论文题目

A New Algorithm for Euclidean Shortest Paths in the Plane

作者：HAITAO WANG

刊物：Journal of the ACM

出处链接：https://dl.acm.org/doi/10.1145/3580475

摘要

本文是对 Wang Haitao 于 2023 年发表在 ACM 的论文《A New Algorithm for Euclidean Shortest Paths in the Plane》的概述与分析。论文首先介绍了计算几何学领域，并对该领域中的常见问题及其解决方法进行了综述。接着，本文概述了 Wang Haitao 提出的算法，并对其进行了分析。该算法旨在解决在二维平面上，给定一组两两不相交的多边形障碍物，找到连接任意两点 s 和 t 的欧几里得最短路径问题。这个问题在计算几何和机器人路径规划领域中具有广泛的应用。

算法的核心思想是构建一个最短路径图（Shortest Path Map, SPM），这个图能够为任意查询点提供最短路径长度的快速计算以及路径的快速生成。作者首先利用三角剖分来简化问题，然后构建了一个 SPM，这是一个数据结构，它将自由空间划分为多个区域，这些区域中的每个点都有相同的最短路径前驱。算法采用了连续 Dijkstra 方法来扩展波阵面，这是一种从源点开始，逐步向自由空间中扩展的过程。

对于非凸障碍物，作者引入了走廊结构来进一步细化最短路径图。走廊结构将自由空间划分为海洋、海湾和运河。这种结构允许算法在更细的粒度上处理障碍物的边界，从而提高最短路径的准确性。为了提高算法的效率，作者还进行了空间和时间上的优化，通过使用更紧凑的数据结构和优化算法流程，算法能够在保持正确性的同时减少所需的时间和空间资源。

本文还分析了平面欧式距离已有算法的不足之处，并尝试提出新算法解决该不足。提出的修改方案包括自适应三角剖分、简化数据结构和动态障碍物处理机制，旨在提高最短路径算法在动态环境、大规模数据处理和内存使用效率方面的性能。通过这些步骤，算法能够有效地构建 SPM，并利用它来快速响应最短路径查询。

关键词：计算几何学，平面欧氏距离最短路径，最短路径图，三角剖分

一、 引言

本文是对 Wang Haitao 于 2023 年发表在 ACM 的论文《A New Algorithm for Euclidean Shortest

Paths in the Plane》的概述与分析。本文将首先对平面欧式最短路径问题所属的领域计算几何学进行简单介绍，介绍该领域常见问题及其解决方法，并对各类解决方法进行比较。然后概述《A New Algorithm for Euclidean Shortest Paths in the Plane》的主要内容，并对其提出的算法进行分析。最后会分析平面欧式距离已有算法的不足之处，并尝试提出新算法解决该不足。

二、 领域综述

1 计算几何学

为了真正理解作者提出的算法，首先应该对该领域有一定的了解。因此，我们先对该领域进行简单的分析与介绍，下一部分，再对论文的具体内容与算法进行分析。

平面欧式最短路径问题是计算几何学中的一个问题。计算几何是计算数学、逼近论、微分几何、代数几何以及计算机科学相互交叉的几何学分支[1]前言。它研究的是几何图形的性质和空间关系，并通过数学与计算机方法进行计算和推导。计算几何学通常涉及到点、线、面、多边形等几何图形的计算问题，如距离计算、面积计算、几何变换等。它在计算机图形学、地理信息系统、机器人学等领域有着广泛的应用。简单来说就是运用计算机方法解决各种几何问题。

当然，计算几何与数学上的几何学并不等价。数学上的几何概念是完美的，连续的，然而在计算机中，所有的几何模型都只能用离散的数据进行表示。所以，使用计算机无法对数学中的几何概念进行精确的表达。同时，数学家们在研究几何问题时，往往更关注能否解决，但算法不仅关注能否解决，还关注能高效的解决，而数学上不能解决的问题，有时候也可以通过启发与近似的方法进行解决。因此计算几何更接近计算而非几何，实际上更像算法的一个分支。

2 问题分类

计算几何常见问题有几何查找，多边形的获取，几何体的划分，路径与回路等问题。

几何查找或者称几何检索是指在属性相同的一批几何对象（如点、直线段、圆、多边形、多面体等）中定位某个指定的几何对象，或者在某个特定的域中寻找该域所包含的具有某

种属性的所有几何对象[1]18。简单来说，几何查找是指在一系列几何对象的数据中，查找指定的几何对象的相关数据。由于往往几何查找不只进行一次，所以对几何数据的预处理，也就是将几何数据按一定结构与规则存放也是几何查找的一部分。一般来说，几何查找的耗费包括询问时间（回答一次询问所需的时间）、存储（数据结构占用的内存）、预处理时间（组织数据或某种结构的时间）、修改时间（将指定几何对象所对应的数据插入数据结构或将其从数据结构中删去所需要的时间）[1]18。

