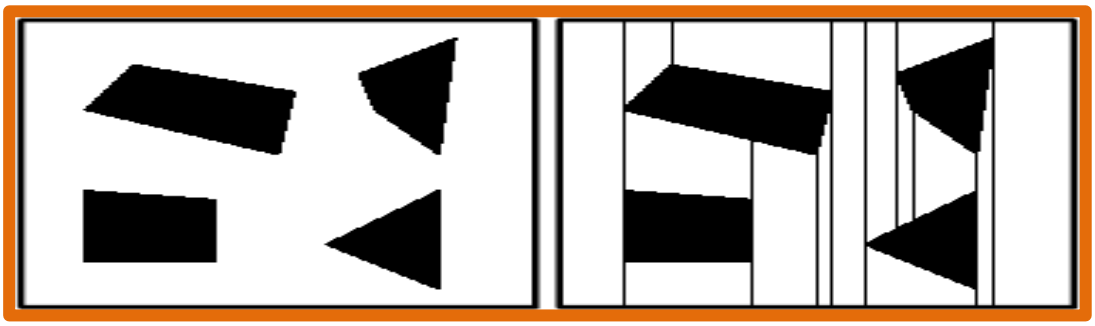


Figure6 :*Décomposition en trapèzes[5]*

4-2Cartes de cheminement :

Une carte de cheminement est une représentation implicite de l'environnement[2], particulièrement bien adaptée pour la recherche de chemins. Ces techniques utilisent un graphe dont les sommets correspondent à des points clés, c'est à dire à des positions où l'humanoïde peut tenir debout sans entrer en collision avec l'environnement. Si une arrête existe entre deux sommets, il est possible de naviguer d'un point à l'autre, nous allons présenté deux familles de cartes cheminemen

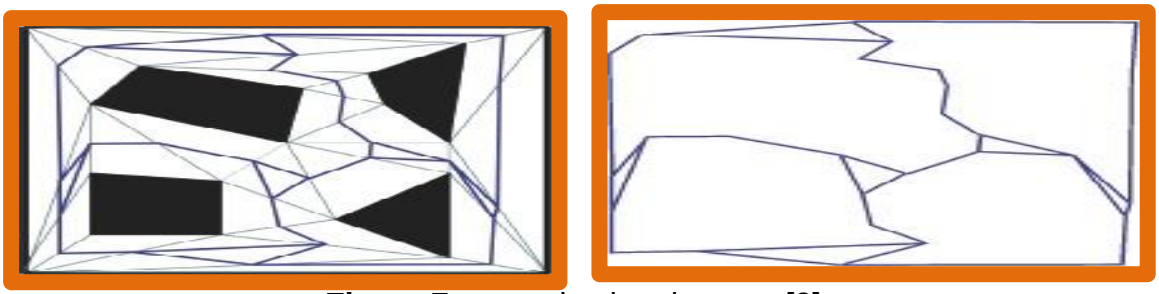


Figure 7 :carte de cheminement[2]

4-2-1carte de chemin déterministes :

Les méthode à bases de cartes de cheminement représentent l'espace sous la fourme d'un réseau de chemins permettant aux entités en évitant les obstacles .pour faire un certain nombre de points clefs sont répartis à l'intérieur de

l'environnement et connectés s'il existe un chemin en ligne droite les reliant sans rencontrer d'obstacle. différentes méthodes , basées sur ce concept existent , elles diffèrent principalement par le mode de génération des points clefs et par la façon de les relier.[14]

a)graphe de visibilité :

Cette méthode consiste à choisir les sommets des polygones d'obstacles comme des points clefs. Ensuite ces points clefs sont relies deux à deux si et seulement s'ils sont mutuellement visibles ,c'est-à-dire si l'on peut tracer une ligne droite passant par les deux points sans d'obtenir des chemins de longueur minimale.[14]

Cette technique à l'inconvénient de produire un grand graphe si l'environnement contient des espaces très ouverts.

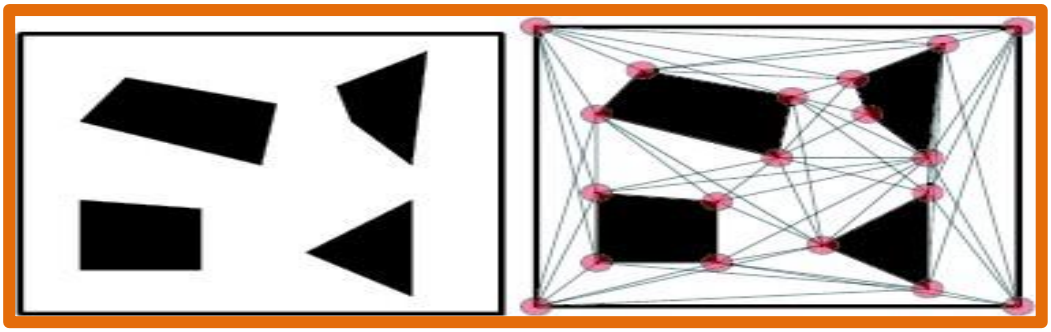


Figure 8 : graphe de visibilité[14]

b)diagramme de voronoï :

C'est une autre méthode pour calculer un roadmap qui maximise le dégagement avec des obstacles .dans ce cas les points de passage du roadmap sont des points équidistants entre les obstacles. L'objectif de ce méthode est de construire un réseau routier qui Rend le robot capable à visiter tous les points possible de l'environnement libre de navigation. [7]

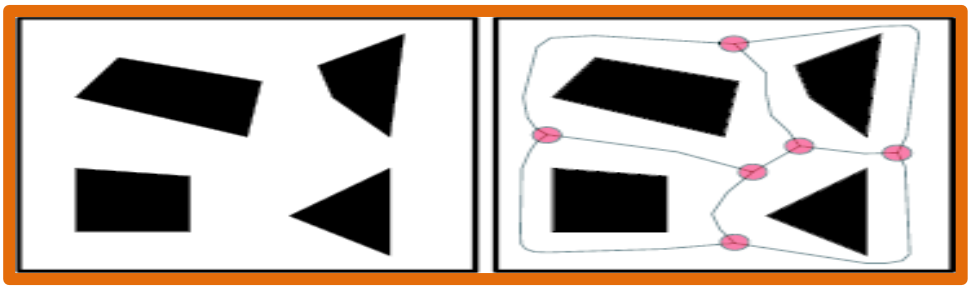


figure 9 :diagramme de voronoï[7]

4-2-2 Cartes de cheminement probabilistes :

Ces méthodes ne trouveront pas forcément le même chemin sans collisions *à chaque exécution*, même avec les mêmes conditions initiales .Ces méthodes ne sont pas complètes en résolution, mais elles garantissent de trouver un chemin sans collisions s'il en existe un. On dit qu'elles sont *complètes en probabilité*[33]. Les *planificateurs probabilistes* font partie de la grande famille des *méthodes d'échantillonnage* (« sampling-based méthodes », en anglais).

- Très efficaces, spécialement pour problèmes définis dans d'une espace des configurations de *très haute dimensionnalité.*

4-Algorithmes *de parcours*

Différents algorithmes de parcours de graphe peuvent être utilisés dans le cadre de la planification de chemin. La minimisation de coût est l'un des critères sur lesquels se basent ces algorithmes pour extraire le plus court chemin. Ce coût représente souvent une notion de distance spatiale.

4-1 ALGORITHME DE DIJKSTRA :

L'algorithme de Dijkstra est un algorithme de planification de trajectoire. Il est conçu pour déterminer le chemin le plus rapide pour se rendre d'un point à un autre Celui-ci utilise deux listes ,sur nommées respectivement la Liste Ouverte et la Liste Fermée, et un système de coût qui représente la difficulté pour se rentre jusqu'à un point.[16]

Dans la Liste Ouverte, on retrouve la liste des points dont les alentours devront être explorés .Dans la Liste Fermée, on retrouve la liste des points dont les alentours ont été explorés Pour ce qui est des étapes de calculs, celui-ci donne initialement un coût infini à tous points du plan .

Les points noirs représentent ici les obstacles. Les points noirs représentent ici les obstacles. Les points noirs représentent ici les obstacles. Par la Ouverte dont le coût est le plus faible et explore les points alentour. Si ceux-ci ne sont pas des obstacles, la valeur de leur coût est changée et ceux-ci sont ajoutés dans la Liste Ouverte.

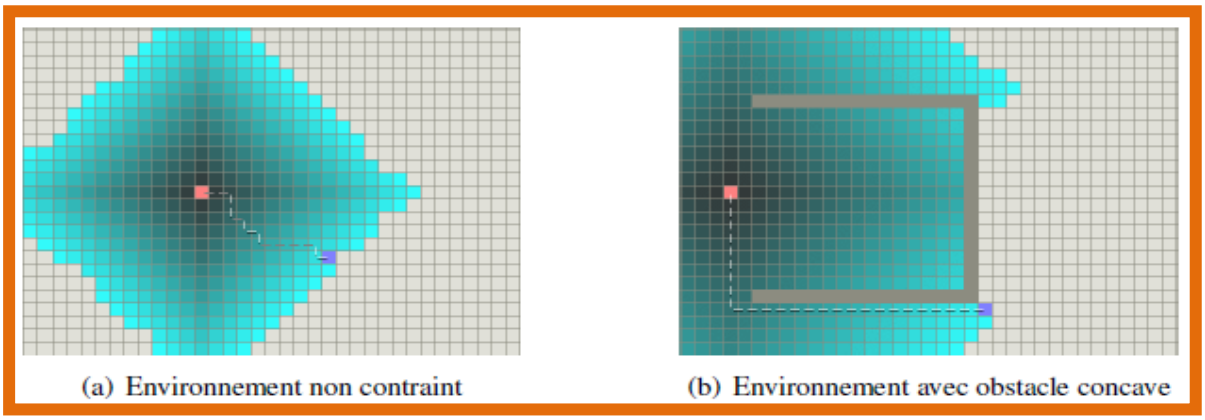


figure10 :algorithme Dijkstra[16]

4-2 ALGORITHME A* :

L'algorithme A* est un algorithme qui est très similaire à l'algorithme de Dijkstra. Les deux fonctionnent exactement de la même façon, mais le coût des points est calculé de façon différente|17] .Contrairement à l'algorithme de Dijkstra, le A* va évaluer le coût des points comme étant le coût parcouru pour s'y rentre plus une prédiction du coût minimal qu'il reste à faire pour arriver à la solution .

Cette prédiction est surnommée l'Heuristique. Dans la plupart des cas, puisque l'on utilise souvent le déplacement comme étant le coût parcouru, c'est souvent la distance euclidienne entre un point et le point d'arrivée qui est utilisée comme heuristique de ce point .Le coût total d'un point sera donc le déplacement parcouru plus la distance euclidienne. Cette méthode tente donc de prédire quel chemin emmènera plus rapidement à la solution.

Pour le reste, celui-ci fonctionne exactement comme l'algorithme de Dijkstra, toutefois, cette différence de coût réduit énormément le nombre de points examiné.

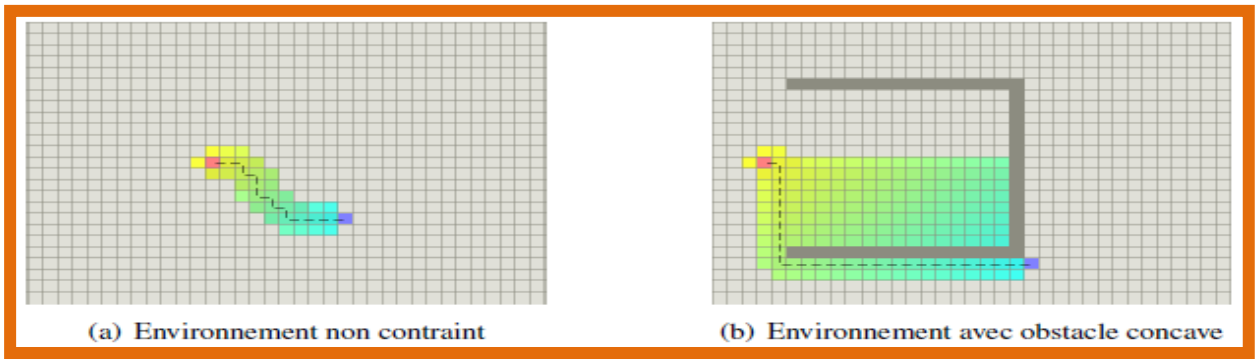


Figure 11 : ALGORITHME A*[17]

4-3ALGORITHME D* :

L'algorithme D* est un autre algorithme fortement utilisé dans l'industrie et celui-ci utilise un principe très similaire à l'algorithme A*. Cet algorithme a été conçu pour fonctionner et réduire le temps de calcul dans un environnement très dynamique. Lorsqu'un obstacle se déplace et entre dans la trajectoire de l'objet en mouvement, celui-ci doit s'arrêter et recommencer ses calculs pour trouver le nouveau chemin qu'il doit prendre. [20]

Pour y arriver, il est possible de démarrer à nouveau l'algorithme A*, mais si ceci se reproduit souvent, le temps de calcul peut devenir très grand .L'algorithme D*, quant à lui, réutilise les données précédentes pour trouver le nouveau chemin.

4-4 ALGORITHME HD* :

Cet algorithme est un autre algorithme fortement utilisé de nos jours. Toutefois, celui-ci ne peut pas, encore une fois, s'appliquer au bras robotique et ce pour deux raisons. Premièrement, celui-ci utilise une méthode de D* [19]. Or, il a été conclu précédemment que l'algorithme D* est nettement moins avantageux que l'algorithme A* pour le bras robotique. Deuxièmement, cet algorithme est utilisé lorsqu'il est impossible pour un système mécatronique de connaitre son environnement en avance . Cet algorithme a été conçu pour fonctionner avec un plan non complet qui se construit au fur et à mesure que l'objet se déplace et découvre son environnement .

4-5-L'algorithme Thêta* :

L'algorithme Thêta* est une modification de l'algorithme A*, qui vise à pallier le problème du lissage. En effet, certaines trajectoires sont plus faciles à lisser que d'autres. L'algorithme Thêta* (qui fait partie d'une manière plus générale de la famille des algorithmes any-angle path planning) va lisser la trajectoire tout en la recherchant. [20]

Il n'y a pas beaucoup de différences de performances avec l'A*, je ne vais donc pas le détailler ; sachez tout de même qu'il existe.

4-6 algorithme RRT :

RRT est un algorithme de planification probabiliste L'algorithme ne généra pas une roadmap qui représente d'une façon exhaustive la connectivité de l'espace libre dans tout l'environnement [30].On explore seulement *une portion de l'espace libre* qui est pertinente à la solution du problème .La méthode RRT utilise une structure des données qui s'appelle« *Rapidly-exploring Random Tree »*(RRT).[31]

- L'expansion incrémentale de l'RRT (on appelle cet arbre « T ») est basée sur une simple procédure stochastique répétée à chaque itération.

La procédure d'expansion du RRT est *simple* et *très efficace* pour « explorer »l'espace libre dans un environnement. Elle est biaiséevers les parties de l'environnement qui n'ont pas été encore visitées .La probabilité que une configuration générique de l'espace libre est rajouté à le RRT tend vers 1 comme le temps d'exécution tend vers l'infini .

