e

**Robotique**

Projet robotique

Enseignant : M.Franco Robles**Sommaire**

1. **Introduction ………………… 3**
2. **Simulation dans Gazebo d'un Lieu Réel ………………… 4**

Choix du lieu

Modélisation de l'environnement

1. **Construction de Carte SLAM ………………… 5**

Configuration initiale

Génération de la carte

Résultats et optimisation

1. **Analyse des Algorithmes de Navigation ………………… 6**

Algorithme A\*

Algorithme de Dijkstra

Greedy Best-First Search

Comparaison des performances

1. **Motion Planning avec Contraintes Dynamiques ………………… 9**

Scénario implémenté

Dynamic Window Approach

1. **Conclusion et Perspectives ………………… 10**

**Introduction :**

La robotique moderne ne se limite pas à la simple programmation de mouvements mécaniques, elle implique une compréhension approfondie des contraintes inhérentes au robot et à son environnement.

Dans l’objectif de valoriser les connaissances acquises dans ce domaine, nous avons réalisé durant une semaine un projet mettant en avant deux aspects essentiels : la planification de trajectoire en tenant compte du terrain (path planning) et la planification du mouvement en intégrant les contraintes physiques du robot (motion planning).

Pour mener à bien ce projet, une connaissance approfondie de l’environnement ROS (Robot Operating System) était indispensable. ROS est un framework open source destiné à la programmation de robots. Il fournit un ensemble d’outils, de bibliothèques et de composants facilitant une programmation fluide et modulaire. Son fonctionnement repose sur une architecture distribuée, dans laquelle les différentes fonctionnalités du robot (comme la navigation ou la détection d'obstacles) sont réparties en nœuds qui communiquent entre eux via un système de topics ou de services. Cette approche permet de concevoir des systèmes complexes, évolutifs et facilement testables.

Une image contenant texte, capture d’écran, diagramme, Police

Le contenu généré par l’IA peut être incorrect.

**Simulation dans Gazebo d'un Lieu Réel**

La première phase du projet a consisté à modéliser un environnement réel dans le simulateur Gazebo. Le lieu choisi est la salle de classe 207. Bien qu’une reproduction parfaite n’ait pas été nécessaire, il était crucial de capturer les caractéristiques essentielles de l’espace, telles que la disposition des tables, des chaises et des zones de circulation dans l’objectif d’obtenir une simulation qui permettrais de démontrer les capacités du robot dans un environnement réel. Il faut recréer cet environnement réel dans le simulateur Gazebo avec des obstacles

Une image contenant capture d’écran, Modèle réduit, conception

Le contenu généré par l’IA peut être incorrect.La modélisation a débuté par la création des structures de base (murs, sol) à l’aide des outils natifs de Gazebo. Les obstacles statiques, comme les chaises, ont été intégrés à partir de modèles standards, tandis que les tables, plus complexes en raison de leur surface hétérogène (ordinateurs, fournitures), ont été simplifiées en retirant leurs textures. Pour ajouter une dimension réaliste, des personnages 3D simplifiés ont été introduits comme obstacles.

La simulation finale a offert un environnement fonctionnel et réaliste pour les étapes suivantes.

**Construction de Carte SLAM**

Avec l’environnement simulé prêt, l’étape suivante a été de générer une carte précise à l’aide de l’algorithme SLAM (Simultaneous Localization and Mapping). Le TurtleBot, équipé d’un capteur LiDAR virtuel, a été déployé dans la salle 207 simulée. Le package gmapping de ROS a permis de fusionner les données de localisation et de détection d’obstacles en temps réel.

Une image contenant capture d’écran, texte, carré

Le contenu généré par l’IA peut être incorrect.

**Analyse des Algorithmes de Navigation**

La planification de trajectoire est un élément central de la navigation autonome. Pour évaluer les différentes approches, nous avons testé trois algorithmes classiques : A\*, Dijkstra et Greedy Best-First Search (Greedy). Chacun présente des caractéristiques distinctes en termes d'optimalité, d'eff icacité et d'adaptabilité aux contraintes environnementales.

Une image contenant texte, capture d’écran, diagramme, conception

Le contenu généré par l’IA peut être incorrect.Algorithme A\*

L'algorithme A\* se distingue par son approche équilibrée entre exploration et optimisation. En combinant le coût réel du déplacement depuis le point de départ en estimant, grâce à la distance, le coût restant jusqu'à l'objectif. Il garantit toujours une solution optimale tout en limitant l'espace de recherche. Dans nos tests, A\* a démontré une grande fiabilité, calculant un trajet optimal dans une échelle de temps raisonnable. Cependant, son temps de calcul reste sensible à la complexité de l'environnement, notamment lorsque de nombreux obstacles obligent à explorer des alternatives.