多边形获取问题是从给定的点集或边集中，寻找符合要求的多边形，该多边形连接线段集中所有的线段或者包含、满意地包含点集、线段集。通常情况下，这些点或线段是在二维平面上给定的，我们需要找到这些点所围成的最小的多边形，即外接矩形最小的多边形。这个问题在计算机图形学、地理信息系统等领域中经常出现，例如在绘制地图、图像处理、边界检

测等方面都需要解决多边形获取问题。

在计算几何中，几何体的划分问题是指将一个几何体分割成更小的几何体或几何形状的过程。这种划分可以通过不同的方法和技巧来实现，例如平面切割、投影、旋转等。划分问题在数学、工程、建筑等领域中都有重要的应用。在工程和建筑领域，划分问题常常涉及到设计和制造复杂的几何体结构。工程师和设计师需要将一个大型几何体划分成许多小部件，以便更好地进行建模、分析和制造。这种划分可以帮助他们更好地理解和处理复杂的几何体结构，提高工作效率和准确性。

路径与回路问题就是本文研究的“平面欧式最短路径问题”所属的问题。路径与回路问题实际上研究的是由一点到另一点的运动规划，一般而言分为三类：（1）判定问题：运动物体能否由起点运动到终点，即起点与终点间是否存在一条自由路径（2）构造路径：寻找运动物体从起点移动到终点的一条自由路径。（3）最短路径：寻找运动物体从起点移动到终点的一条最短自由路径。路径与回路问题常应用在机器人学，路径规划等领域，有较大的实用价值。

3 解决方法

本部分将介绍计算几何问题种不同问题的常见解决方法，并进行比较。由于路径与回路问题与欧几里得最短路径问题关联性较强，本文将重点介绍该部分。

3.1 几何查找问题

在解决几何查找问题时，常见的解决方法可以分为暴力搜索、空间索引、几何哈希等几类方法。

暴力搜索是最简单直接的方法，即对每个可能的解进行遍历搜索，然后根据特定的几何条件进行筛选。虽然简单易懂，一定能保证获得最优解，但在处理大规模数据时效率极低。

空间索引是一种常用的方法，通过构建空间索引结构来加速几何查找操作。这样可以减少不必要的计算量，提高查询效率。常见的空间索引方法包括：R 树，一种多维索引结构，适用于范围查询和最近邻查询；Quadtree，一种二维索引结构，将空间递归划分为四个象限，适用于范围查询和点查询；KD 树，一种多维索引结构，通过不断交替选择坐标轴进行划分，适用于范围查询和最近邻查询；Grid 索引：将空间划分为网格单元，每个对象被映射到一个或多个网格单元，适用于范围查询和点查询。这些空间索引方法可以根据具体的应用场景和查询需求选择合适的方法来提高查询效率。在实际应用中，选择合适的空间索引方法可以显著提高查询性能和减少计算开销。

几何哈希方法是一种将几何对象映射到哈希表中的技术，以便快速查找相交或相邻的几何对象。其基本思想是将几何空间划分为多个小区域，并将每个几何对象映射到相应的区域中。当进行查询时，只需在查询对象所在的区域及其相邻区域中查找几何对象，从而减少搜索的范围，提高查询效率。

综合比较这几种解决方法，暴力搜索简单易懂但效率低下；空间索引适用于大规模数据的快速查找；几何哈希适用于快速匹配几何对象，适用于中规模数据。在实际应用中，可以根据数据规模和具体问题的特点选择合适的解决方法。

3.2 多边形获取问题

多边形获取问题是在计算机视觉和图像处理领域中常见的一个问题，涉及到从图像中识别和提取多边形的形状信息。针对这个问题，常见的解决方法可以分为基于几何形状的方法、基于边缘检测的方法和基于深度学习的方法。

基于几何形状的方法是一种传统的解决方案，它通常通过检测多边形的顶点和边的关系来识别多边形。这种方法的优点是计算简单，易于理解和实现。常见的算法包括霍夫变换、多边形逼近等。然而，这种方法对于复杂的多边形形状和噪声敏感，准确率有限。

基于边缘检测的方法则是通过首先检测图像中的边缘信息，然后根据边缘信息来提取多边形。这种方法的优点是对噪声具有一定的鲁棒性，能够处理复杂的多边形形状。常见的边缘检测算法包括 Canny 边缘检测、Sobel 算子等。然而，边缘检测的结果可能存在断裂或者不完整的情况，影响多边形的准确提取。

基于深度学习的方法是近年来发展起来的一种新型解决方案，它通过训练神经网络来学习多边形的特征，从而实现多边形的识别和提取。这种方法的优点是能够处理复杂的多边形形状和噪声，具有较高的准确率。然而，深度学习方法需要大量的标注数据和计算资源，训练过程较为复杂。

综合比较这几种解决方法，基于深度学习的方法在处理复杂多边形和噪声方面具有优势，但需要大量的数据和计算资源；基于边缘检测的方法对于一般情况下的多边形提取效果较好，但可能存在断裂或不完整的情况；基于几何形状的方法简单易懂，但对于复杂多边形的提取准确率有限。因此，在实际应用中，可以根据具体情况选择合适的方法来解决多边形获取问题。