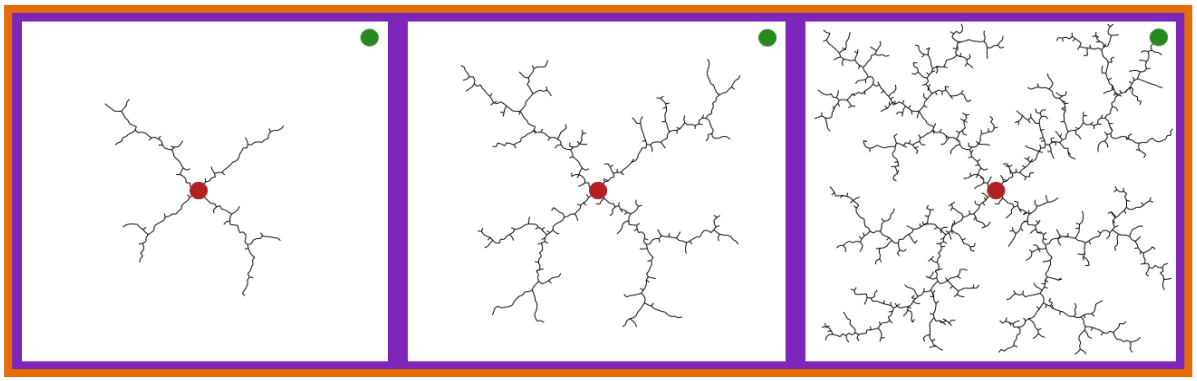


Figure 12 :algorithme RRT[31]

4-7Algorithme RRT star :

RRT star est l'un des algorithmes récents basés sur l'échantillonnage incrémental qui permet de trouver rapidement un chemin initial, puis travaille plus tard pour améliorer et

optimiser ce chemin au fur et à mesure que l'exécution a lieu(Karaman, 2011). Bien que l'algorithme RRT soit un algorithme à efficacité relativement élevée qui puisse gérer les problèmes de planification de chemin avec des contraintes non holonomiques et présente de grands avantages à de nombreux égards, l'algorithme RRT ne garantit pas que le chemin réalisable obtenu est relativement optimisé.[11]

Par conséquent, de nombreuses améliorations sur l'algorithme RRT sont également dédiées à la résolution du problème d'optimisation de chemin, et l'algorithme RRT* est l'un d'entre eux. La principale caractéristique de l'algorithme RRT* est qu'il peut trouver rapidement le chemin initial, puis à mesure que les points d'échantillonnage augmentent, il continuera à s'optimiser jusqu'à ce que le point cible soit trouvé ou que le nombre maximal de cycles soit atteint.

L'algorithme RRT* est une optimisation graduelle, c'est-à-dire qu'au fur et à mesure que le nombre d'itérations augmente, le chemin obtenu est de plus en plus optimisé, et il n'est jamais possible d'obtenir le chemin optimal en un temps limité. En d'autres termes, il faut un certain temps de calcul pour obtenir un chemin d'optimisation relativement satisfaisant. Par conséquent, le temps de convergence de l'algorithme RRT* est un problème de recherche plus important. Mais il est indéniable que le coût du chemin calculé par l'algorithme RRT* est bien inférieur à celui du RRT.[12]

Les algorithmes RRT et RRT * diffèrent principalement par le fait que l'algorithme pour le processus de calcul du poids des deux nouveaux nœuds , respectivement.

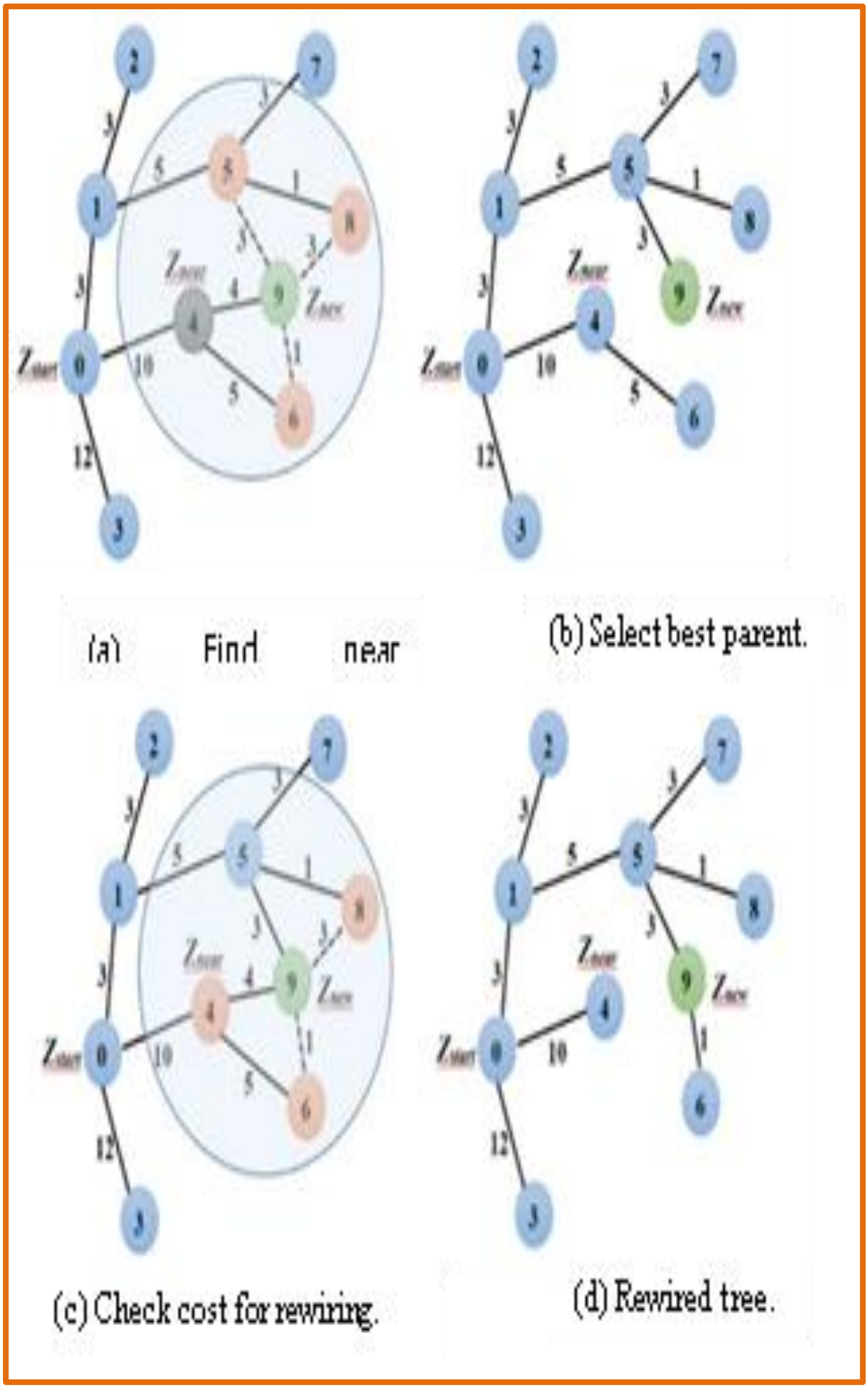


Figure 13 :algorithme RRT *[12]

4-8 informed RRT * :

RRT informé Cette méthode conserve les mêmes garanties probabilistes sur l'exhaustivité et l'optimalité que RRT* tout en améliorant le taux de convergence et la qualité de la solution finale.

Nous présentons l'algorithme comme une simple modification de RRT* qui pourrait être étendue par des algorithmes de planification de chemin plus avancés. Nous montrons expérimentalement qu'il surpasse RRT* en termes de taux de convergence, de coût de la solution finale et de capacité à trouver des passages difficiles tout en démontrant moins de dépendance vis-à-vis de la dimension d'état et de la portée du problème de planification. [16]

Une illustration de l'algorithme Informed RRT* ,Le problème continu est approximé avec un arbre basé sur des échantillons enraciné au départ et initialement construit en utilisant RRT*,Une fois que une solution est trouvée, Informed RRT* élague et focalise la recherche sur l'ensemble renseigné qui contient toutes les solutions éventuellement meilleures, Cet ensemble se rétrécit et concentre la recherche au fur et à mesure que de meilleures solutions sont trouvé, et accélère la convergence asymptotique presque sûre vers le solution

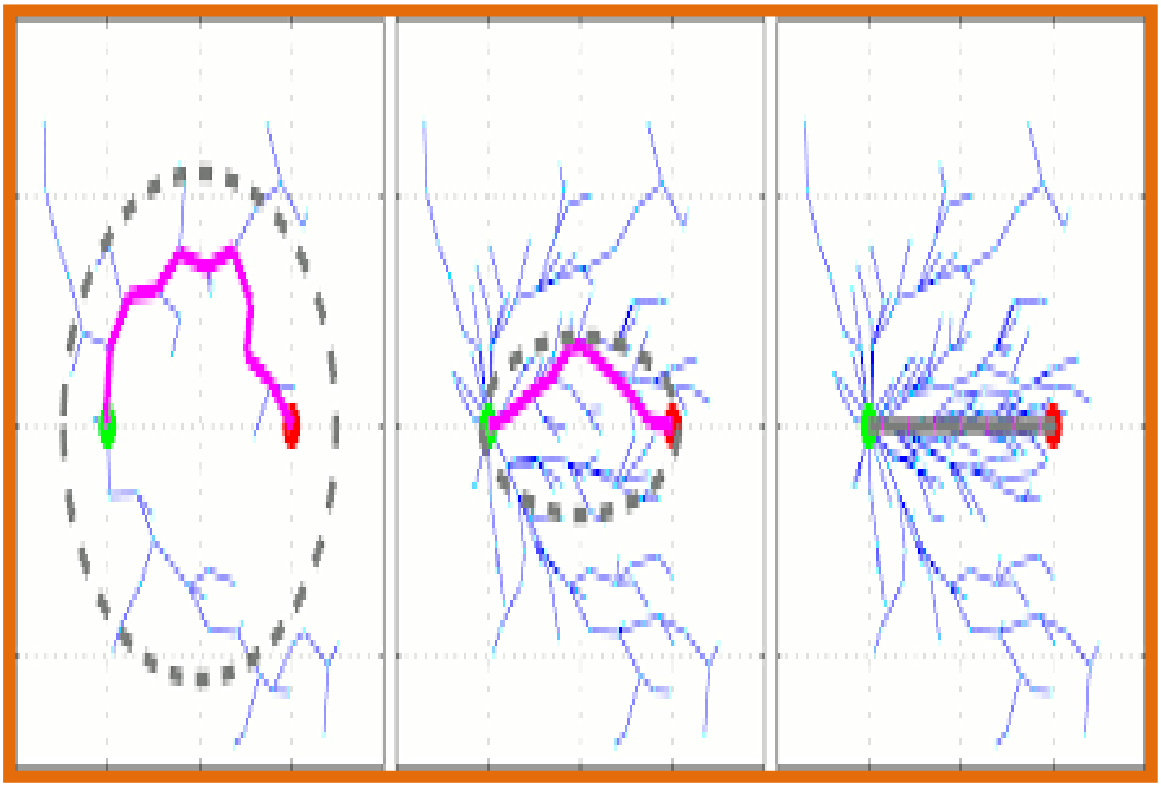


Figure14.algorithme informed RRT *[16]

5-comparaison des algorithme :

La comparaison des algorithme est décrite dans le tableau ci-dessous

Algorithme	*complexité*	*Structure données*	*heuristique*
Djikstra	N^2	*listes*	*Pas d'heuristique*
*A**	*Nlog(n)*	*File de priorité*	*Manhattan/euclidienne*
*Theta**	*Nlog(n)*	*File*	*Manhattan/euclidienne*
RRT	*Nlog(n)*	*Arbre*	*Pas d'heuristique*
*RRT**	*Nlog(n)*	*Arbre*	*Pas d'heuristique*
*INFORMED RRT**	*Nlog(n)*	*Arbre*	*Pas d'heuristique*

Tableau 1 :comparaison des algorithme[20]

6-conclusion :

De nombreuses solutions ont été proposées pour la planification de chemin au sein d'environnements virtuels. Ce domaine de recherche est très actif dans le contexte des mondes virtuels et de l'animation comportementale dont le but consiste à automatiser le peuplement des environnements virtuels à l'aide d'agents virtuels autonomes.

Toutefois, les solutions proposées s'adressent chacune à des problèmes de planification de chemin particuliers pouvant être classés en différentes catégories.

Les premières techniques proposées se sont tout d'abord concentrées sur la navigation au sein d'environnements . En réduisant le problème , il est ainsi possible de prétraiter les environnements afin notamment de simplifier l'exploration de l'espace des configurations lors de requêtes de planification. Certaines méthodes proposent de représenter l'environnement à l'aide de subdivision exactes ou approchées de l'environnement. Ces méthodes permettent d'obtenir de bonnes représentations des Obstacles, mais conviennent principalement à des environnements de faible dimensions. Les méthodes basées sur les cartes de cheminement permettent d'un autre côté d'explorer l'environnement pour en construire une représentation explorant de manière plus performante les espaces de configurations de plus grandes dimensions.

Plus récemment, la problématique de la planification de chemin s'est étendue à des environnements peuplés d'obstacles . Au sein de tels environnements, l'agent doit être capable de détecter les obstacles qui se sont déplacés, et d'adapter son chemin en fonction

Chapitre 2

Conception du système

1-introduction :

Dans ce chapitre nous présentons l'algorithme informed RRT* ,notre principal objectif est de clarifier le cadre formel permettant de planifier un chemin qui permettant à une entité virtuelle de déplacer au sein de son environnement virtuel en prenant en compte la topologie de l'environnement et ses obstacles et permettre à l'éntité virtuel d'éviter en collision avec les autre entités .

Dans ce travail nous considérons les environnement de simulation telle que le cas des environnements réels, donc on choisissons l'algorithme informed RRT* pour trouver un chemin et faire doter l'entité virtuel les capacités qui sont nécessaires pour faire suivre le chemin trouver en préservant le réalisme de simulation.

Son but global est préciser les service et le objectifs principaux qui seront réaliser et rendus par le logiciel à l'utilisateur .cette phase est suivre par une phase cruciale qui est la phase de conception .dans n'importe quel système, la conception est l'étape la plus importante.

La phase de conception est réalisée par un processus itératifs pour déterminer les divers composants et modules du système et leur interactions .une bonne conception est la clé de développement d'un logiciel efficace .un système bien conçu est facile à réaliser ,à maintenir facile à comprendre .la dernière phase consiste en une réalisation ,lors de cette étapes on réalise un ensemble d'unités de programme , écrites dans un langage de programmation précis.

2-objectifs

L'objectif de ce travail est l'étude d'une planification probabiliste incrémentale. L'étude est centrée sur les méthodes du type RRT appliquées à recherche de chemin utilisant un robot. La réalisation. d'un planificateur de chemin probabiliste suppose l'assemblage d'un ensemble de composants responsable d'opérations spécialisées; les principaux composants de cet ensemble, issus de la géométrie algorithmique, sont la détection de collision et la recherche de plus proche voisin .

Pour arriver au notre objectif, les étapes proposées sont comme suit :

- ✓ Faire une planification de l'environnement en utilisant l'algorithme informed RRT*.
- ✓ Le rendu en utilisant Ogre 3d.

3-conception générale de notre système :

Dans ce section nous visons à présenter la structure globale de notre système proposé ,cette structure est illustrée dans le schéma de la figure 15 .la première étapes on représente l'environnement ,dans La deuxième étape consiste à faire une planification à l'aide de l'algorithme d'exploration des arbres rapide informed RRT*.

