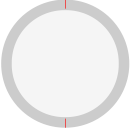
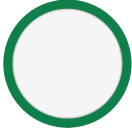


PLAGIARISM SCAN REPORT

	0% Plagiarised		100% Unique	Date	2023-01-10
				Words	996
				Characters	6888

Content Checked For Plagiarism

This algorithm is expected to perform better than the grid search algorithms, as it condenses a large amount of pixel/ grid information into a lesser number of nodes; keeping the geometry data intact. Thus lesser computational time in the graph traversal/search step.

CHAPTER 2

PROBLEM STATEMENT

The project work aims at creating a new algorithm in Path Planning using Computational Geometry approaches. And building a complete end-to-end package, that can read obstacle/map data from input visual feeds/ images and output planned path along with all metrics and details that might be required by the robotic agent.

This involves several smaller algorithms to load, and handle obstacle data. And representations for easy traversals and core geometric concepts to ease out complexities from existing steps in conventional algorithms. Some important algorithmic aspects include:

- * Approximation of obstacles as ellipses within accuracy limits.
- * Geometry-based checks to filter out "nearest-neighbors" obstacle pairs.
- * Creating a new quad data structure to efficiently handle and use nearest-neighbor information.
- * Strategies to assign weights to these quads based on their geometry and geographical position.
- * Extending the algorithm to make it compatible with 3D environments.

CHAPTER 3

LITERATURE REVIEW AND RELATED WORKS

Path planning algorithms are used by mobile robots, unmanned aerial vehicles, and autonomous cars in order to identify safe, efficient, collision-free, and least-cost travel paths from an origin to a destination. Based on the use case, several approaches can be used to plan paths. These algorithms can be categorized in many different ways.

Figure 1: Various classifications of Path Planning Algorithm

Based on the nature of the algorithm, broadly these can be classified as:

- * Grid Based Search
- * Geometry Based Algorithms
- * Sampling-Based Algorithms
- * Artificial Potential Algorithm

3.1 Visibility Graph

A visibility graph is a graph of inter-visible locations, typically for a set of points and obstacles in the Euclidean plane. Paths are searched in terms of line segments instead of points or grid cells.

Figure 2: Visibility graph

3.2 Voronoi Graph

The Voronoi diagram is a partition of a plane into regions close to each of a given set of objects and the smallest area entity resulting from the intersection of edges is called a Voronoi cell. The path search methods to plan the path in terms of edges and cells from this Voronoi diagram are termed as Voronoi-based graph search algorithms.

Figure 3: Voronoi Graph

In our algorithm, some features have been wisely picked from these algorithms to formulate our version. In visibility graphs (or VGRAPH), free space is represented as line segments, and paths are planned in terms of these lines. Similarly, a quad structure is used to represent free space, and graphs are constructed similarly in our algorithm. In the Voronoi graph method, a Voronoi diagram is generated first to get information regarding neighboring obstacles of a given obstacle. Here we are using some geometrical checks to rule out far-off obstacles and identify neighboring obstacles.

The idea of representing obstacles as ellipses has already been thought off and worked upon. Lounis Adouane, Ahmed Benzerrouk and Philippe Martinet(2011) in their paper on Mobile Robot Navigation in Cluttered Environment using Reactive Elliptic Trajectories have approximated walls and other obstacles as ellipses.

Figure 4: Interpolated wall using a circle and an ellipse shape

Taking inspiration from this work, we are going to approximate obstacles similarly. But rather than approximating each obstacle as 1 ellipse, we are using a set of differently sized intersecting ellipses to capture the geometry more efficiently.

CHAPTER 4

ALGORITHM

4.1 Inputs:

- * It takes binary images of an obstacle course as input where obstacles will be represented by black color and white color represents the free area.
- * For real-time applications, video frames will be passed to image processing libraries to get masked images which can then be read by this algorithm.

4.2 Step I: reading the data and storing it as polygons

- * Read the image row by row
- * Identify the strip obstacles in each row; store these strip data in vectors
- * Traverse through these vectors top down, and keep merging the strips

Figure 5: Locating obstacles from the binary image

4.3 Step II: "Nearest-neighbour computation":

- * Iterate through the list of obstacles:
- * For each obstacle, iterate through the remaining $n-1$ obstacles in order of the least distance from the selected obstacle
- * Check if this obstacle gets shielded by the neighbors of chosen obstacle.
- * If this obstacle contributes to the Voronoi diagram of chosen obstacles; add it to the "neighbors" list of chosen obstacle.

Figure 6: Obstacles falling in the black region don't contribute to Voronoi edge with the first obstacle

Shielding Check :

Let's say, in the outer loop; obstacle 1 is being iterated through.

(remaining $n-1$ obstacles are to be iterated in order of least distance from obstacle 1).

Let's assume that obstacle 2 and obstacle 5 have been already iterated through resulting in the addition of these obstacles to the neighbours list of obstacle 1.

- * For the next coming iterations; the remaining obstacles need to be checked if they get shielded by neighbours of obstacle 1, ie, obstacle id 2 and 5.

- * For shielding check; find the internal tangents between obstacle 1 and obstacle 2

- * If the rectangular bounds of the k th obstacle fall fully inside the shielding region formed by internal tangents; the k th obstacle will no more share the Voronoi edge with obstacle 1 (next page).

- * In the case of partial shielding; look for the diagonals of rectangular bounds. Delete the part of these diagonals which is

shielded by obstacle 2. And continue with the algorithm.

* While iterating through neighbours of chosen obstacle (obstacle 1), if these diagonals get completely deleted at the end; The kth obstacle will no more contribute to the Voronoi edge. Otherwise in case of partial or complete exposure; add the kth obstacle to the neighbours list of chosen obstacle (obstacle 1).

Figure 7: Shielding Check (Complete Shielding)

Figure 8: Shielding Check (Partial Shielding)

Matched Source

No plagiarism found

Check By:  Dupli Checker