3.3 几何体的划分问题

在几何体的划分问题中，常见的解决方法可以分为几种分类，包括平面切割、体积计算、投影法和三维建模等。每种方法都有其独特的特点和适用范围。

首先是平面切割方法，这种方法通过在几何体上进行平面切割，将几何体划分为多个简单的几何形状，然后计算每个部分的面积或体积，最终得到整个几何体的划分情况。这种方法适用于各种几何体，如立方体、圆柱体等，但对于复杂的几何体可能需要较多的计算步骤。

其次是体积计算方法，这种方法通过计算几何体的体积来确定其划分情况。可以通过公式或数值积分等方式来计算几何体的体积，然后根据需要进行划分。这种方法适用于需要准确计算几何体体积的情况，但对于复杂的几何体可能需要较高的数学知识

接着是投影法，这种方法通过在几何体上投影出不同的视图，然后根据投影的形状和位置来确定几何体的划分情况。这种方法适用于需要直观展示几何体划分情况的情况，但对于非规则几何体可能需要较多的几何知识

最后是三维建模方法，这种方法通过使用计算机软件对几何体进行三维建模，然后根据需要进行划分。这种方法适用于需要进行复杂几何体的划分和可视化展示的情况，但对于初学者可能需要一定的学习成本。

综合比较这几种方法，平面切割方法适用于各种几何体，体积计算方法适用于需要准确计算体积的情况，投影法适用于直观展示划分情况，三维建模方法适用于复杂几何体的划分和可视化展示。选择合适的方法取决于具体的需求和情况。

3.4 路径与回路问题

平面上的路径与回路问题中，常假定障碍物是多边形的，虽然这种假设可能与现实情况有一定偏差，但可以极大地降低计算的难度与复杂性，同时产生的结果偏差较小。在这种假设中，多边形障碍与起止点的关系可以比较容易地通过图来表示，所以，该部分问题常常转换为图论进行求解。

3.4.1 判定问题

解决路径与回路问题的判定问题时，常见的方法可以分为几类，包括搜索法、凸包法和拓扑排序法等。

搜索法常见的有深度优先搜索，广度优先搜索等。深度优先搜索法通过深度优先遍历图的所有节点，判断是否存在环路。具体步骤包括从起始点开始深度优先搜索，遍历所有相邻节点，并标记已访问节点，若出现已访问节点则存在回路。深度优先搜索法适用于判定简单的路径和回路问题。广度优先搜索法也可用于路径与回路的判定问题。该算法通过逐层遍历图的节点，判断是否存在环路。具体步骤包括从起始点开始广度优先搜索，逐层遍历相邻节点，并标记已访问节点，若出现已访问节点则存在回路。广度优先搜索法适用于判定简单的路径和回路问题。

凸包法是一种基于凸包概念的方法，通过计算给定点集的凸包来判断是否构成一条路径或回路。具体步骤包括计算凸包，确定凸包上的点是否包含所有给定点，若包含则构成路径或回路。凸包法适用于简单的路径与回路判定问题。

拓扑排序法是一种有向无环图(DAG)排序算法，也可用于路径与回路的判定问题。该算法通过拓扑排序判断是否存在环路。具体步骤包括构建有向图，进行拓扑排序，若存在环路则判定为回路。拓扑排序法适用于判定有向图中路径与回路的问题。

对比各类方法，深度优先搜索法和广度优先搜索法适用于简单的路径与回路判定，深度优先搜索更适用于判定回路，而广度优先搜索更适用于判定路径。凸包法适用于判定简单的路径与回路，通过凸包的构建来判断是否构成路径或回路。拓扑排序法适用于有向图中路径与回路的判定，能够有效地判断是否存在环路。

3.4.2 构造路径

构造路径问题常见的方法可以分为几类，包括蛮力搜索法、几何构造法、最短路径算法和回溯法等。

蛮力搜索法是一种简单但计算复杂度较高的方法，适用于小规模问题。该方法通过穷举所有可能的路径来找到连接给定点的路径或回路。具体步骤包括列举所有可能的路径，计算每条路径的长度，并找到最短路径或回路。蛮力搜索法的缺点是计算量大，但在规模较小的问题上可以得到精确解。

几何构造法是一种基于几何关系的方法，通过构造几何图形来找到连接给定点的路径或回路。这种方法通常涉及利用直线、圆弧等几何元素构造路径，并通过几何性质来确定最优路

径或回路。几何构造法适用于简单的情况，能够快速找到路径或回路。

最短路径算法是一类用于找到两点之间最短路径的算法，如 Dijkstra 算法、Floyd-Warshall 算法和 A*算法等。这些算法通过不同的策略来寻找连接给定点的最短路径，考虑了路径长度和节点间的关系。最短路径算法适用于复杂的路径构造问题，能够高效地找到最优路径。