Le schéma suivant se résume en la réalisation de la planification chemin de l'entité virtuel .le rendu de la scène est réalisé par le moteur du rendu « Ogre 3D »

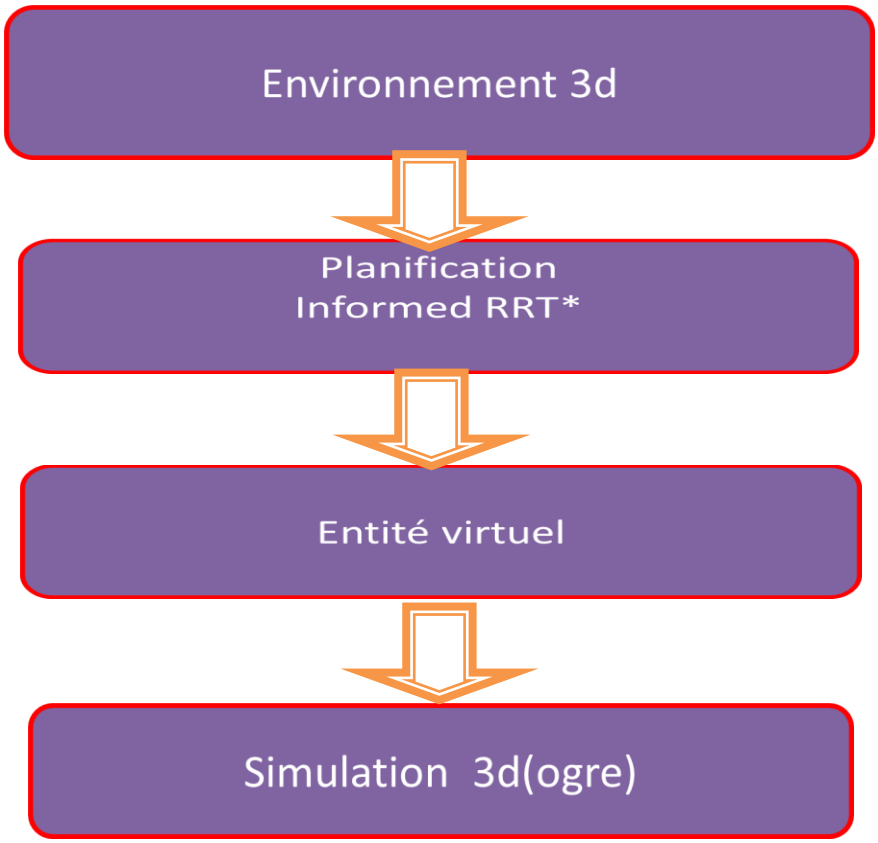


Figure 15 :schéma générale de l'application

-les entrées du système sont :

Scène en 3D :représente l'environnement qui contient les obstacles .

-les sorties du système sont :

Chemin optimal sans collision avec des obstacles pour arriver au but.

4-conception détaillée de notre système :

Notre système est composé des modules suivants

4-1Représentation l'environnement et planification de chemin

La représentation de l'environnement et la planification de chemin dans notre système pour but de le raffiner et l'enrichir en précisant exactement d'où l'entité virtuel va passer dans chaque sous-espace de l'environnement.

4-2 Algorithme RRT :

La représentation de l'environnement et la planification de chemin peut également être effectuer à l'aide de RRT car cette méthode et complète en probabiliste basées sur l'échantillonnage probabiliste nécessitent ,pour leur rapidité ,de pouvoir tirer aléatoirement des points dans l'espace des configurations facile de mise en place après que RRT à terminé itérations il y aura un chemin qui utilisé .[32]

Le pseudo code de l'algorithme :

Algorithme RRT (X_{init},X_{goal})

T.init(qinit) ;

pour i= 1 to NB_it **do faire** **//**NB_it :c'est le nombre d'itération

X_{rand} ← Random(M) ;
X_{near} ← NEAR(X_{rand},T) ;
X_{new} ← STEER(X_{rand}, X_{near},StepSize) ;
E_i ← EDGE(X_{new}, X_{near}) ;
Si collisionFREE (M,E_I) alors // pas d'intersection avec les obstacle
T.addNode(X_{new});
T.addEdge(E_i);
Finsi

SI $X_{new=}$ X_{goal} **alors**
Success() ;
Finsi

Fin pour

Fin

FONCTION :

- Rand() : échantillonne un point dans l'espace libre;
- Near(X_{rand},T) : point le plus proche de *x* dans le graphe *T ;*
- Steer(X_{rand}; X_{near}) : point à une distance donnée de X_{rand} vers X_{near} ;
- • CollFree(**M,E_I**) : pasd'obstacle entre ;

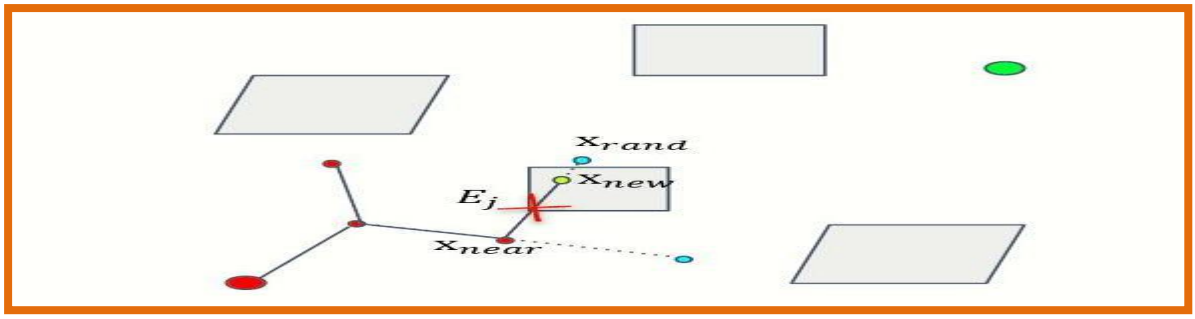


Figure .16 algorithme RRT

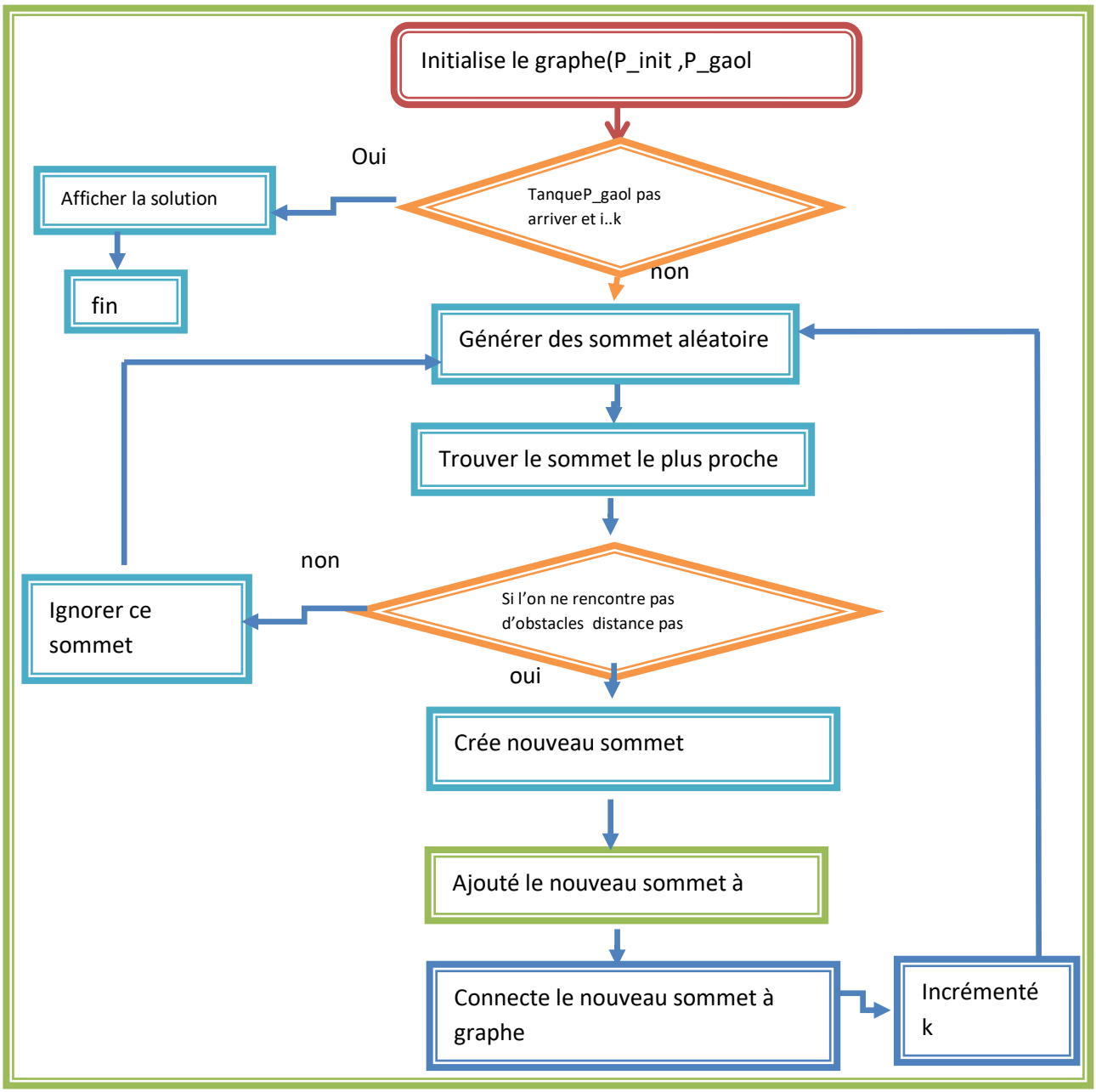


Figure 17 :organigramme de l'algorithme RRT

Dans RRT on ne choisi pas à l'avance quel sommet sera le courant :on choisit le plus proche point du graphe de celui qui vient d'être tiré ,à savoir si c'est un sommet ,on tente de rajouté une arête à ce sommet et si ce point tombe sur une arête ,on crée un nouveau sommet à cet endroit et on rajoute l'arête correspondante jusqu'à le nouveau sommet représente le point de destination ou la distance converge vers le but.

4-3 L'algorithme RRT* :

L'algorithme RRT* étend d'abord le voisin le plus proche vers l'échantillon cependant il

relie le nouveau sommet ,au sommet qui entraine le cout accumulé ce sommet se situe dans l'ensemble des voisinages de ce sommet .De ce fait l'algorithme RRT* permet de trouver une solution qui converge à chaque fois ver le chemin optimal.[29]

Le pseudo code de l'algorithme RRT*:

Algorithme RRT STAR (X_{init},X_{goal})

Debut

Initialiser d'arbre avec Pinit ;

pour i= 1 to NB_it **do faire//**NB_it :c'est le nombre d'itération

X_{rand} ← Rand(M) ;
X_{near} ← NEAR(X_{rand},T) ;
X_{new} ← newconfig(X_{near}, X_{rand})

Si (la configuration de X_{new}est valid) alors
//trouver le nœud de meilleur cout permis les voisins (inclus dans le cercle de centre X_{new}et de rayon R)
min ← choix parent (X_{new},R)
Mettre min comme parent de X_{new}
Inserer Nœud(X_{new}) ;

Fin si

Fin pour

fin

processus spécifique :

1. Générer un rand de points aléatoire。

2. Trouvez et x dans l'arborescence jusqu'au nœud x le plus proche。

3. Connectez le rand avec le plus proche。

4. Prenez le rand centré, r par rapport au rayon, trouvez les nœuds dans l'arbre.

5. Trouver le nœud parent possible Xpotential_parent Il est prévu de mettre à jour le rand pour voir s'il existe un meilleur nœud parent.

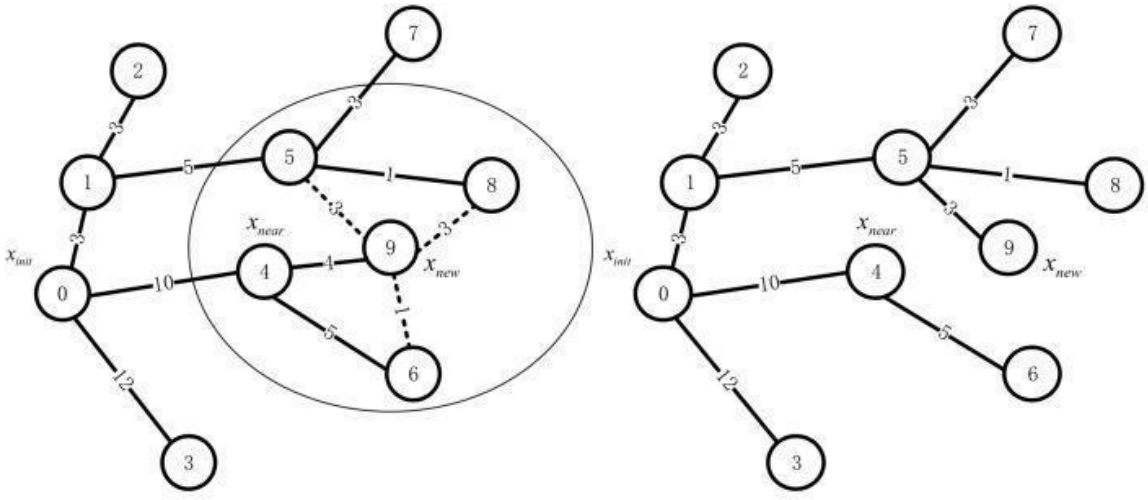
6. À partir d'un nœud potentiel xpotential_parent, commencez à réfléchir.

7. Calculez le coût supplémentaire d'être le nœud parent.

8- Comparez le coût du nouvel itinéraire au coût de l'itinéraire d'origine. Si le coût du nouvel itinéraire est inférieur, effectuez une détection de collision. Si le coût du nouvel itinéraire est plus élevé, passez à l'actif potentiel suivant. nœud.

9-Supprimez l'arête précédente dans l'arborescence.

10-Parcourez tous les nœuds parents possibles pour obtenir l'arborescence mise à jour.[15]



1)initialisation de l'arbre **2)trouver le plus proche sommet**

Figure 18 :les opération de algorithme RRT*[15]

Dans le nœud nouvellement créé x_{new}, recherchez "Near Neighbors" dans une plage spécifiée comme alternative x_{new} pour le nœud parent. Calculez le coût du chemin du nœud « plus proche voisin » au point de départ à son tour x_{nouveau} montre le coût du chemin par « voisin ».

montre un moment dans le processus d'expansion de l'arbre aléatoire, l'étiquette du nœud indique l'ordre dans lequel le nœud est créé, le nœud 0 est le nœud initial, le nœud 9 est le nœud nouvellement créé x_{nouveau}, 4 nœuds produisent 9 nœuds

x_{nouveau}(Désolé Ici 6 nœuds sont 9 nœuds x_{nouveau}, l'image est fausse） , le nombre sur le bord reliant le nœud au nœud représente la distance euclidienne entre les deux nœuds (ici nous utilisons la distance euclidienne pour représenter le coût du chemin).

Dans le processus de raffinement du nœud parent, 9 nœuds x_{nouveau} comme centre du cercle, trouvez le rayon à l'intérieur du cercle x_{nouveau} Les voisins les plus proches sont 4, 5 et 8 nœuds.

