Navigation intérieure - création de graphes routiers intérieurs

Maxime Callico[1,2], Jean Carrière[1], Benoît Darties[2], Rodolphe Giroudeau[2]

[1] Web Geo Service, Montpellier, France
jcarriere@webgeoservices.com mcallico@webgeoservices.com
[2] LIRMM - CNRS UMR 5506 - Montpellier, France
benoit.darties@lirmm.fr rodolphe.giroudeau@lirmm.fr

1 Introduction

La navigation en intérieur est un problème dont l'importance augmente de jour en jour sachant que de plus en plus de centres commerciaux, d'aéroports ou encore de musées évoluent avec notre société et deviennent progressivement plus complexes. La figure 1 illustre un exemple de cas d'utilisation pour la navigation en intérieur. Notre étude concerne en particulier la création de graphes pour ces recherches de routes en intérieurs. Actuellement, notre entreprise reçoit les plans des lieux dont la navigation en intérieur est demandée et les données relatives aux plans (gares, les magasins commerciaux ...) sont créées manuellement. La stratégie utilisée est de placer les sommets tous les x-mètres (selon la taille du graphe), ce qui conduit à un graphe spécialement dense. Le travail occupe plusieurs personnes sur plusieurs jours. Notre motivation est d'automatiser, d'accélérer ce processus en utilisant des algorithmes efficaces de division de polygones pour permettre de réduire la charge de travail sur ce sujet.

2 Objectifs

Pour créer un graphe en intérieur, notre objectif est de subdiviser l'espace traversable, délimité par un polygone extérieur et des obstacles représentés par des trous dans ce polygone, en un ensemble d'espaces convexes. Ces subdivisions, réalisées à l'intérieur du polygone, sont appelées des portails. On peut transformer cette subdivision en un graphe en plaçant un sommet à chaque portail et en construisant un graphe de visibilité à partir de ces sommets. Nous allons examiner un algorithme [2] qui se base sur le concept de la Navigation Mesh, également appelée Navmesh. L'objectif de cet algorithme est de diviser un polygone sans intersection en une partition de polygones convexes. Il convient de souligner que le système Navmesh est largement employé dans l'industrie des jeux vidéos[3]. Notre intention est de l'adapter et le dédier spécifiquement au contexte de la navigation en intérieur. Nous visons à évaluer et à identifier les catégories optimales de polygones qui facilitent la création d'un graphe de navigation pour un espace intérieur à partir des données OpenStreetMap (OSM). À cette fin, nous effectuons divers tests et les évaluons en fonction de leur temps d'exécution, ainsi que d'autres mesures détaillées dans la section suivante.

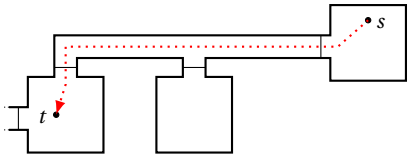

FIG. 1 – Exemple de cas d'utilisation pour la navigation en intérieur. L'utilisateur, représenté par s, souhaite se rendre au point t. Le résultat attendu est une ligne brisée, donnée en pointillé rouge.

3 Travaux réalisés

Lors de la subdivision du polygone en polygone convexes, l'algorithme [2] recherche tous les angles intérieurs supérieurs à 180 degrés. Ces sommets sont ensuite traités selon un ordre arbitraire non précisé dans l'article. Nous proposons une déclinaison de cet algorithme dans laquelle nous faisons varier l'ordre de traitement des sommets afin d'en étudier les variations topologiques qui en résultent.

Nous avons testé trois ordres différents : l'ordre **trigonométrique**, l'ordre **anti-trigonométrique** ainsi qu'un ordre **aléatoire**. Nous avons considéré trois mesures d'efficacité : la vitesse de l'algorithme, le nombre de portails créés dans la division ainsi que le nombre de polygones convexes finaux. D'autres mesures, comme la surface moyenne des polygones, et la surface du plus grand et du plus petit polygone dans le découpage, sont également proposées.

Ces mesures ont potentiellement un impact significatif sur la qualité de la solution obtenue, consistant en un plus court chemin qui sera calculé dans le graphe en intérieur défini à partir du découpage du polygone initial. A terme, nous souhaitons ainsi, au travers de différentes simulations, déterminer quels types de subdivisions permettent d'établir des trajets obtenus grâce à des plus court chemins entre paires de sources-destinations, tout en maintenant des temps de calcul très faibles.

Des batteries de tests sont actuellement menées sur plusieurs types de polygones, afin de comprendre l'impact de la topologie de départ sur le découpage obtenu.

— **Polygones aléatoires** : ces polygones sont générés de manière aléatoire [1] et servent de base pour évaluer les performances de l'algorithme.

— **Polygones orthogonaux** : un polygone orthogonal est un polygone dont les angles intérieurs sont exclusivement de 90 ou 270 degrés. Intuitivement, un polygone orthogonal est un ensemble de rectangles unis entre eux. Ils seront générés de manière aléatoire [4].

— **Polygones isothétiques** : un polygone isothétique est une figure géométrique définie par une propriété particulière liée à la disposition de ses côtés. Plus précisément, les côtés d'un polygone isothétique appartiennent à deux familles de lignes, et chacune de ces familles passe par un point fixe. Si l'on considère ces points fixes comme étant à l'infini, les lignes de chaque famille sont alors rendues parallèles entre elles ; en conséquence, un polygone isothétique peut être vu comme une généralisation des polygones orthogonaux, où les angles intérieurs sont de 90 ou 270 degrés. La notion d'isothétisme ajoute une flexibilité supplémentaire en permettant aux côtés du polygone de suivre des trajectoires plus variées.

— **Plans réels** : nous disposons d'un ensemble d'instances réelles.

— **Plans semi-simulés** : Les salles uniques dans les plans réels sont isolées et utilisées pour générer de nouveaux plans via des permutations aléatoires de ces salles.

Les tests préliminaires confirment que l'algorithme présente un découpage cohérent. Les exemples réels, comparés aux graphes existants, retournent un chemin correct et naturel, et une adaptation de l'algorithme aux polygones orthogonaux accélère l'algorithme substantiellement. Les découpages obtenus ouvrent la voie à certaines améliorations, comme par exemple la compression de graphes ou l'ajout d'arcs de transition pour accélérer le calcul de plus court chemins.

Références

[1] Pratik Shankar Hada. Approaches for generating 2d shapes. *UNLV Theses, Dissertations, Professional Papers, and Capstones*, 2014.

[2] Ramon Oliva and Nuria Pelechano. Automatic generation of suboptimal navmeshes. In *MIG'11 : Proceedings of the 4th International Conference on Motion in Games*, pages 328–339, 2011.

[3] Greg Snook. Simplified 3d movement and pathfinding using navigation meshes. In *Game Programming Gems*, pages 288–304. Charles River Media, 2000.

[4] Ana Paula Tomás and António Leslie Bajuelos. Generating random orthogonal polygons. In *Current Topics in Artificial Intelligence*, pages 364–373, Berlin, Heidelberg, 2004.