回溯法是一种基于递归的搜索方法，用于在有限的搜索空间中找到路径或回路。该方法通过深度优先搜索的方式逐步构造路径，当遇到无法继续前进的情况时，回溯到上一步进行尝试。回溯法适用于具有多个分支选择的问题，能够找到所有可能的路径或回路。

对比各类方法，蛮力搜索法适用于小规模问题，但计算复杂度高；几何构造法适用于简单情况，速度较快；最短路径算法能够高效地找到最优路径，适用于复杂问题；回溯法能够找到所有可能的路径或回路，适用于多分支选择的问题。

3.4.3 最短路径

解决最短路径问题时，常见的方法可以分为几类，包括蛮力搜索法、几何构造法、Dijkstra 算法和最小生成树算法等。

蛮力搜索法是一种简单但计算复杂度较高的方法，适用于小规模问题。该方法通过穷举所有可能的路径来找到连接给定点的最短路径。具体步骤包括列举所有可能的路径，计算每条路径的长度，并找到最短路径。蛮力搜索法的缺点是计算量大，但在规模较小的问题上可以得到精确解。

几何构造法是一种基于几何关系的方法，通过构造几何图形来找到连接给定点的最短路径。这种方法通常涉及利用直线、圆弧等几何元素构造路径，并通过几何性质来确定最优路径。几何构造法适用于简单的情况，能够快速找到最短路径。

Dijkstra 算法是一种用于解决单源最短路径问题的经典算法。该算法通过不断更新起始点到其他点的最短距离来找到最短路径。具体步骤包括初始化起始点到其他点的距离，然后逐步更新距离直到找到最短路径。Dijkstra 算法适用于有向图和无向图，能够有效地找到最短路径。许多最短路径算法都是将数据构造为图利用该方法求解的。

最小生成树算法是一种用于解决无向图的最小生成树问题的算法，如 Prim 算法和 Kruskal 算法。这些算法通过构建一个不含环路的树结构来找到最短路径。具体步骤包括选择起始点，逐步添加边直到构建完整的树。最小生成树算法适用于无向图，能够找到最小生成树的最短路径。

对比各类方法，蛮力搜索法适用于小规模问题，但计算复杂度高；几何构造法适用于简单情况，速度较快；Dijkstra 算法适用于单源最短路径问题，能够有效地找到最短路径；最小生成树算法适用于无向图的最小生成树问题，能够找到最小生成树的最短路径。

3.4.4 平面欧几里得最短路径问题

虽然平面欧几里得最短路径问题是路径与回路问题中最短路径问题的一种，可以合并到上一部分，但该问题对比其他的最短路径问题，亦有一定的特殊性，所以这里将平面欧几里得距离最短路径问题的常见算法进行单列，作为对下面部分的引子。

一种简单的算法是利用图论求解。由起点到终点的最短路径是一条折线链，该链的顶点是多

边形障碍物的顶点。利用 Dijkstra 的最短路算法和可视图算法可以求解 ESPO 问题，其时间复杂性为 0(n^2)。如果可视图是稀疏的，则在 O(m+nlogn)时间内可以确定解，其中 m 是可视图边的数目。

另一种解法是最短路径映射法（SPM）。该算法的基本思想是将原始问题转化为一个新的问题，通过在新问题上求解最短路径来得到原始问题的最短路径。

还有一种直观形象的算法称作波浪线算法，又称 continuous Dijkstra 算法，该算法一度是理论最优的算法，由 J.S.B.Mitchell[2]首先提出，后由 Hershberger 和 Suri 改进[3]，复杂度降至理想的 O(NlogN)[4]。虽然其实现较为复杂，但思想清晰直观很好理解。将平面想象成一片平静的湖面，当在起点投入一颗石子时，会产生一系列波浪向外扩散。障碍多边形的边就像湖岸，波浪会在碰到湖岸时消失；同时，当两条波浪相遇时也会消失。每条波浪都有一个圆心，最初的圆心位于起点，之后可能在各个顶点处产生新的圆心，形成新的波浪。最短路径的求解方法很简单：从终点开始依次回退至各个圆心，直到回到起点。这些逆向路径即为最短路径。

在《A New Algorithm for Euclidean Shortest Paths in the Plane》Wang Haitao 结合了 SPM 方法与波浪线法提出了一种解决平面欧几里得最短路径问题的新算法。

三、 论文概述

1 解决的问题

本文解决的问题是在二维平面上，给定一组两两不相交的多边形障碍物，找到连接任意两点 s 和 t 的欧几里得最短路径。这个问题是计算几何和机器人路径规划领域中的一个经典问题，具有广泛的应用，例如在地图上规划最短行驶路线、在多障碍物环境中的机器人导航等。