Le chemin d'origine 0-4-6-9 a un coût de 10 + 5 + 1 = 16, et les trois nœuds alternatifs sont x_{new} le chemin de 0-1-5-9, 0-4-9 et 0-1-5-8-9 Cela coûte 3 + 5 + 3 = 11, 10 + 4 = 14 et 3 + 5 + 1 + 3 = 12, donc si le nœud 5 est le nouveau nœud parent du nœud 9, le le coût du chemin est relativement faible, nous changeons donc le nœud parent du nœud 9 Du nœud d'origine 4 au nœud 5, l'arbre aléatoire reconstruit [15]

redirection de processus d'arborescence aléatoire :

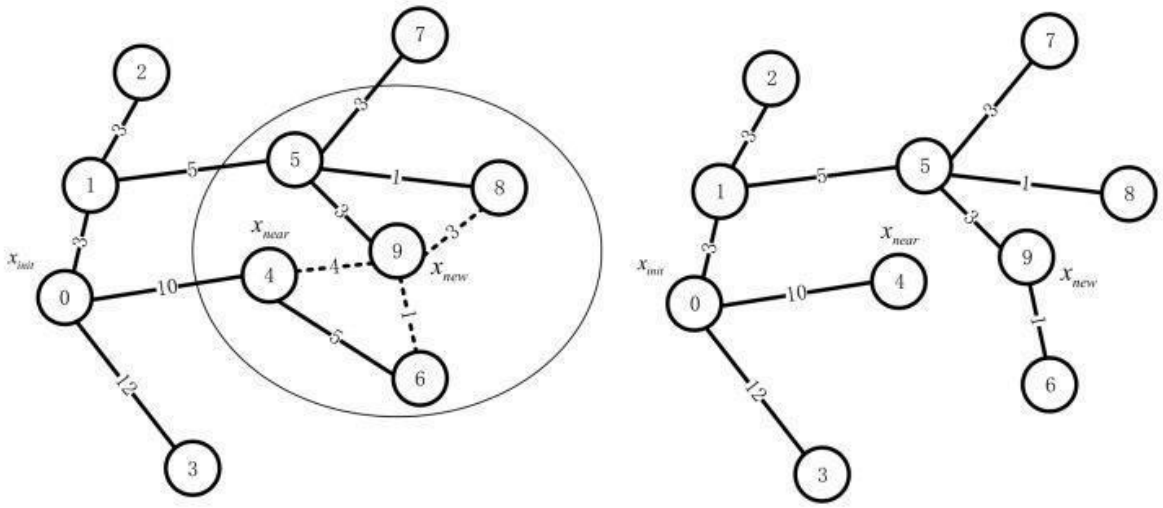


Figure 19 :les opération de algorithme RRT*[15]

Pour x_{nouveau} Une fois le nœud d'origine redéfini, afin de réduire au maximum le coût de communication entre les nœuds de l'arbre aléatoire, l'arbre aléatoire est redirigé. Le processus est illustré à la figure 4. Le processus de reconnexion peut également être exprimé comme suit : Si le nœud parent du nœud adjacent est changé en x_{nouveau} si le coût du chemin peut être réduit, alors changez-le.

9 nœuds sont des nœuds x_{nouveau} nouvellement créés, les nœuds adjacents sont respectivement 4, 6 et 8 nœuds. Les nœuds d'origine sont les nœuds 0, 4 et 5. Les chemins sont 0-4, 0-4-6, 0-1-5-8, et les coûts sont 10, 10 + 5 = 15 et 3 + 5 + 1 = 9.

Si vous changez le nœud d'origine de 4 nœuds à 9 nœuds x_{nouveau} alors le chemin de nœud 4 devient 0-1-5-9-4, le coût est de 3 + 5 + 3 + 4 = 15 qui est supérieur à l'original coût du chemin de 10, donc Le nœud d'origine du nœud 4 ne change pas.

De la même manière, si le nœud d'origine à 8 nœuds est modifié, le coût du chemin passera de 9 à 14 sans changer le nœud d'origine à 8 nœuds. Si vous modifiez le nœud d'origine de 6 nœuds en x_{nouveau}, le chemin devient 0-1-5-9-6 et le coût est de 3 + 5 + 3 + 1 = 12 , ce qui est inférieur au coût du chemin d'origine de 15, donc le nœud est changé Le 6 d'origine au nœud 9, et le nouvel arbre aléatoire généré est montré.

L'importance du recâblage est de savoir s'il est possible de réduire le coût de routage de certains nœuds en recâblant chaque fois qu'un nouveau nœud est créé. D'un point de vue global, tous les nœuds redirigés n'apparaîtront pas dans le chemin final généré, mais dans le processus de génération d'un arbre aléatoire, chaque redirection créera des opportunités pour réduire autant que possible le coût du chemin final. .

L'essence de l'algorithme RRT* réside dans les deux opérations ci-dessus : re-sélectionner le nœud d'origine et reconnecter les fils. Ces deux processus se complètent. Réinitialisez le nœud d'origine pour réduire au maximum le coût du chemin du nœud nouvellement créé et reconnectez l'arbre aléatoire après la création du nouveau nœud pour réduire les chemins redondants et le coût du chemin.[21]

Certaines fonctions ont les mêmes définitions et fonctions que l'algorithme RRT :

calculate(dist(x_{new},x_{near-neighbor})+cost(x_{near-neighbor},x_{init})) la fonction dist calcule la distance entre deux points, et la fonction cost calcule le coût du chemin à partir d'un point donné le long de chaque arête de l'arbre aléatoire jusqu'au point de départ. Ici, le coût du chemin est la somme de la distance euclidienne d'un point donné le long de chaque bord de l'arbre aléatoire jusqu'au point de départ.

min(dist(x_{near-neighbor},x_{new})+cost(x_{new},x_{init})) signifie choisir celui qui réduit le coût de l'itinéraire x_{near-neighbor}。

De même, min(dist (x_{new},x_{near-neighbor})+cost(x_{near-neighbor},x_{init})) signifie choisir celui qui réduit le coût de l'itinéraire x_{new}。

4-4 Algorithme Informed RRT* :

RRT* est asymptotiquement optimal partout Ceci n'est pas nécessaire pour la planification de requête unique Nous devrions limiter la recherche au sous-problème qui

aurait une meilleure solution Ce sous-problème est facile à définir lors de la minimisation de la longueur du chemin C'est une ellipse symétrique à n dimensions L'échantillonnage de ce sous-ensemble est nécessaire pour améliorer la solution La probabilité de faire soc an devient arbitrairement petite :

Gros problèmes de planification Bonne solution Dimensions de l'état élevé Informed RRT* échantillonne uniformément ce sous-ensemble directement :

Concentre la recherche uniquement sur le chemin entre le départ et le but Assure que chaque échantillon pourrait améliorer la solution Simple et peu coûteux en calcul Efficace même dans des dimensions élevées Peut être combiné avec d'autres techniques de planification lissage de chemin.[3]

Informed RRT* est une simple modification de RRT* qui démontre une nette amélioration. En simulation, il effectue ainsi que les algorithmes RRT* existants sur des configurations simples, et démontre des améliorations de l'ordre de grandeur au fur et à mesure que le les configurations deviennent plus difficiles. En conséquence desa recherche ciblée, l'algorithme dépend moins de la dimension et domaine du problème de planification ainsi que la capacité de trouver de meilleurs chemins topologiquement distincts plus tôt.

Il est également capable de trouver des solutions dans des tolérances plus strictes de l'optimum que RRT* à calcul équivalent, étend l'absence d'obstacles peut trouver la solution optimale pour dans le zéro machine en temps fini. Cela pourrait aussi être utilisé en combinaison avec d'autres algorithmes, tels que chemin lissage, pour réduire encore l'espace de recherche.[2]

Le pseudo code de l'algorithme informed RRT*:

Algorithm 1: InformedRRT*($\mathbf{X}_{start}$, x_{goal})

$X_{solution} \leftarrow \emptyset$ // Solutions de groupe vide

pour i=1 **to** NB_it **faire** //NB_it :c'est le nombre d'itération

$C_{best} \leftarrow \min_{xsolution \epsilon X_{solution}}$;

$X_{rand} \leftarrow$ **sample(**$\mathbf{x}_{init}$,$\mathbf{x}_{goal}$) ;

$X_{nearest} \leftarrow$ Nearest(T, X_{rand}) ;

$X_{new} \leftarrow$ steer($X_{nearest}$,X_{rand}) ;

Si collisionfree($X_{nearest}$, X_{new})**alors**

$X_{nearest} \leftarrow$ Near(t, X_{new}) ;

$x_{min} \leftarrow X_{nearest}$;

pour x_{near} ε X_{near}**fair**

$c_{new} \leftarrow$ **cost (**X_{near})+c.line($X_{nearest}$,X_{new}) ;

si $c_{new} < c_{min}$ **alors**

Si collisionfree($X_{nearest}$,X_{new})**alors**

$X_{min} \leftarrow x_{near}$;

$c_{min} \leftarrow c_{new}$;

finsi

finsi

fin pour

$E \leftarrow E \cup (X_{min},X_{new})$;

Pour x_{near} xnear **fair**

$C_{near} \leftarrow$ cost(x_{near}) ;

$c_{new} \leftarrow$ cost(x_{new})+c.line(x_{new},x_{near}) ;

si $c_{new} < C_{near}$ **alors**

si collisionfree(X_{new}, x_{near}) alors

$x_{parent} \leftarrow$ parent (x_{near}) ;

$E \leftarrow E \backslash (x_{parent}, x_{near})$;

$E \leftarrow E \cup (X_{new},x_{near})$;

finsi

finsi

fin pour

si in goal region (X_{new})**alors**

$x_{soln} \leftarrow x_{soln} \cup (X_{new})$

fin si

fin si

fin pour

fin

fonction sample :

sample(x_{init},x_{goal},c_{max})

si $c_{max} < \infty$ **alors**

$c_{min} \leftarrow \|x_{gaol} - x_{init}\|_2$;

$x_{centre} \leftarrow (x_{init} + x_{gaol})/2$;

c $\leftarrow$ **rotation world Frame(**x_{init},xgaol) ;

$r_1 \leftarrow c_{max}/2$;

$r_i \leftarrow (\sqrt{c2max - c2min}$;

$L \leftarrow$ diag(r_1,r_2,…r_n) ;

Sinon

$X_{rand} \cup (x)$;

Fin si

fin

sample : Étant donné deux poses et une valeur heuristique maximale, cmax R, la fonction L'échantillon (x , cmax) renvoie indépendamment et à l'identique distribué (échantillons de l'espace d'état, xnew X ,tel que le coût d'un chemin optima qui est contraint de passer par xnew est inférieur à cmax comme décrit dans la Section III et dans la plupart des problèmes de planification, xinit, xgoal, peut être calculé une fois au début du problème.

InGoalRegion :renvoie True si et seulement si l'état est dans InGoalRegion : étant donné une pose la fonction de la région cible, Xgoal, telle que définie par le problème de planification, sinon, il renvoie False. Une région de but commune est une boule de rayon rgoal centrée sur le but.

`RotationToWorldFrame` :étant donné deux poses comme points focaux d'un hyperellipsoïde, la fonction RotationToWorldFrame renvoie la matrice de rotation, du cadre aligné sur l'hyperellipsoïde au cadre universel sous la forme .[7]

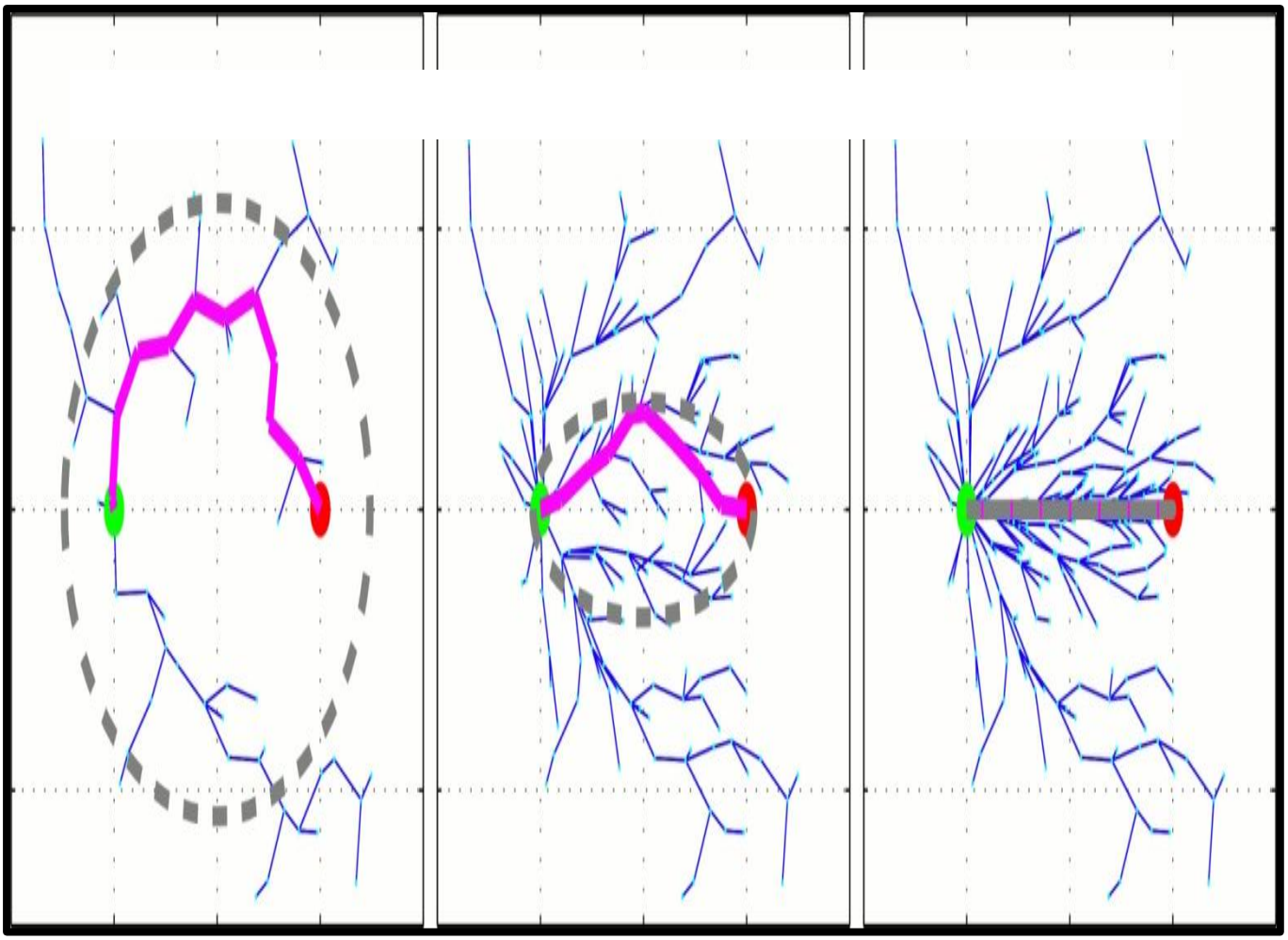


Figure 20 : algorithme informed RRT*[7]

