Détermination de trajectoires rapides et sûres pour robots

François-Xavier Courel

Les compétitions de robots sont fascinantes par leur complexité : ces derniers doivent en effet se déplacer jusqu'à l'arrivée le plus rapidement possible, mais en évitant toute collision avec un obstacle, qui serait fatale à de telles vitesses.

Je me suis donc intéressé à un algorithme permettant de trouver le plus court chemin entre deux points évitant les obstacles, utilisant les diagrammes de Voronoï et l'algorithme de Fortune.

Positionnements thématiques

— INFORMATIQUE (Informatique théorique)
— INFORMATIQUE (Informatique pratique)
— MATHEMATIQUES (Mathématiques appliquées)

Mots-clés

Courses de robots / Robot racing
Diagramme de Voronoï / Voronoi diagram
Algorithme de Fortune / Fortune's algorithm
Recherche du plus court chemin / Shortest path problem
Graphe planaire / Planar graph

Bibliographie commentée

Depuis les débuts de la robotique, des courses de robots sont régulièrement organisées. Parmi celles-ci, *Micromouse* [1] voit s'affronter des robots souris dans des labyrinthes. Les différentes éditions ont vu les rapides progrès de la robotique. Si les problématiques des premières éditions étaient principalement liées à la cartographie des labyrinthes et à un simple calcul de chemin, les vitesses atteintes par les robots, avec des accélérations de l'ordre de $2g$ [1], ont nécessité des adapations. Parmi celles-ci, ces derniers doivent absolument éviter les collisions avec les murs [2] qui détruiraient le matériel, souvent fruit de longues années de conception.

Les algorithmes de recherche de chemin doivent alors prendre en compte la position des obstacles pour déterminer les chemins s'en éloignant le plus possible. Or, les algorithmes classiques de recherche de chemin, comme l'algorithme de Dijkstra ou A*, ne prennent pas en compte ce paramètre. Vient alors la nécessité de trouver une modélisation pouvant résoudre ce problème.

Les diagrammes de Voronoï sont des outils mathématiques permettant de déterminer des zones de proximité, en créant des cellules de plus proches points autour de points souches, ce qui constitue un pavage du plan. Les arêtes qui délimitent les cellules correspondent alors aux points équidistants entre deux points souches. Leurs applications aux problèmes de planification de mouvements ont rapidement étés saisis par les chercheurs [3]. Cette particularité nous est en effet très utile pour déterminer les chemins les plus sûrs, puisqu'en plaçant les obstacles comme points souches, les arêtes des cellules délimitent les chemins les plus sûrs, c'est à dire les chemin s'éloignant des obstacles [4]. Ces arêtes forment alors un graphe dans lequel on peut alors appliquer les algorithmes classiques de recherche de chemin.

Si ce procédé permet de s'assurer que la navigation soit en zone sûre, il a en outre l'avantage d'être efficace puisque le graphe calculé est bien plus simple qu'un plan de la région. Le temps de calcul de ce graphe est certes plus long que d'appliquer naïvement une recherche de chemin, mais il n'est fait qu'une fois et optimise largement les temps de calculs des requêtes de navigation qui suivent [5].

L'algorithme utilisé pour le calcul du diagramme de Voronoï est l'algorithme de Fortune [6] qui possède la meilleure complexité possible pour cette opération, qui en $O(n \log n)$ où n est le nombre de points souches. Il utilise diverses structures de données [7], comme les arbres binaires de recherche équilibrés, les files de priorité et les listes d'arêtes doublement chaînées, abrégées en DCEL, qui permettent de représenter des graphes planaires [8], la structure utilisée pour représenter le diagramme de Voronoï construit.

En revanche, cette méthode ne peut garantir l'optimalité du chemin trouvé, puisque les arêtes du diagramme de Voronoï ne représentent qu'une partie du plan. Plus les obstacles sont espacés, plus les contournements pour passer exactement entre deux obstacles sont longs et inutiles. Pour cela, il est possible d'optimiser et d'arrondir le chemin trouvé par optimisations itérées, tout en conservant la sûreté du chemin trouvé grâce à un paramètre définissant une distance de sécurité minimale à respecter avec les obstacles [5]. Toutefois, dans les cas où la zone de navigation est restreinte et laisse peu de marges, ce qui est le cas dans les labyrinthes de *Micromouse*, le chemin obtenu s'éloigne peu du résultat réellement optimal est est donc suffisant pour les applications pratiques.

Il se pose aussi la question de la modélisation des obstacles : une modélisation ponctuelle des obstacles ne suffit pas pour représenter des obstacles réels qui s'apparentent à des courbes. Il est alors nécessaire de modéliser les obstacles comme des polygones plus ou moins détaillés [4], ce qui augmente le coût de l'algorithme de Fortune.

Problématique retenue

Pour améliorer les performances et réduires les risques lors de courses de robots, comment déterminer, à l'aide de diagrammes de Voronoï, le meilleur chemin entre deux points qui soit sûr et rapide ?

Objectifs du TIPE

J'ai décidé de réaliser une implémentation complète de l'algorithme de Fortune qui permet de claculer le diagramme de Voronoï d'un ensemble de points souches. Cette implémentation pourra ainsi, en modélisant les données du problème, exploiter le diagramme produit, qui est un graphe planaire pour déterminer le meilleur chemin entre deux points. J'examinerai alors les performances de cet algorithme en fonction de la taille de l'ensemble de points souches et de la complexité de la modélisation des obstacles, pour le comparer à des méthodes plus naïves pour résoudre ce problème. Je propose enfin de comparer les résultats obtenus avec les chemin sûrs et optimaux théoriques.

Liste des références bibliographiques

— **[1]** Wikipedia, *Micromouse*, `fr.wikipedia.org/wiki/Micromouse`, 15/01/2024
— **[2]** Martin Komák, Elena Pivarčiová. *Creating a Simulation Environment for the Micromouse.* TEM Journal (2022), Volume 11, Issue 1, pages 479-483, ISSN 2217-8309, DOI : 10.18421/TEM111-61.
— **[3]** INRIA, *Diagrammes de Voronoï*, `www-sop.inria.fr/prisme/fiches/Voronoi/`, 23/12/2023
— **[4]** E. Magid, R. Lavrenov, I. Afanasyev. *Voronoi-based trajectory optimization for UGV path planning.* 2017 International Conference on Mechanical, System and Control Engineering, (2017), St Petersburg, Russia
— **[5]** P. Bhattacharya, M.L. Gavrilova. *Voronoi diagram in optimal path planning.* 4th International Symposium on Voronoi Diagrams in Science and Engineering, (2007), Glamorgan, United Kingdom.
— **[6]** M. De Berg, *Computational geometry : algorithms and applications.* Springer Science & Business Media (2000), ISBN-978-3-540-77973-5, DOI : 10.1007/978-3-540-77974-2
— **[7]** Université de Liège, *Géométrie Algorithmique*, `www.cgeo.uliege.be/CG/CG_07.pdf`, 22/10/2023.
— **[8]** Simon Fraser University, *Doubly Connect Edge List (DCEL)*, `www2.cs.sfu.ca/~binay/813.2011/DCEL.pdf`, 25/10/2023

DOT