具体来说，问题有以下几个要素：

多边形障碍物集合：在平面上定义了一个由多边形组成的障碍物集合ϕ，其中每个多边形障碍物由一系列顶点和边组成，障碍物之间互不相交。

自由空间：自由空间 F 是指除去所有障碍物内部之后的剩余区域，即平面上可以自由通行的部分。

源点和目标点：在自由空间 F 中给定两个点 s（源点）和 t（目标点），需要找到从 s 到 t 的最短路径。

欧几里得最短路径：所求的最短路径是按照欧几里得距离（即直线距离）来定义的，这是最常见的最短路径定义方式。

障碍物回避：最短路径必须避开所有障碍物的内部，只能在自由空间 F 内进行路径规划。

2 符号约定

为了方便叙述，在本文中，我们进行如下的符号约定，并保证大部分符号与原论文中约定相同。

符号	意义
ϕ	障碍物集
$\mathcal{F}$	平面
n	障碍物总顶点数
h	障碍物总数
s	路径起点
t	路径终点
$\pi(s,t)$	s 点到 t 点最短欧几里得路径
$d(s,t)$	s 点到 t 点最短欧几里得路径长度
∂A	A 区域的边界
$SPM(s)$	最短路径图
$\boldsymbol{W(f,e)}$	波阵面

3 解决问题的思想

作者解决欧几里得最短路径问题的核心思想是构建一个最短路径图（Shortest Path Map, SPM），这个图能够为任意查询点提供最短路径长度的快速计算以及路径的快速生成。

作者首先利用三角剖分（Triangulation）来简化问题。三角剖分是一种将平面划分为三角形区域的方法，它能够处理多边形障碍物之间的复杂关系。通过三角剖分，自由空间被划分为简单的三角形区域，这为后续的最短路径计算提供了一个良好的基础。

在三角剖分的基础上，作者构建了一个最短路径图（SPM）。SPM 是一个数据结构，它将自由空间划分为多个区域，这些区域中的每个点都有相同的最短路径前驱（即最短路径上的最后一个障碍物顶点）。这样，对于任意查询点，都可以通过查找其所在的区域来快速确定其最短路径。

算法采用了连续 Dijkstra 方法来扩展波阵面（Wavefront），这是一种从源点开始，逐步向自由空间中扩展的过程。波阵面是所有与源点有相同或更少距离的点的集合。通过不断更新波阵面，算法能够逐步发现自由空间中的新区域，并计算出这些区域的最短路径。

对于非凸障碍物，作者引入了走廊结构来进一步细化最短路径图。走廊结构将自由空间划分为海洋（Ocean）、海湾（Bays）和运河（Canals）。这种结构允许算法在更细的粒度上处理障碍物的边界，从而提高最短路径的准确性。

为了提高算法的效率，作者还进行了空间和时间上的优化。通过使用更紧凑的数据结构和优化算法流程，算法能够在保持正确性的同时减少所需的时间和空间资源。

SPM 的构建允许动态更新和查询。对于任意查询点，算法可以快速地在 SPM 中定位该点，并找到从源点到该点的最短路径。这种动态性使得算法不仅适用于单次查询，还适用于多次查询场景，如路径规划中的连续查询。

通过上述思想，作者提出的算法能够在相对更短的时间内，使用更少的空间资源，为给定的查询点计算出避开障碍物的最短路径。这种方法在理论上和实践上都具有重要意义，尤其是在处理复杂障碍物环境时。

4 基本算法的描述

4.1 预备阶段：三角剖分

三角剖分是计算几何中一个重要的概念，它涉及将平面区域划分为三角形的过程。在本文中，三角剖分是算法的预备步骤，为后续的最短路径计算打下基础。三角剖分的主要目的是将包含多边形障碍物的自由空间划分成简单的三角形区域，这样可以简化路径查找过程，因为三角形是一种简单且易于处理的几何形状。

首先识别所有障碍物的顶点和边，这些顶点和边是三角剖分的基础。然后在障碍物内部，识别可以形成三角形的顶点组合。这些三角形完全位于障碍物内部，不会与自由空间相连。其次，对自由空间的三角剖分，使用算法（如 Delaunay 三角剖分或凸包技术）来连接自由空间中的顶点，形成三角形。确保生成的三角形不会与任何障碍物相交，每个三角形的内角都满足一定条件（例如，对于 Delaunay 剖分，任意三角形的外接圆内不包含其他顶点）。在障碍物的边界上，顶点和边将与自由空间的顶点一起形成边界三角形。这些三角形一部分位于障碍物内部，一部分位于自由空间。

在本文中，三角剖分是算法的第一步，为后续的最短路径计算提供了一个良好的基础。通过三角剖分，复杂的自由空间被划分成简单的三角形区域，使得可以在这些区域上应用更高效的路径查找算法。

4.2 构建符合子划分（Conforming Subdivision）

符合子划分的目的是创建一个平面划分，这个划分能够适应障碍物的边界，并且能够用于后续的最短路径计算。这个划分将平面划分成多个单元，每个单元对应一个或多个障碍物的顶点，并且能够指导波阵面的扩展。