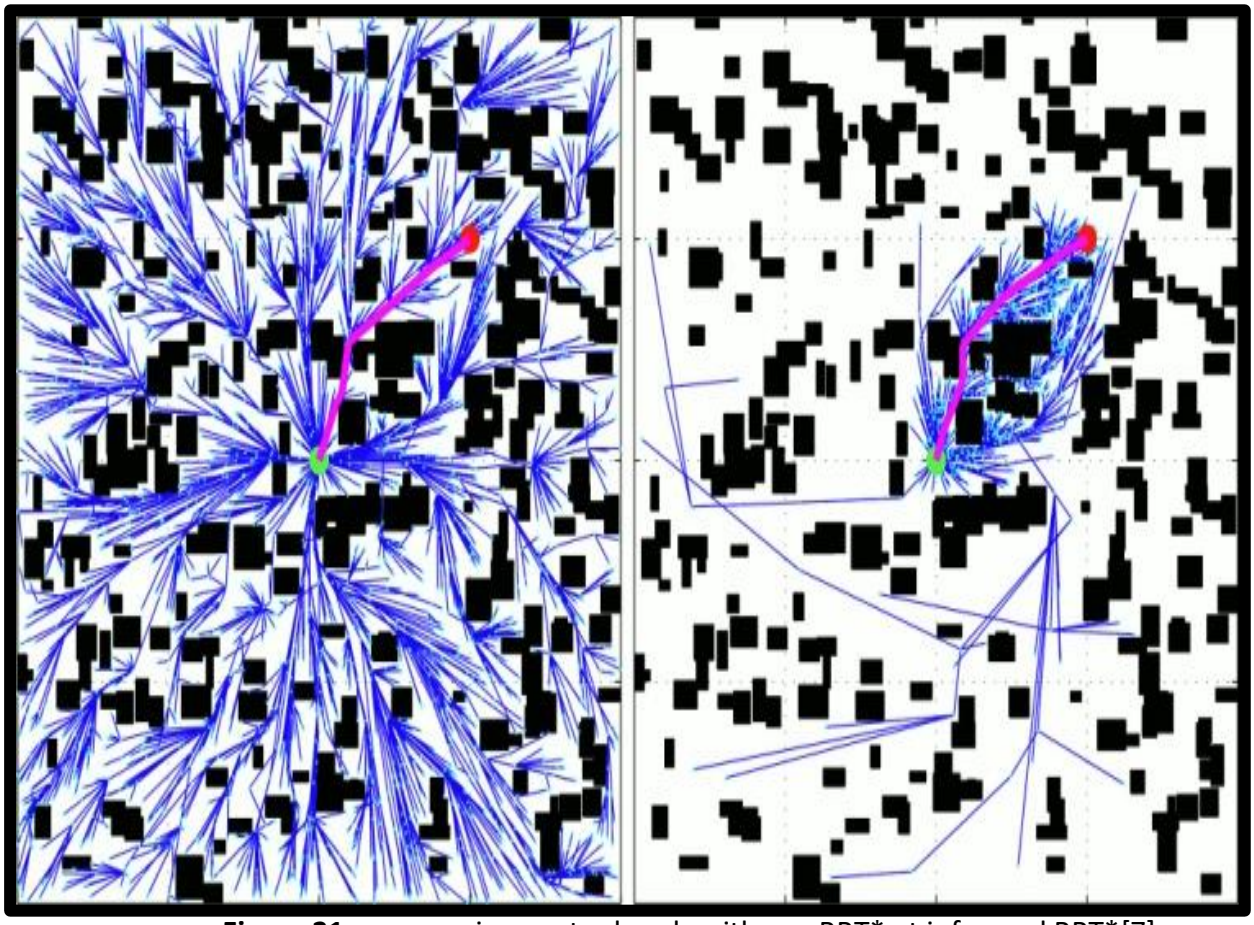


Figure 21 :comparaison entre les algorithmes RRT* et informed RRT*[7]

5-conclusion

Dans cet chapitre, nous discutons qu'une condition nécessaire pour que les algorithmes RRT et RRT* améliorent une solution est l'ajout d'un état à partir d'un sous-ensemble du problème de planification, .Pour les problèmes cherchant à minimiser la longueur de chemin dans Rn, ce sous-ensemble peut être estimé , , par un hyper sphéroïde allongé (un problème à des sous-ensembles du domaine d'origine. Cela le rend intrinsèquement dépendant du coût de la solution actuelle, car il ne peut pas concentrer la recherche lorsque l'hypersphéroïde allongé associé est plus grand que le problème de planification lui-même. De même, il ne peut que réduire le sous-ensemble jusqu'à la limite inférieure définie par la solution optimale.

Chapitre 3 :

Implémentation et résultats

1-introduction :

Dans ce chapitre nous allons décrire la mise en œuvre des différentes étapes de notre système conçu dans le chapitre précédent .nous allons commencer par dévoiler l'environnement de développement utilisé ,puis présenter les structures de données et les fonctions utilisées ,nous allons enfin présenter quelques résultats expérimentaux de notre travail .

2-outils de développement

2-1 langage de programmation

Après l'étude de nos besoin, nous avons choisi l'environnement de programmation Visual studio 2010 programmé notre application en utilisant le langage c++ , notre choix est motivé par les faits suivantes :

- Le langage de programmation c++ est un langage très puissant .
- Le langage c++ offre la possibilité de programmer avec les classes et les objets .

Figure 22 :Microsoft Visual studio 2010

2-2 moteur de rendu :

Ogre 3d est un moteur graphique open source (object oriented graphics rendering engine) ,qui veut dire moteur de rendu graphique orient objects ,un moteur graphique est la solution pour afficher en temps réel des formes 3D. c'est la base de toute application graphique (jeu vidéo ,simulateurs ,réalité virtuelle)

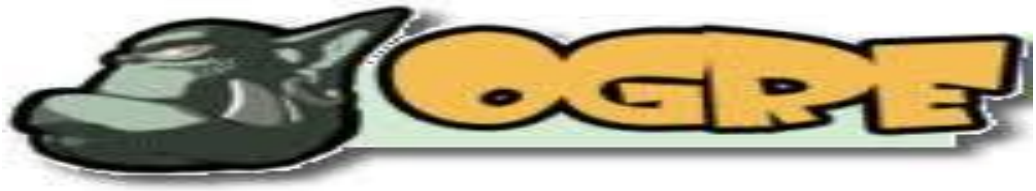


Figure 23 :moteur de rendu ogre

scène menager :

Scène manager est un élément d'ogre responsable d'organiser et de rendu les objets dans une scène .Tout ce qui apparait dans une scène est contrôlé par le scène manager .quand un objet est placé sur la scène , la classe scène manager s'occupe de garder en mémoire leurs locations .

Entity :

Une entité est la représentation dans la scène d'un modèle 3D ,aussi appelé mesh est l'ensemble des polygones élémentaires que l'on crée dans un logiciel de modélisation et qui constitue le modèle 3D complet .tous ces polygones sont reliés eux par leurs sommet plus il y a de sommets et donc de polygones .plus le mesh est précis généralement ,le mesh possède aussi une ou plusieurs textures qui lui donnent un aspect plus réaliste qu'une couleur unie lors du rendu .

On peut créer une entité comme suit :

```
Entity *head= mSceneMgr->createEntity("Tete", "ogrehead.mesh");
```

Tel que le premier paramètre da la fonction createEntity est le nom de l'entité et le deuxième est le nom du fichier qu'on veut charger .ce fichier a l'extention «.mesh»
Et contient les modèles reconnus par ogre .

- sceneNode :

un sceneNode est un objet invisible auquel on va pouvoir attacher un nombre indéfini d'entités ,lesquelles deviennent solidaires de ce nœud et subissent donc les mêmes transformations qui lui .c'est donc une sorte de conteneur qui contient les

informations de positionnement de chacune des entités de la scène qui lui sont rattachées .

- camera

la camera est élément qui défini la position de notre point de vue dans la scène dans laquelle direction on regarde ,mais aussi jusqu'à quelle distance il est possible de voir s'afficher les objets éloignés

la fonction suivante permet de crée une caméra :

```
mCamera= mScenemgr->create Camera( '' Ma_camera '') ;
```

le déplacement de la camera se fait par la fonction **setPosition()** en déterminant

coordonnés x,y,z :

```
mCamera->setPosition(vector3(x,y,z)) ;
```

lookAT(),comme son nom l'indique ,permet de déterminer le point de la scène observé par notre caméra

mCamera->lookAt

(verctor3(0.0,100.0,0.0)) ;

3-structures de données :

3-1Rprésentation de l'environnement :

3-1-1 Représentation du nœud :

On à utilise la class Nœud pour représenter l'élément de base de notre graphe de recherche

Classe Nœud{

X,Y,Z :entier ; //coordonnées du sommet

Parent :pointeur vers le nœud parent ;

3.1.2 Représentation de l'arbre et du chemin calculé :

Dans notre système on utilise deux tableau de l'objet Nœud :

Tableau Graph :qui est constitué de tous les Nœuds représentant l'arbre d'exploitation

Tableau Path :qui contient les Nœuds constituant le chemin reliant le point de départ avec celle de destination.

Ces tableau dynamique en utilisant classe vector de la bibliothèque standard de c++

3-1-3 Représentation de l'espace de navigation :

On utilise la class RRT pour représenter chaque sous espace de navigation et le chemin planifié associer

Class RRT{

P_init :le nœud de départ.

P_goal :le nœud arrivé.

P_ref :nœud de référence

Haut,Long ,NB_it,Step :entier

}

Telque :

Haut :représente la longueur du sous espace.

Long :représente la largeur du sous espace .

Step :le pas d'extension de l'arbre d'exploration .

NB_it :c'est le nombre d'itération.

4-quelque méthode utilisé :

1-méthode pour générer des nœud aléatoire :

Cette fonction consiste à générer des nœud aléatoires dans chaque sous espace

Soit

X,Y :coordonnées d'un nœud ;

FonctionGet_Rand_Point()

Debut

```
x=P_ref->getx()+rand()%Long;
y=P_ref->gety()+rand()%Haut;
P=newNoeud(x,y,0,NULL);

Return p ;
```

Fin

4-1-méthode permet de trouvé le nœud le plus proche :

Cette fonction elle come à rôle de trouver le nœud le plus proche à un nœud_P_near
FonctionGet_near_point(nœud*P_rand)

Debut

Calculer la distence entre les point de Graph et la point P_rand

Retourner le nœud le plus proche

Fin

4-2Méthode pour générer parent :

Cette fonction consiste à générer parent dans algorithme RRTStar

`P_new :le nouveau Noeud`

`Distance` :calcule la distance entre deux points

`Cost` :La fonction de coût calcule le coût du chemin d'un point donné le long de chaque bord de l'arbre aléatoire jusqu'au point de départ

Function `chooseparent(Noeud* P_new)`

Debut

Calculer le min entre distance et le cost

```
min =Distance(P_new,T[0])+T[0]->getcost();
```

//C'est choisir celui qui réduit le coût du chemin x_{near-neighbor}。

Fin

5-Méthode rewire :

`Cette fonction consiste à générer` Redéfinir le processus du nœud parent

Fonction rewire(Noeud* P_new)

Debut

```
for (int i=0; i<T.size();i++)
d=Distance(T[i],P_new)+P_new->getcost();
if (d<T[i]->getcost())

    T[i]->setcost(d);
    T[i]->setparant(P_new);
```

Fin

6-méthode sample :

Cette fonction Étant donné deux poses et une valeur heuristique maximale, la fonction L'échantillon renvoie indépendamment et à l'identique distribué (échantillons de l'espace d'état .

```
Sample(float c_max)
  {
    floatc_min = distance(P_init,P_goal);
     floatx_centre = (P_init->getx()+ P_goal->getx()) / 2;
    floaty_centre = (P_init->gety()+ P_goal->gety()) / 2;
    Ogre::Vector2 v;
    v.x=P_goal->getx()-P_init->getx();

    v.y=P_goal->gety()-P_init->gety();
```

```
v.normalise();

      //ofVec2f dir = SMP::goal - SMP::start;
      //dir = dir.getNormalized();
      float angle = std::atan2(-v.y, v.x); //Frame is with y
pointing downwards
      float r1 = c_max / 2;

float r2 = std::sqrt(std::pow(c_max, 2) - std::pow(c_min, 2)) / 2;

      float x = static_cast<float>
(rand())/(static_cast<float>(RAND_MAX/2))-1;   //(-1, 1);
      float y = static_cast<float>
(rand())/(static_cast<float>(RAND_MAX/2))-1;   ///rand()%2-
1;//ofRandom(-1, 1);

      float x2 = x * r1 * std::cos(angle) + y * r2 *
std::sin(angle);
      float y2 = -x * r1 * std::sin(angle) + y * r2 *
std::cos(angle);

      //ofVec2f rot_sample, rot_trans_sample;
      //rot_sample.set(x2, y2);
      //rot_trans_sample = rot_sample + x_centre;
      Float xx,yy;
      xx=x2+x_centre;
      yy=y2+y_centre;


      Noeud* n= new Noeud(xx,yy,0,NULL);
      //n.setx(xx);// = rot_trans_sample.x;
      //n.setx(yy);//n.location.y = rot_trans_sample.y;

      return n;
}
```

7.Création et édition de l'environnement virtuel :

Scène édit :déterminer le espace de l'environnement

Figure 24:création de l'environnement

Object édit :ajouté des objets dans espace et déterminé la nature de chacun (obstacle ou bien entité).

Figure 25 : l'environnement d'origine

8-résultat expérimentaux :

8-1algorithme RRT :

Pour nos simulation nous avons utilisé la même configuration pour notre scène les points de départ et de destination sont gardés fixe

Le tableau résume le temps de calcule de l'algorithme RRT par rapports au nombre d'obstacles imposés.

Nombre d'obstacles	1	2	3	4	5	6	7	8	9	10
Temps de calcule $x10^{-3}$(s)	0.8	1	1.2	3	5	1	11	10	20	90
Lenombre d'itération	30	50	75	90	120	150	200	250	500	700

Tableau 2 :résume le temps de calcule de RRT par rapports au nombre d'obstacles

La représentation graphique de ces résultats est reportée sur la figure suivante ou l'on constate une variation croissante presque linéaire du temps de calcule par l'algorithme RRT avec l'augmentation du nombre d'obstacles .

On constate aussi que le nombre d'itération pour trouver la solution est presque le même malgré l'augmentation des nombres d'obstacles pour donc le nombre d'obstacles n'a pas de grande influence sur la convergence de l'algorithme RRT.

