Dupli Checker

PLAGIARISM SCAN REPORT

0% Plagiarised	100% Unique	Date	2023-01-10
		Words	803
		Characters	5796

Content Checked For Plagiarism

Outputs at the end of step 2:

Figure 9: Outputs from Step II

Quad Data-structure:
* For each pair of nearest neighbours, a quadrilateral region is defined by joining the chords formed by internal tangents between these polygons.
* This quad data structure is special in the fact that; one pair of opposite sides represents obstacles and the other pair of opposite sides represents free space.
* Hence a robotic agent can enter this quad space only through a specific pair of opposite sides. Similarly, it can go from one quad to another only if the sides corresponding to its free spaces intersects with one another.
* And the shape of the path inside this quad is only defined by the geometry of the nearest neighbours.

4.4 Step III: Graph building using quad data structure:
* Based on the intersection criteria of quads, a bi-directional graph is created, taking these quads as the nodes.
* Also, these quads are ranked based on their geometries. And stored in a sorted order according to ranks. Bigger the area of the quad, or the lengths of free sides; the better will be the rank of that quad. And in any neighbor list/traversal order, the quad will appear the earlier.

*** All the steps till here are a one-time computation. And it eases the path search for upcoming steps.

4.5 Step IV: Path search
* The quad-graph is now searched for suitable paths.
* Preference is given to nodes with a higher rank. They have been already sorted according to areas and length of free-side. So first few nodes itself is supposed to cover the maximum area, thus attacking the problem greedily.
* Once we get paths in terms of these quad data structures.
* Local path planning methods are used to plan out subsequent paths inside these quads. And based on criteria such as the number of turns, straightness, width, and length of the path; the best path is sent as output.

A peek into the final optimized algorithm:
Category
Algorithm
Complexity
Pre-processing
Data reading
Complexity : O(m*n)
For m*n pixel image

Storing obstacle data as strips
O(m*n)

Merging strips into polygons
Complexity : O(m*k)
The average number of polygons in one row: k
Nearest-neighbours
Iterating for shielding test
O(p*p*l)
p: no. of obstacles
l: average number of neighbours of an obstacle

Tangent Computation
O(v^4); v : number of vertices in contributing polygon(obstacle)

Shielding test: fully shielded case
O(constant)

Shielding test: partial shielding case
O(l); l: average number of neighbours of an obstacle

Map Building
Quad intersection check
O(constant)

Tree Building

Path Search
Tree Traversal

Local Path Planning

Figure 10: Algorithm Summary

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Results

* A bare minimum and functional algorithm is ready which considers obstacles as polygons (no approximations) and not ellipses.
* The current version of the coded algorithm is not optimized to its full potential, and thus it doesn't guarantee significant accuracy.
* But it acts as a proof of concept and validates the proper functioning of nearest-neighbours and quad data-structure part of the algorithm.
* The quad-ranking strategy needs to be thought of to eliminate unnecessary back-and-forth turns in the planned paths.
* There are certain quads, which get covered up by 2-3 other quads as a group. This redundancy needs to be resolved to further speed up traversals.

5.2 Venues of Improvement and Further Research

* Approximation of polygons as ellipses without losing geometry information, and keeping the complexity within limits.
* Improving the method of ranking quads to achieve better, shorter paths in even lesser time.
* Extending the algorithm to 3D environments.

5.3 Proposed Timeline

* Jan 2023 – Complete till Elliptical approximation part
* Feb 2023 – Fine tune 2D algorithm and optimize it to full potential.
* March 2023 – Extending it to 3D environments
* April 2023 – Further fine-tuning and converting to ready-to-use format/package

Outputs from some failed and successful attempts :

Figure 11: Results from the current version of the algorithm

References

1. Sanjana Agrawal & R. Inkulu, (2022), Visibility polygons and visibility graphs among dynamic polygonal obstacles in the plane, Journal of Combinatorial Optimization volume 44, pages 3056–3082

2. Mandalika, Aditya Vamsikrishna, (2021), Efficient Robot Motion Planning in Cluttered Environments

3. Magid, Evgeni & Lavrenov, Roman & Afanasyev, Ilya. (2017), Voronoi-based trajectory optimization for UGV path planning, 383-387.10.1109/ICMSC.2017.7959506

4. Lounis Adouane, Ahmed Benzerrouk, Philippe Martinet, (2016), Mobile Robot Navigation in Cluttered Environment using Reactive Elliptic Trajectories, LASMEA, UBP-UMR CNRS 6602, France

5. Omar Souissi, Rabie Benatitallah, David Duvivier, AbedlHakim Artiba, Nicolas Belanger, Pierre Feyzeau, (2013), Path planning: A 2013 survey, IEEE Conference, Agdal, Morocco, 28-30 October 2013

6. Armin Hornung, Mike Phillips, E. Gil Jones, Maren Bennewitz, Maxim Likhachev, Sachin Chitta, (2012), Navigation in Three-Dimensional Cluttered Environments for Mobile Manipulation, IEEE International Conference on Robotics and Automation.

19

Matched Source

No plagiarism found