Une image contenant texte, capture d’écran, graphisme, conception

Le contenu généré par l’IA peut être incorrect.Algorithme de Dijkstra

Contrairement à A\*, l'algorithme de Dijkstra ne repose sur aucune heuristique : il explore systématiquement tous les chemins possibles par ordre de coût croissant depuis le point de départ. Cette exhaustivité lui permet de garantir l'optimalité du chemin, quelle que soit la configuration de l'environnement. Dans notre salle de classe simulée, Dijkstra a effectivement trouvé le même chemin optimal qu'A\*, mais avec un temps de calcul significativement plus long. Cette lenteur s'explique par sa complexité algorithmique qui le rend peu adapté aux grandes cartes ou aux systèmes nécessitant des réponses en temps réel.

Une image contenant texte, capture d’écran, Police, diagramme

Le contenu généré par l’IA peut être incorrect.Greedy Best-First Search

À l'opposé de Dijkstra, le Greedy Best-First Search privilégie la rapidité en se focalisant exclusivement sur la proximité à l'objectif via une fonction heuristique. Bien qu'il soit capable de générer des chemins rapidement (Jusqu’à plusieurs fois plus vite qu'A\*), cette efficacité s'est souvent faite au détriment de la qualité. L'algorithme est moins précis dans ces trajectoires souvent plus longues que nécessaire, notamment en choisissant des détours. Ces chemins trop optimistes ont conduit le TurtleBot à se rapprocher dangereusement des obstacles compromettant la sécurité de la navigation.

Une image contenant texte, diagramme, nombre, ligne

Le contenu généré par l’IA peut être incorrect.

Synthèse Comparative

|  |  |  |  |
| --- | --- | --- | --- |
| Critère | A\* | Dijkstra | Greedy |
| Optimalité | Oui | Oui | Non |
| Temps de calcul | Moyen | Elevé | Faible |
| Longueur du chemin | Optimal | Optimal | Non-optimal |
| Adaptabilité | Environnements complexes | Cartes statiques | Espaces ouverts |

Cette analyse met en lumière les compromis inhérents au choix d'un algorithme. Si A\* s'impose comme un bon compromis pour la plupart des scénarios, Dijkstra reste utile pour des applications exigeant une précision absolue, tandis que Greedy peut être envisagé pour des tâches prioritaires à la rapidité, sous réserve d'une supervision accrue. Ces résultats guideront nos choix dans les phases ultérieures de navigation autonome et de motion planning.

Une image contenant diagramme, capture d’écran, Plan, texte

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

**Motion Planning avec Contraintes Dynamiques**

La planification de mouvement va au-delà de la simple trajectoire en intégrant les contraintes physiques du robot.

Comparé au path planning classique, le motion planning offre des trajectoires plus fluides et adaptatives, mais exige davantage de ressources computationnelles. Cette approche est donc idéale pour des environnements dynamiques, tandis que le path planning reste efficace dans des espaces statiques prévisibles.

Le Dynamic Window Approach est une stratégie d'évitement des collisions en ligne pour les robots mobiles. Il correspond à un algorithme de planification de mouvement local, utilisé pour faire naviguer un robot en temps réel à partir d’une génération d'un espace de recherche valide ainsi que la sélection d'une solution optimale dans cet espace.​

Elle prend en entré les vitesses linéaires et angulaires​ et les contraintes cinématiques​ du robot avec la présence d’obstacles dynamiques.

Une image contenant Graphique, graphisme, dessin humoristique, capture d’écran

Le contenu généré par l’IA peut être incorrect.

**Conclusion et Perspectives**

Ce projet a permis de maîtriser les outils clés de la navigation robotique sous ROS, de la modélisation d’environnements à l’optimisation des algorithmes. Les principaux apprentissages incluent l’importance de l’étalonnage des paramètres SLAM et le compromis entre rapidité et sécurité dans le choix des algorithmes.

Pour l’avenir, l’intégration de techniques d’apprentissage automatique pourrait améliorer l’évitement d’obstacles dynamiques, tandis qu’une collaboration interdisciplinaire avec des experts en optimisation renforcerait l’efficacité des solutions.

Lien vers le dépôt git :

[Depot Git](https://github.com/stevendemayo/Projet-Robotique)