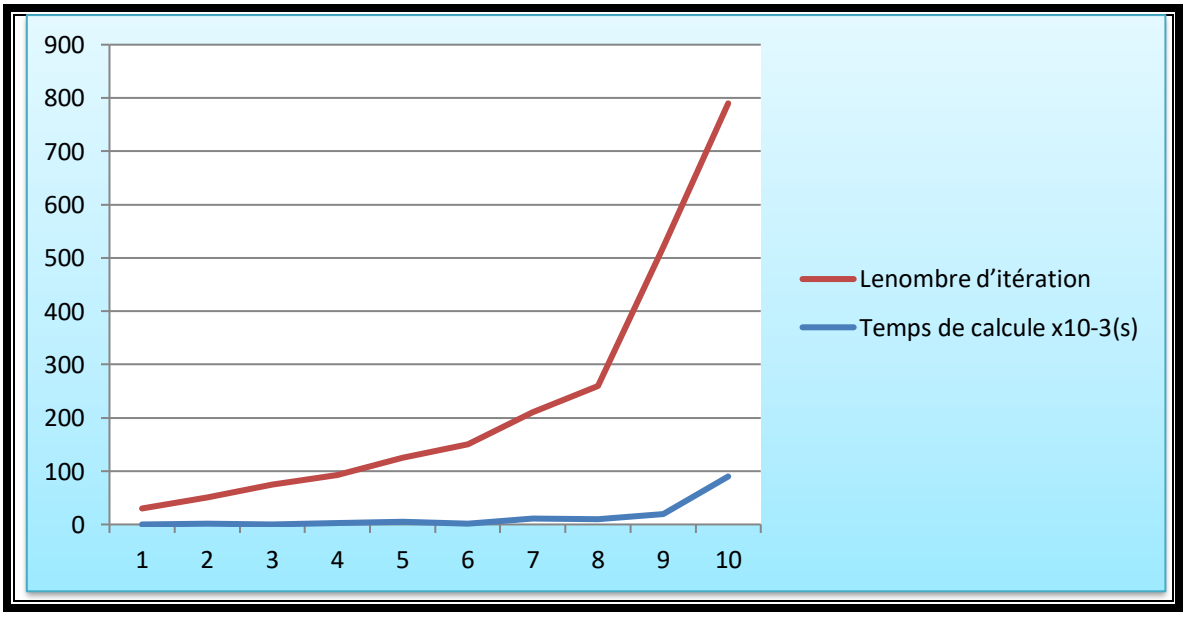


Figure 26:temps de calcule du RRT par rapports au nombre d'obstacles

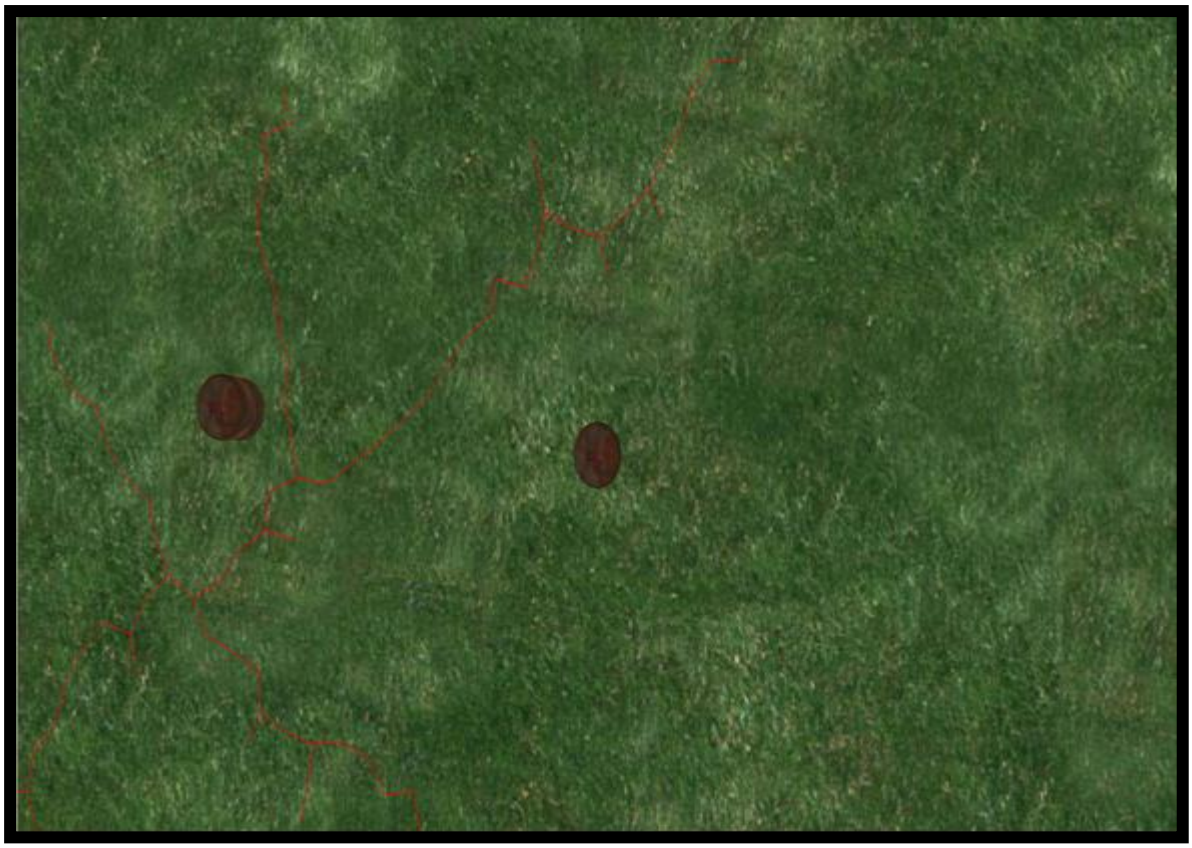


Figure 27:environnement d'origine et chemins planifies par RRT pour 100 itération

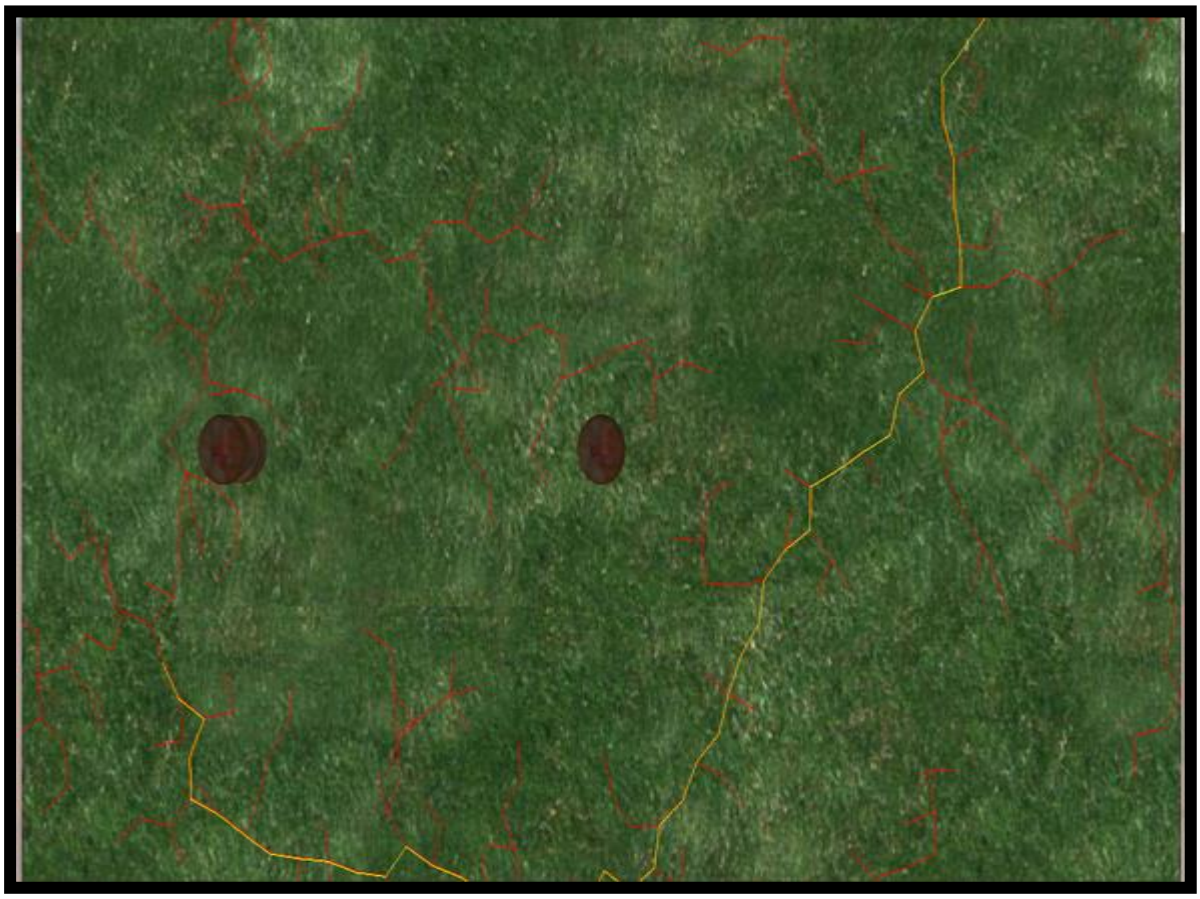


Figure 28:environnement d'origine et chemins planifies par RRT pour 1000 itération

Le tableau 3 donne le temps d'élaboration RRT en fonction du nombre d'itération max dans ce scénario nous augmentons chaque fois le nombre d'itération max avec 1000.

Le nombre d'itérations max	100	200	300	400	500	600	700	800	900	1000
Temps de calcule x10^{-3}(s)	4	10	20	30	40	60	50	30	30	40

Tableau 3:le temps de calcule calcul de RRT par rapport au nombre d'itérations max

La représentation graphique de ces résultats est reportée sur la figure ou l'on constate Variation croissante.

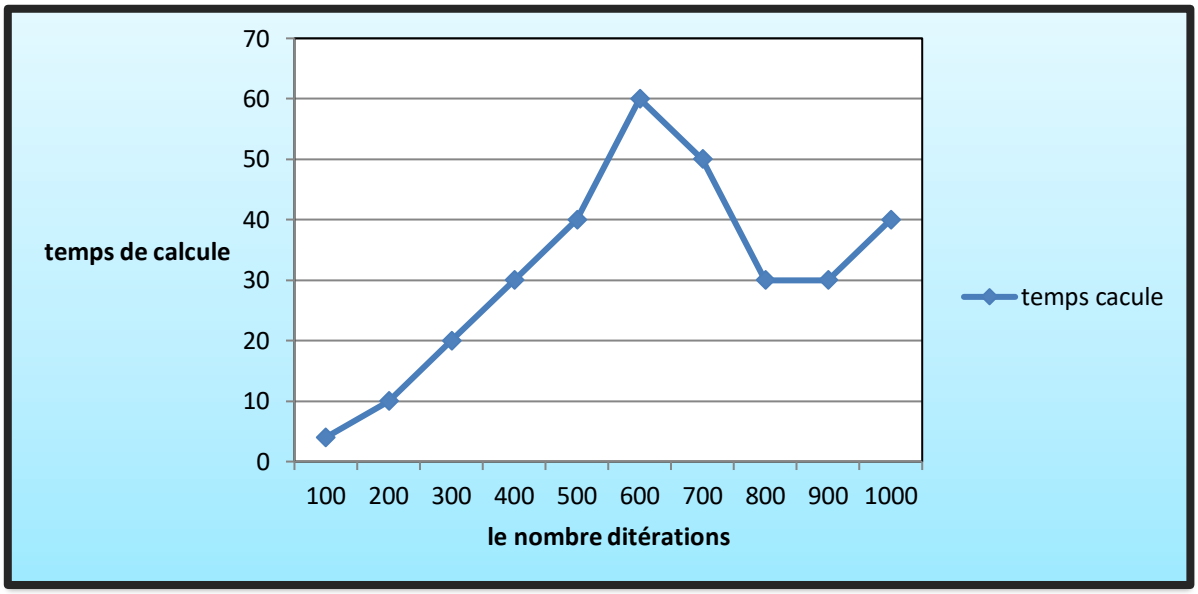


Figure 29:temps de calcule de RRT par rapport au nombre d'itération générés

Le tableau 4 donne le temps d'élaboration RRT en fonction du pas , on fixe le nombre d'obstacles et le nombre d'itération .

Le pas	20	40	60	80	100	120	140	160	180	200
Temps de calcule x 10^{-3}(s)	1	20	40	100	20	10	2	20	1	110

Tableau 4 :résumé le temps de calcule de RRT par rapport le nombre de pas

La représentation graphique de ses résultat est reportée sur la figure 30

On peut interpréter ces résultat en les divisant en deux parties

- partie 1 :variation décroissante lorsque le pas est augmenté de 20 à chaque fois jusqu'à la valeur 100 .cela est expliqué parle fait que à chaque fois on augmente le pas on se rapproche plus rapidement de la destination .
- partie 2 :après la valeur 100 on remarque que le temps de calcule de RRT recommence à croite cela expliquer par un plus de calcule du fait que plusieurs points sont ignorés par le test d'itération avec les obstacles.

Donc il faut bien choisir le pas de l'algorithme RRT(un pas plus petit pour les environnement denses par contre un pas plus grand pour les environnements moins denses d'obstacles).

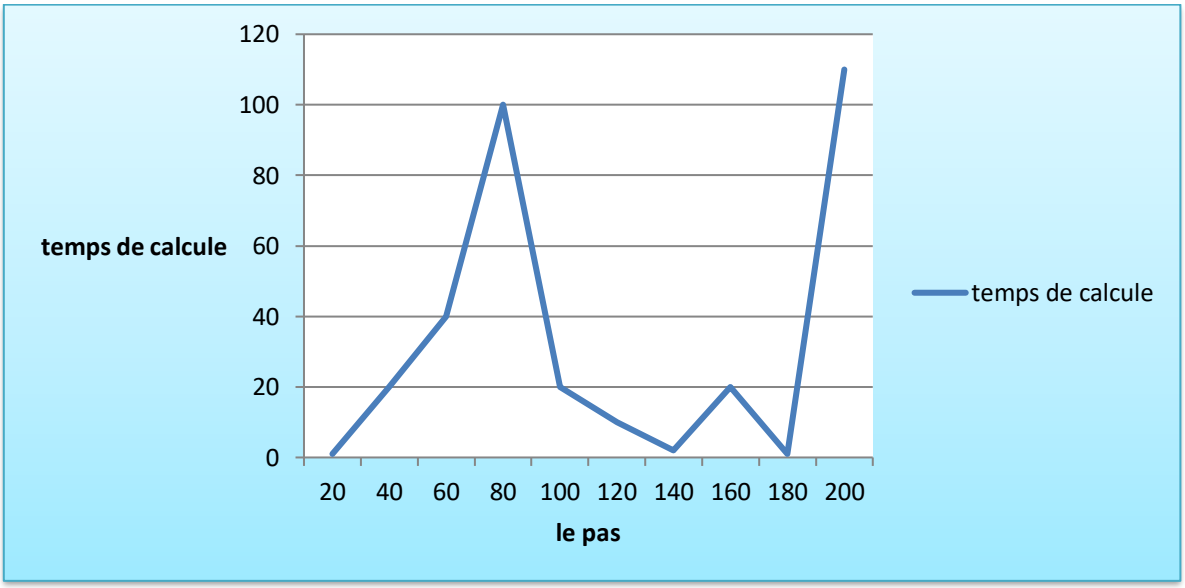


Figure 30 :temps de calcule de RRT par rapport au pas

Figure31:chemins planifiés par RRT le pas =40

8-2 algorithme RRTSTAR :

Le tableau 5résumé le temps de calcule de RRT* et le nombre d'itération généré pour atteindre le but par rapports au d'obstacles imposées à chaque fois que le nombre d'obstacles augmente d'une unité.

Nombre d'obstacles	1	2	3	4	5	6	7	8	9	10
Temps de calcule $x10^{-3}$(s)	1	1.3	4	6	8	13	19	20	80	160
Lenombre d'itération	30	50	75	90	120	150	200	250	500	700

Tableau 5:résume le temps de calcule de RRT* par rapports au nombre d'obstacles

La représentation graphique de ces résultats est reportée sur la figure suivante ou l'on constate une variation croissante presque linéaire du temps de calcule par l'algorithme RRT* avec l'augmentation du nombre d'obstacles .

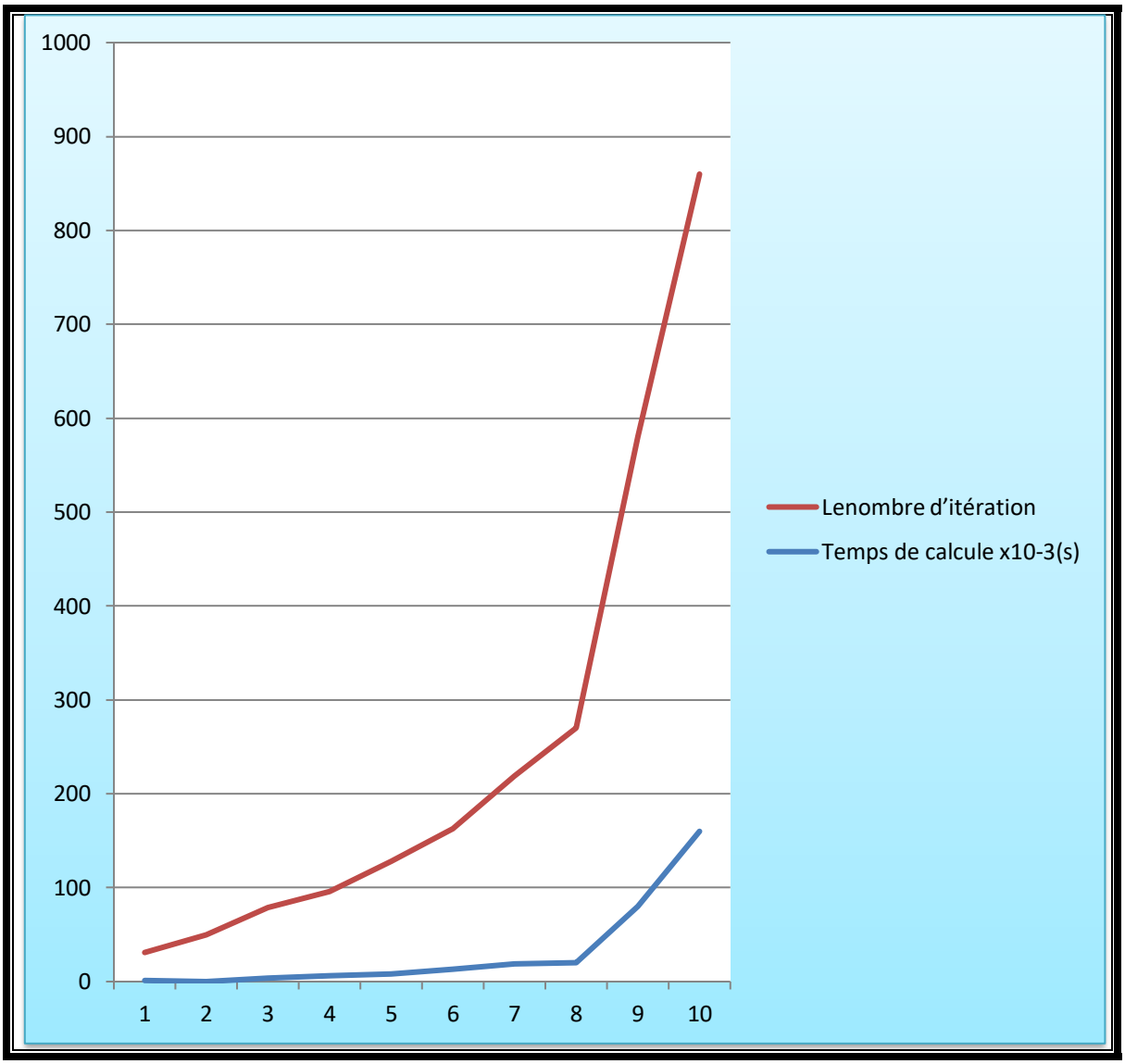


Figure 32:temps de calcule du RRT* par rapport au nombre d'obstacles

Figure 33 :chemins planifiés par RRT*

Le tableau 6 donne le temps d'élaboration RRT *en fonction du nombre d'itération max à chaque fois en augmenté le nombre d'itération 1000 unités

Le nombre d'itérations max	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Temps de calcule x10^{-3}(s)	**7**	**20**	**30**	**60**	**80**	**120**	**160**	**210**	**260**	**320**

Tableau 6 :le temps de calcule calcul de RRT *par rapport au nombre d'itérations max

La représentation graphique de ces résultats est reportée sur la figure 34 ou l'on constate une variation croissante linéaire du temps de calcule par l'algorithme RRT* par rapport au nombre d'itérations max.