首先，确定每个障碍物的“顶点”，这些顶点将作为构建符合子划分的基础。然后使用 Hershberger 和 Suri 算法（或其他类似的算法）基于顶点集合 V 初始化一个符合子划分 S。这个划分不考虑障碍物的边缘，只考虑顶点。S 是一个四叉树风格的划分，将平面划分为多个单元（正方形或正方形环），每个单元包含一个或多个顶点。将障碍物的边缘插入到 S 中，形成一个新的符合子划分 S'。这个步骤是将障碍物的实际形状考虑进来，确保划分能够正确地反映障碍物的布局。每个单元的边界由透明边（划分构造时引入的边）和障碍物边缘组成。需要确保每个单元的边界是连续的，并且满足一定的几何属性。每个单元需要维护一些属性，例如边界上的透明边数量、单元内的顶点数量等，这些属性对于后续的波阵面扩展是必要的。对于每个透明边 e，计算一个覆盖区域 U(e)，这个区域包含了所有从 e 出发不会与障碍物相交的最短路径。

构建符合子划分的算法复杂度通常与障碍物顶点的总数有关，一个好的算法能够在 O(n log n)时间内完成，其中 n 是所有障碍物顶点的总数。

在构建符合子划分时，需要确保每个单元的边界是连续的，并且不包含任何障碍物的内部。

划分应该能够适应障碍物的实际形状，不能简单地按照网格划分。划分的单元大小应该适中，以便于后续的波阵面扩展和最短路径计算。

构建符合子划分是算法的基础步骤之一，它为后续的波阵面扩展提供了必要的结构化环境。通过这个划分，算法能够有效地管理和扩展波阵面，从而找到避开障碍物的最短路径。

4.3 初始化阶段

初始化部分是算法开始执行前的准备阶段，它为算法的后续步骤奠定基础。在本文提出的算法中，初始化步骤主要包括以下几个方面：

1 定义源点和目标点

确定算法的源点s和目标点t。

2. 构建初始波阵面

以源点s为中心，创建一个初始的波阵面。波阵面是一组点的集合，这些点与源点等距离或更近距离，并且这个集合随着算法的进行而动态扩展。

3. 计算透明边的初始状态

对于符合子划分S中的每个透明边（即不包含障碍物的边），计算其与源点s的距离，并初始化波阵面的状态。

4. 设置数据结构

初始化所需的数据结构，如平衡二叉搜索树，用于维护波阵面和处理波阵面的合并与传播。

5. 计算覆盖时间

对于每个透明边e，计算一个覆盖时间 $\mathrm{covertime}(e)$，这个时间是波阵面到达边e的预计时间，用于后续排序和处理。

6. 生成初始波阵面

在源点s周围的透明边或单元上生成初始波阵面。这通常涉及到计算从s到这些边或单元的最短路径，并确定波阵面扩展的顺序。

7. 标记生成器

如果源点s位于某个单元内，或者某个透明边的端点在单元内，将相关的生成器（即障碍物顶点）标记为初始生成器。

8. 计算可达区间

对于每个初始生成器，计算它能够到达的区间（即波阵面上的一段）。这些区间将构成初始波阵面的一部分。

9. 确定算法启动条件

确保所有必要的信息和数据结构都已准备就绪，算法可以开始执行波阵面扩展步骤。

初始化部分是算法成功执行的前提，它确保了算法能够从已知的源点出发，系统地探索自由空间，并为后续的最短路径计算做好准备。通过初始化步骤，算法为处理动态查询或静态路径规划问题奠定了坚实的基础。

4.4 波阵面扩展算法

波阵面扩展算法是本文提出算法的核心部分，它用于逐步探索自由空间，并构建最短路径图（SPM）。

首先，初始化波阵面，从源点s开始，创建一个初始的波阵面，通常是一个以s为中心的最小覆盖圆。使用平衡二叉搜索树维护波阵面的状态。然后定义波阵面和生成器。波阵面表示已知最短路径的点集合，随着算法的进行，波阵面将逐渐扩展，直到覆盖整个自由空间。生成器是波阵面扩展的驱动力，每个生成器与一个障碍物的顶点相关联，并有一个方向。生成器用于在波阵面上生成新的点。

随后，处理透明边，对于符合子划分S'中的每个透明边（不包含障碍物的边），根据其与源点s的相对位置，初始化波阵面的状态。计算透明边端点到源点s的最短路径，并将这些端点作为波阵面的一部分。从初始波阵面开始，逐步扩展到自由空间中的每个点。对于每个单元，根据波阵面的状态和透明边的信息，更新单元内的波阵面。当波阵面通过透明边扩展到新单元时，可能需要与从其他方向到达的波阵面合并。合并过程需要解决不同波阵面之间的冲突，并确定最终的波阵面状态。

波阵面到达障碍物边界时，需要正确处理与障碍物的交互。在障碍物边界上可能需要创建新的生成器，或者更新现有生成器的状态。随着波阵面的扩展，算法记录从源点s到每个点的最短路径长度。对于每个点，算法可以确定其在 SPM 中的前驱点，即最短路径上的最后一个障碍物顶点。

当波阵面扩展到覆盖整个自由空间，或者达到目标点t时，算法终止。利用波阵面扩展过程中的信息，构建完整的最短路径图（SPM）。SPM 将自由空间划分为多个区域，每个区域内的点都有相同的最短路径前驱。