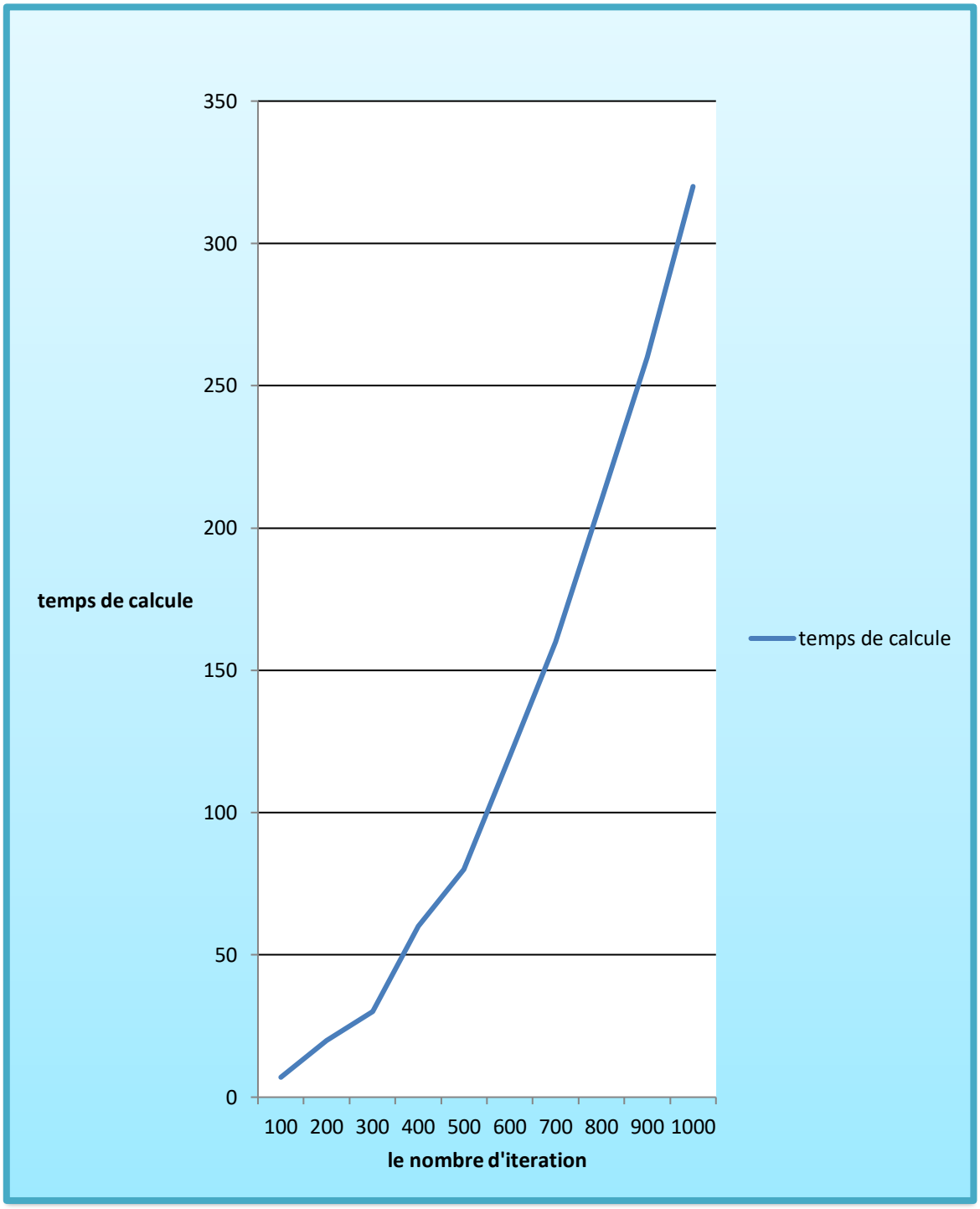


Figure 34:le temps de calcule de RRT* par rapport le nombre d'itérations max

Figure 35 :chemins planifiés par RRT* au nombre d'itérations =1000

8-3 l'algorithme informed RRT STAR :

Le tableau 7 résumé le temps de calcule de informed RRT* et le nombre d'itération généré pour atteindre le but par rapports au d'obstacles imposées à chaque fois que le nombre d'obstacles augmente d'une unité.

Nombre d'obstacles	1	2	3	4	5	6	7	8	9	10
Temps de calcule $x10^{-3}$(s)	2	7	5	6	10	14	28	38	125	200
Lenombre d'itération	50	65	80	95	130	160	250	300	600	1000

Tableau7:résume le temps de calcule de informed RRT* par rapports au nombre d'obstacles

La représentation graphique de ces résultats est reportée sur la figure suivante ou l'on constate une variation croissante presque linéaire du temps de calcule par l'algorithme informedRRT* avec l'augmentation du nombre d'obstacles .

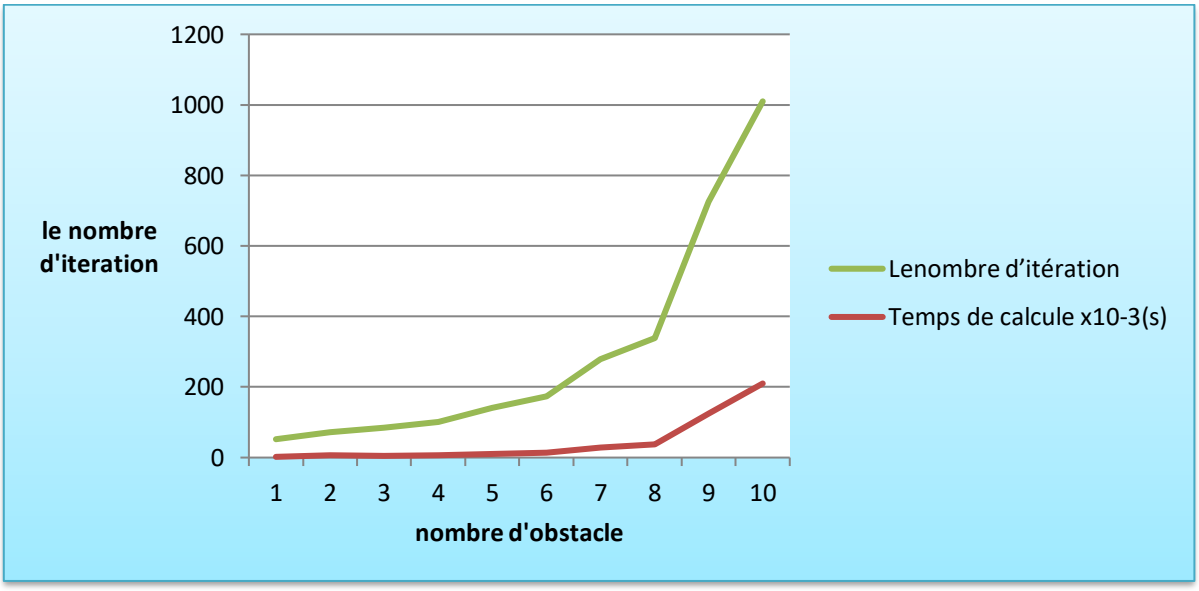


Figure 36 :temps de calcule du informedRRT* par rapport au nombre d'obstacles

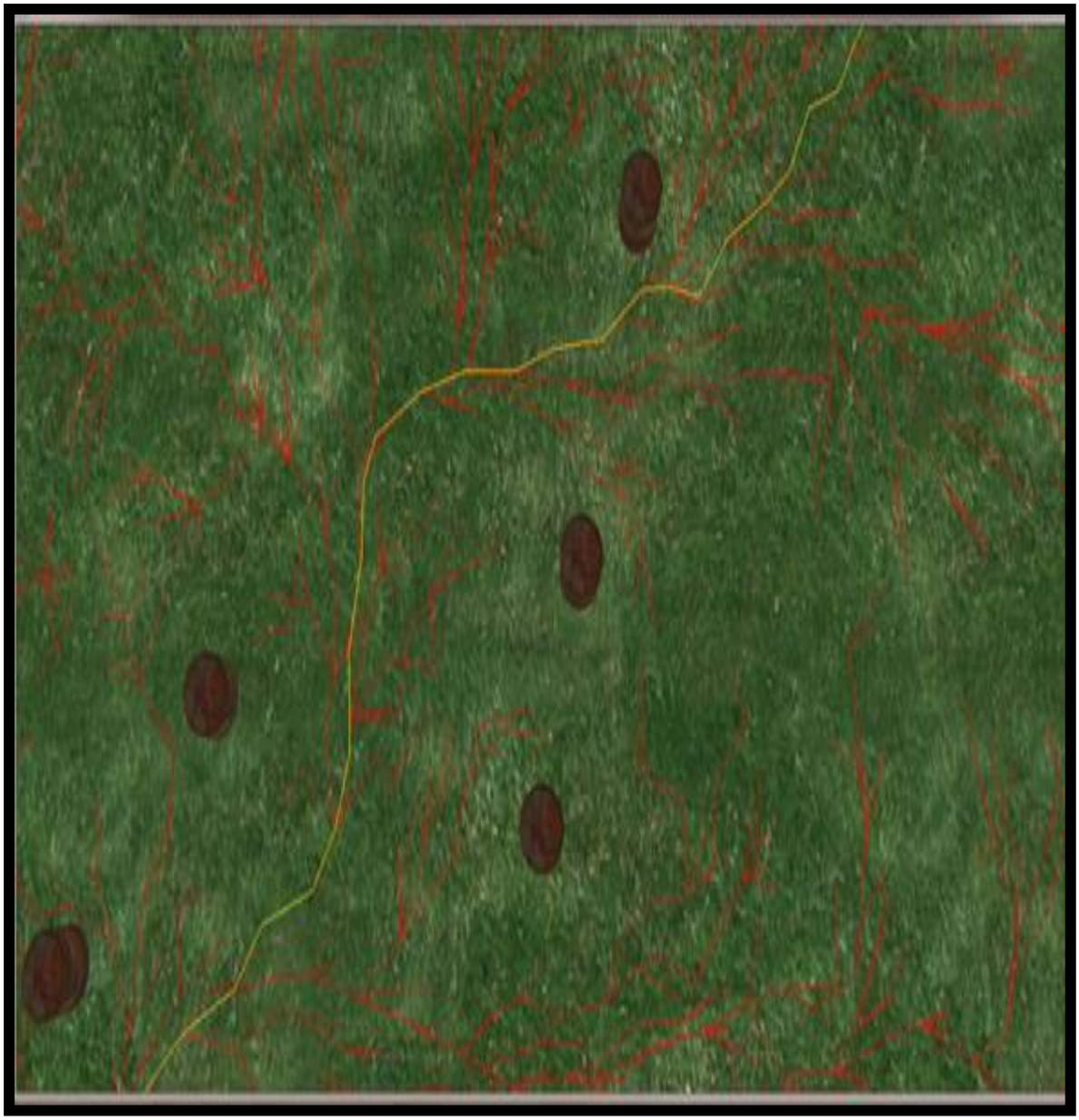


Figure 37 :chemins planifiés par InformedRRT*

Le tableau 8 donne le temps d'élaboration InformedRRT *en fonction du nombre d'itération max à chaque fois en augmenté le nombre d'itération 1000 unités

Le nombre d'itérations max	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Temps de calcule x10^{-3}(s)	**10**	**20**	**40**	**60**	**90**	**128**	**170**	**210**	**250**	**320**

Tableau 8 :le temps de calcule calcul de informedRRT *par rapport au nombre d'itérations max

La représentation graphique de ces résultats est reportée sur la figure 34 ou l'on constate une variation croissante linéaire du temps de calcule par l'algorithme RRT* par rapport au nombre d'itérations max.