波阵面扩展算法的关键在于有效地管理和更新波阵面的状态，同时处理波阵面与障碍物的交互。通过这个算法，可以逐步探索自由空间，并为每个点确定最短路径。这个过程不仅需要精确的几何计算，还需要高效的数据结构来支持快速的查询和更新操作。

4.5 波阵面合并过程

波阵面合并操作是波阵面扩展算法中的步骤重要，它涉及将来自不同方向的波阵面合并成一个单一的波阵面。这个操作需要精确处理，以确保算法能够正确地更新波阵面的状态，并为后续的最短路径计算提供正确的信息。

波阵面合并的目的是解决当多个波阵面到达同一个透明边或单元时，如何确定最终波阵面的形态。合并操作需要决定哪些区域被哪个波阵面所覆盖，并解决可能出现的冲突。

首先,确定所有对e有贡献的波阵面$W(f,e)$，这些波阵面是从透明边e的两侧到达e的。然后，初始化一个空列表$W(e)$，用于存储合并后的波阵面。根据贡献波阵面到达e的时间对它们进行排序。通常，较早到达的波阵面具有更高的优先级。对于e的每个端点，确定哪个贡献波

阵面最先到达该端点。这涉及到计算端点到各个贡献波阵面的距离。从优先级最高的贡献波阵面开始，将其添加到$W(e)$中。对于每个后续的贡献波阵面，检查其是否与$W(e)$中已有的波阵面冲突。如果不冲突，直接添加到$W(e)$。如果冲突，需要解决冲突，这可能涉及到比较波阵面的距离或选择一个最优的波阵面。

对于e上的每个点，确定哪个贡献波阵面覆盖了该点。这通常涉及到计算点到各个贡献波阵面的距离，并选择最近的波阵面。更新$W(e)$中每个波阵面的生成器，确保它们正确地反映了合并后的波阵面状态。

波阵面合并操作的复杂度通常与贡献波阵面的数量和它们包含的生成器数量有关。如果每个贡献波阵面有 $O(h)$ 个生成器，其中 h 是障碍物的数量，那么合并操作的复杂度可以是O(hlogh)。

波阵面合并是波阵面扩展算法中的一个关键步骤，它确保了算法能够在面对多个贡献波阵面时，正确地更新波阵面的状态，并为计算最短路径提供准确的信息。

4.6 处理障碍物边界

处理障碍物边界的步骤在波阵面扩展算法中是为了确保波阵面在遇到障碍物时能够正确地调整其扩展方向和状态。

首先识别接触点，确定波阵面与障碍物边界接触的点，这些点是波阵面扩展过程中需要特别注意的位置。然后在波阵面与障碍物相交的点上，根据需要创建新的生成器。生成器是波阵面上的一个点，代表从源点出发到达该点的最短路径。对于新生成的生成器，更新波阵面的状态，包括其位置、方向和权重（即到源点的距离）。计算可对于每个新生成器，计算其可达区域，即从该生成器出发不会与障碍物相交的最短路径覆盖区域。维护与每个生成器相关的双曲线边界，这是波阵面的一部分，表示与生成器等距离的点的集合。

当波阵面的双曲线边界相交或与障碍物边界相交时，处理这些事件，可能涉及删除或更新生成器。在障碍物边界处更新波阵面后，继续扩展波阵面，直到覆盖所有自由空间或到达目标点。

处理障碍物边界的复杂度取决于障碍物边界上与波阵面相交点的数量，以及需要更新的生成器数量。在最坏情况下，这个步骤的复杂度可以是 O(hlogh)，其中 h 是障碍物数量。

这个步骤的关键在于确保波阵面在障碍物边界上的正确行为，以便能够正确计算避开障碍物的最短路径。

4.7 构建最短路径图（SPM）

SPM 是一个数据结构，它将自由空间划分成最大区域，使得区域内所有点到源点的最短路径长度相同，并且具有相同的前驱节点。这样，对于任意查询点，可以直接在 SPM 中查找对应的最短路径信息。

在算法开始时，SPM 是空的。初始化时，将源点 s 作为 SPM 中的一个特殊节点，并将其添加到 SPM 中。使用波阵面扩展算法逐步构建 SPM。波阵面是一个动态扩展的边界，表示已知最短路径的点集合。从源点开始，波阵面向外扩展，直到覆盖整个自由空间。