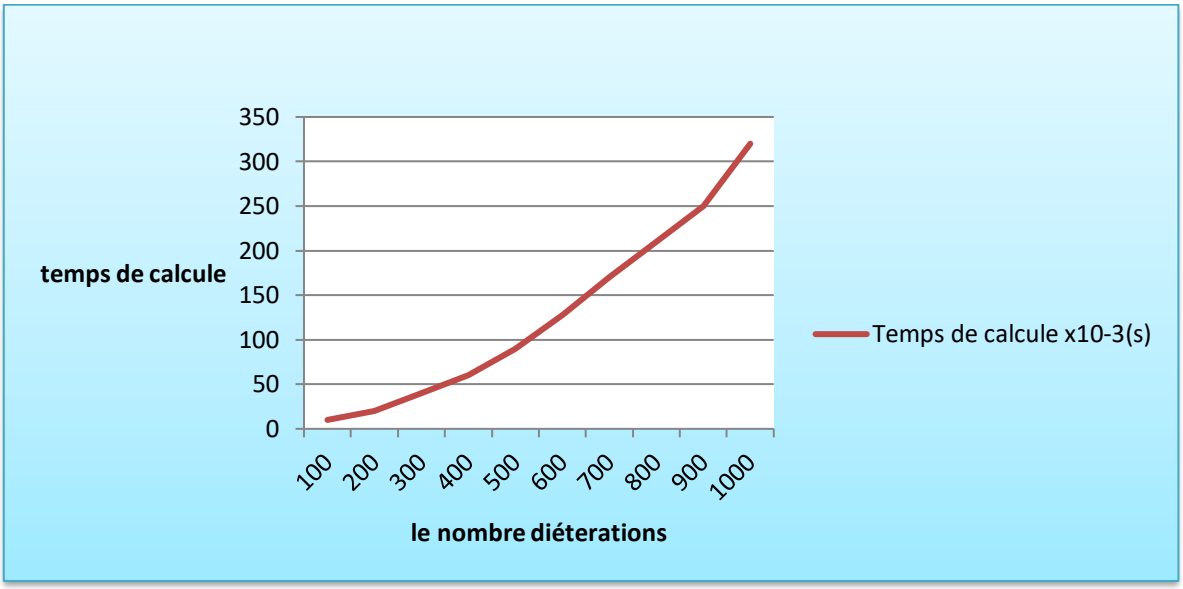


Figure 38 :le temps de calcule de InformedRRT* par rapport le nombre d'itérations max

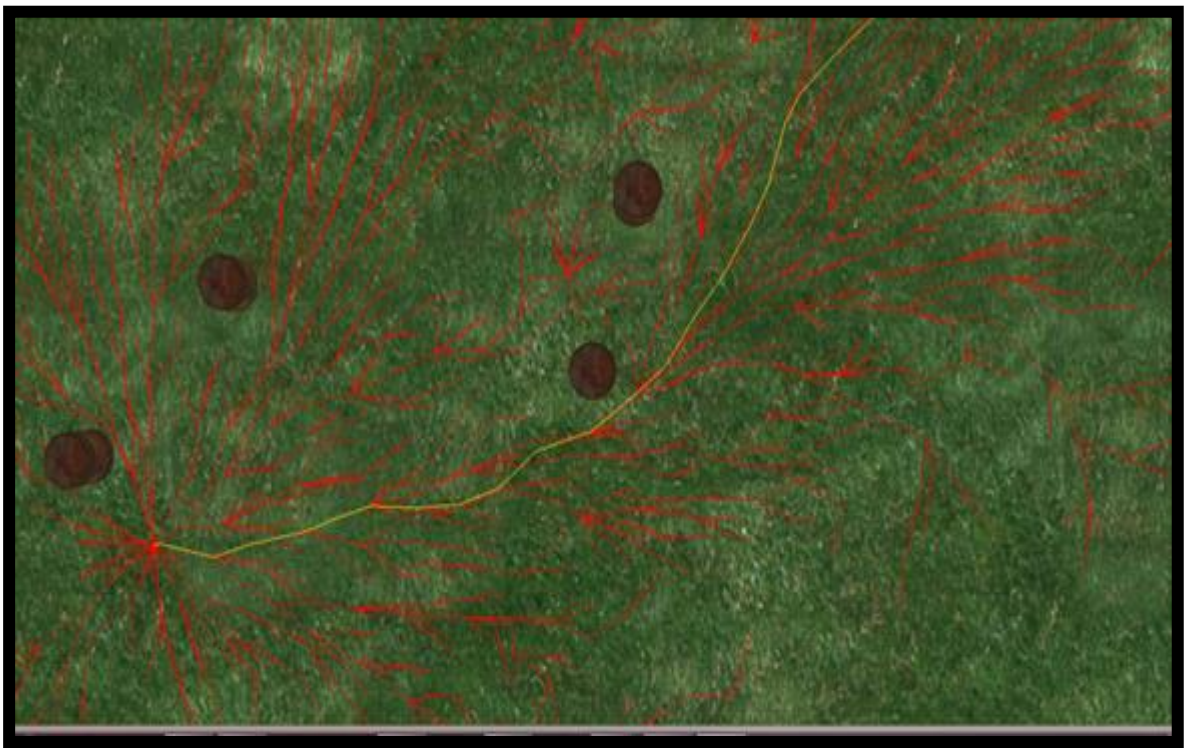
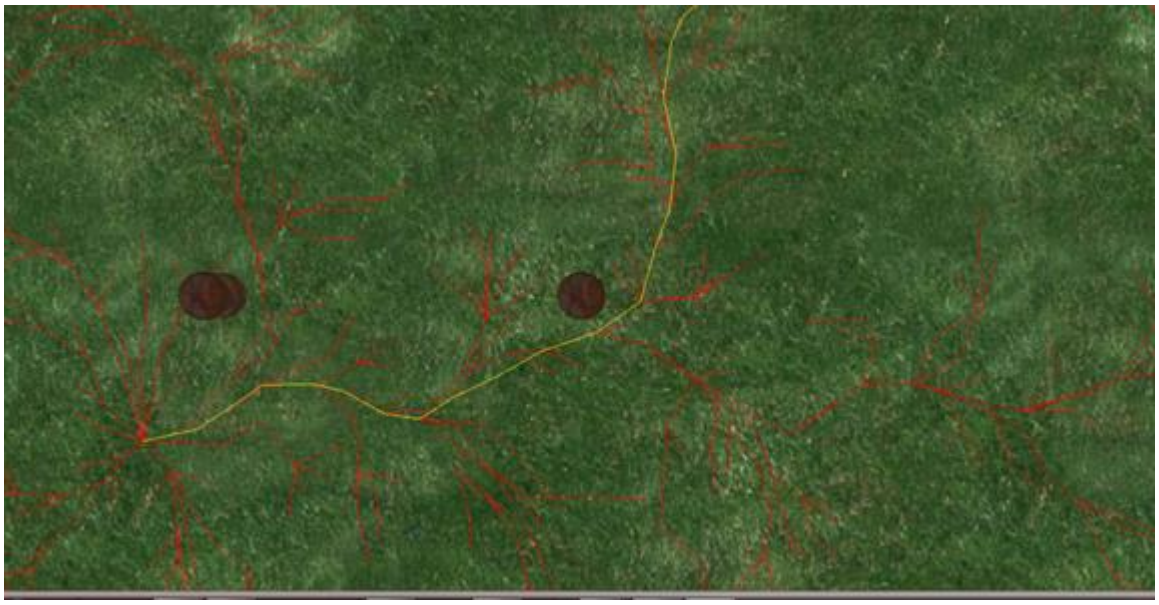


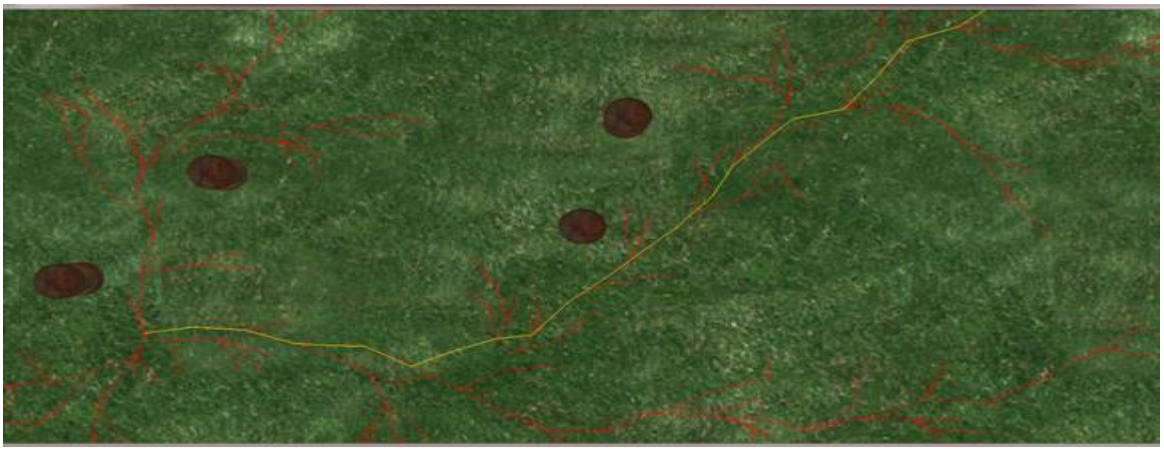
figure 39 :chemins planifies par informed RRT* au nombre d'itération=2500

Dans cette section ,l'analyse de deux algorithme RRT,RRT* ,InformedRRT*est présenté pour évaluer leur performance la figure 40représente le résultat de la comparaison entre RRT et RRT* et InformedRRT*à chaque fois on augmente le nombre d'itération .

a)RRT avec 600 itérations



b)RRT* avec 600 itérations



c)Informed RRT *avec 600 itérations

figure 40 :résultat de la comparaison entre RRT et RRT* et Informed RRT*

e)RRT* avec 800 itérations

f)Informed RRT* avec 800 itérations

g)RRT* avec 2000 itérayions

h)Informed RRT* avec 2000 itérations

figure41 :résultat de la comparaison entre RRT* et Informed RRT*

9-Conclusion :

Nous avons présenté les déférente résultat des algorithmes (RRT,RRT*,INFORMED RRT*) , on choix la langage de programmation utilisé pour implémenter notre système donc on faite la comparaison entre les déférent résultat entre les 3 algorithme , la comparaison entre le temps de calcule et le nombre d'itération et le de chemin planifiés .

10-Conclusion et perspectives :

Nous avons traité au sein de ce document la problématique de la planification de chemin . Cette problématique est renforcée dans notre cadre d'application par la nécessité de tenir compte des besoins d'interaction des humains.

Le choix de notre thème qui est la recherche d'un chemin optimum dans un monde virtuel et à moindre coût d'effort, nous conduit à l'utilisation de toutes les techniques mises au point par les chercheurs dans ce domaine.

Notre approche consiste à l'élaboration de la scène on utilise les algorithme de famille RRT surtout algorithme informed RRT* ,nous a permis à la construction d'un graphe sur lequel on a appliqué les dernières techniques de calcul pour arriver au résultat attendu qui est le chemin optimum mais notre robot, au cour de son déplacement rencontre des obstacles qu'il faut les résoudre.

Pour réalisé cette application nous avons basé sur :

- une planification en utilise les algorithme RRT-RRT*-INFORMED RRT*.
- Le rendu en utilisant Ogre 3D.

Les résultat obtenus sont satisfaisant du point de vue temps de calcule et qualité du chemin obtenu .Néanmoins ,notre projet pourra étre amélioré par l'ajouté d'autre fonctionnalités comme :

- Utiliser une fonction de lissage de chemin obtenu.

Bibliographie

Bibliographie

[1] A.Bouguetitiche , ‘’modélisation topologique et sémantique le l’environnement ‘’,mémoire de magister en informatique ,option synthèse d’image et vie artificielle ,université de mohamed khider biskra 2011.

[2] Marc Shakour ‘’conception et implémentation d’un algorithme de planification de chemin dans un jeu vidéo comportant en environnement triangularisé’’ ,mémoir de maitrise en informatique université du Quebec à montérial ,septembre 2012.

[3] Thomas lopez ,"planification de chemin et adaptation de posture en Environnement dynamique "thèse de doctorat de l’INSA,de rennes université Européenne de bretagne mars 2012.

[4] Ali Gagui , "Navigation réactive d’humain virtuel par planification locale "Mémoire de magister en informatique option d’image et vie artificielle Université de mohamed khider Biskra 2012.

[5] F.lamarche , "Humanoides virtuel ,réaction et cognition :une architecture Pour leur autonomie ",thèse de Doctorat Université de Rennes I, décembre 2003

[6] S.Thrun et A.Bucken , "Intergrating grid –based and topological maps for Mobile robot navigation",proceedingof the AAAI thirteenth National Conference on Artificiel Intelligence ,pages 944-951 ,AAAI Press/MIT Press 1996.

[7] W.SHao , "Animating Autonomous Pedestrians ",Thèse de Doctorat ,Institude of Mathematical Sciences ,Université de New York 2006.

[8] F.Tecchia ,C.loscos et Y.chrysanthou, "Visualizing Crowds in real-time ",Computer Graphics Forum ,pages 753-765-2002.

[9] R.andersen,J.LBerrou,et Alex Gerodimos , "on somme limitations of grid-based (ca)Edestrian simulation models",First International Workshop on Crowd Simulation (V-Crowds’05),VRlab,EPFL 2005 .

[10] S.Paris ,"Caractérisation des niveaux de services et modélisation des circulation De personnes dans les lieux d’échanges ",thèse de Doctorat Université de Rennes I 10 octobre 2007.

[11] J.Barraquand et J-C latombe , "Robot motion planning : a distributed representation Approach ",International Journal of Robotics Research pages 628-649 1991.

[12] S.caselli,M.Reggiani et R.Rocchi"Heuristic methods for randomized path planning in Potential Fields " 2001.

[13] C.Gloor ,P.Stucki et k.Nagel , "Hybrid technique for pedestrian simulation .4th swiss Transport Research Conference ",Monte Verita ,Ascona 2004.

[14] H.jiang ,W,Xu ,T.Mao ,C.li,S.Xia and Z.wang "A semantic environnement model for Crowd simulation in multilayered complex environnement ", Proceeding of the 16th ACM Symposium on virtuel Reality software and Technology , VRST '09, pages 191-198 , New York , NY ,USA,2009 ACM.

[15] j.Valentin ,"simulation du comportement humain en situation d'évacuation de Bâtiment en feu ",thèse Doctorat, Université de Pau et des pays de l'adour 3 avril2013

[16] JedLengyel ,Mark Reichert ,Bruce R.Donald et ,Donald P.Greenberg, "Realtime robot Motion planning using rasterizing computer graphics hardware ",dans SIGGRAPH '90. Preceeding of the 17th annual conference on computer graphics and interactive Technique,pages 327-335.ACM Press ,1990 1.5.4

[17] Nilsson (N.J), "Principles of artificial intelligence ",Springer –Verlag Berlin Heidelberg New York 1982.

[18] B.Logan et N.Alechina , "A* with boudedcosts",proceeding of the Fifteenth National Conferenceon Artificiel Intelligence (AAAI-98) pages 444-449,1998 1.5.4.

[19] MichaelE.Bratman , "Faces of intention ",Cambridge Studies in PHilosophyCambridgeUnversity Press 1999 2.1.2.

[20] AdiBotea,Martinmuller et Jonathan Schaeffer , "Near optimal hierarchical pathfinding ",journal of Game Development ,1(1) :7-28,2004.1.5.4

[21] ReynoldsCW.Flocks,Herds,and Schools : "A Distributed Behavioral Model ",computer Graphics 21,1987 ;4 :25-34.

[22] Craig w.Reynolds , "Steering Behavrios for Autonomous Characters " :Game developers Conference 1999.

[23] D.Helbing ,I.farkas ,and T.Vissek, "simulating dynamical feature of espace panic "Nature ,407 :487-490,2000 article f.lamarch.

[24] Franck Feurtey ,"Simulating the collision avoidance behavior of pedestrains"Department of Electronic Engineering 2000.

[25] SoterisStylianou and YiorgosChrysanthou , "Crow self-organisation streaming an Short path smoothing ",journal of WSCG –science press ,14 2006.

[26] Fabricelamarche and StéphaneDonikian, "Crowd of virtual humans :a new approach For real time navigation in complex and structured environnements",eurographics,2004

[27] Frey P.et George P,"le maillage Facile ",Hermès Science Publication Paris ,paris2003.

[28] N.M Amato and Y.Wu ,A randomized roadmap method for path manipulation planning In Proc .IEEE int .Conf .Robot Autom (ICRA).1996,pp.{113.120}.

[29] La Valle S.M Rapidly –exploring random trees :a new tool for path planning .Computer Science departement ,Iowa state University .On –line (June 25.2010).

[30] Karaman ,S. and frazzoli ,E :incremental sampling –base Algorithme for optimal motion Planning .2010.

[31] Dalibard. S .Nakhaei ,A,Lamiraux ,F.& Laumond ,J-P (2009) ,Whole –body task planning For a hummanoid robot :a way to integrate collision avoidance ,in 'humanoid Robots 2009 .Hamanoids 2009 .9th IEEE-Ras International Conference on ' ,pp.355-360.

[32] Kavraki,L ,Svestka ,P,Latombe,J and Overmars ,M :Probabilistic roadmaps for path Planning in high-dimensional configuration spaces .1996.

[33] S.DSteck et H. A. Mallot .the role of global and local landmarks in virtuelenvironnement,pages 69-83,2000.

[34] P.Doyle Believability through context using "Knowledge in the world " to create intelligent characters .In Proceeding of the first internatinal joint conference on Autonomous agents and multiagent systems ,page 342-349.ACM press,2002.