对于每个透明边（即不包含障碍物的边），执行以下操作：

1　计算波阵面通过透明边时的状态。

更新 SPM 以包含透明边端点的最短路径信息。

2　创建生成器

在波阵面扩展过程中，每个障碍物的顶点可以成为一个生成器。生成器是波阵面上的一个点，代表从源点出发到达该点的最短路径。

3　更新 SPM

当波阵面扩展到新的单元或障碍物边界时，根据生成器的信息更新 SPM，为新覆盖的区域分配最短路径长度和前驱节点。

4　处理双曲线边界

生成器之间的双曲线边界表示波阵面上不同生成器声称区域的分界线。在 SPM 中，这些边界转换为区域边界。

5　波阵面合并

当波阵面从不同方向到达同一单元时，需要合并波阵面。这涉及到，确定哪些生成器声称单元内的点。解决任何冲突，并更新 SPM 以反映合并后的波阵面。

6　处理障碍物边界

当波阵面到达障碍物边界时，若需要，在边界上创建新的生成器。更新 SPM 以包含障碍物边界上的最短路径信息。

当波阵面覆盖整个自由空间时，SPM 构建完成。此时，SPM 包含了从源点到自由空间中每个点的最短路径信息。构建完成后，对于任意查询点，可以在 SPM 中快速查找最短路径长度和前驱节点，从而重构最短路径。

构建 SPM 的过程的时间复杂度为 O(n + h log h)，其中 n 是所有障碍物顶点的总数，h 是障碍物的数量。空间复杂度为 O(n)。

通过以上步骤，算法能够有效地构建 SPM，并利用它来快速响应最短路径查询。SPM 的构建是算法性能的关键，因为它直接影响到查询效率和算法的实用性。

4.8 扩展到一般情况

在处理了凸障碍物的特殊情况后，作者进一步将算法扩展到一般情况，即障碍物可能不是凸多边形。为了处理非凸障碍物，作者引入了扩展走廊结构的概念。这种结构将自由空间分解为三个部分：海洋（Ocean M）、海湾（Bays）和运河（Canals）。

海洋 M 是自由空间中除去所有海湾和运河之后剩余的连通区域。它由所有与障碍物相连的三角形（即在三角剖分中形成的三角形）组成，这些三角形被称为“接头三角形”（Junction Triangles）。海湾是指在扩展走廊结构中，由障碍物所包围的简单多边形区域。当一个沙漏（Hourglasses）是开放的，即其两侧的最短路径不相交时，海湾就形成了。运河是指在扩展走廊结构中，连接两个海湾的细长区域。当一个沙漏是封闭的，即其两侧的最短路径相交时，会在沙漏的两侧形成运河。

首先，在海洋 M 中构建最短路径图 SPM(M)。由于海洋 M 的边界由 O(h)个凸链组成，可以应用修改后的算法来处理这些凸链。算法需要稍作修改以纳入走廊路径，因为走廊路径为波阵面提供了“快捷方式”。走廊路径被视为波阵面的“快捷方式”。如果波阵面击中走廊路径

的端点，它将从走廊路径的另一端出现，但会有等于走廊路径长度的延迟。

接下来，算法将 SPM(M)扩展到所有海湾。每个海湾通过其门与海洋相连。对于每个海湾，算法单独处理，通过门将 SPM(M)扩展到海湾内部，并构建该海湾内的最短路径图。任何最短路径都必须穿过其对应的门。通过门扩展 SPM(M)时，算法实际上解决了一个加权地理最近点 Voronoi 图问题，即在简单多边形内对一组点进行分区，使得同一区域内的点到同一最近点的距离最短。

对运河部分，算法需要将 SPM(M)通过这两个门扩展到运河内，并构建运河内的最短路径图。当通过不同的门扩展 SPM(M)到运河时，可能会得到两个不同的最短路径图。算法需要使用标准的分而治之 Voronoi 图算法中的合并步骤，来合并这两个图，从而得到整个运河内的最短路径图。

最后，通过将 SPM(M)与所有海湾和运河中的最短路径图结合起来，构建出整个自由空间的最短路径图 SPM(s)。构建完成后，对于任意查询点，可以在 SPM 中快速查找最短路径长度和前驱节点，从而重构最短路径。

扩展到一般情况的算法保持了良好的时间复杂度 O(n + h log h)，其中 n 是所有障碍物顶点的总数，h 是障碍物的数量。空间复杂度为 O(n)。

通过这些步骤，算法能够处理包括非凸障碍物在内的一般情况，为更广泛的应用场景提供了有效的最短路径计算方法。

4.9 查询处理

在 SPM 构建完成后，算法已经具备了快速响应查询的能力。SPM 将自由空间划分为若干区域，每个区域内的点到源点的最短路径是已知的。算法接收一个查询点 t 作为输入。这个查询点可以是任意位于自由空间内的点。利用构建的点位置数据结构（如优先搜索树或 k-d 树），快速定位查询点 t 在 SPM 中的位置。这个步骤通常在 O(log n)时间内完成，其中 n 是划分区域的数量。一旦确定了查询点 t 所在的区域，就可以直接从 SPM 中检索出从源点 s 到 t 的最短路径长度以及路径上的关键节点（即前驱节点）。如果需要具体最短路径的几何表示而不仅仅是长度，算法将利用检索到的关键节点信息来重构路径。

5 算法分析

5.1 创新点

1. 时间复杂度优化：作者提出的算法在时间复杂度上进行了优化，达到了 O(n + h log h)，其中 n 是所有障碍物顶点的总数，h 是障碍物的数量。这比之前的 O(n log n)算法有显著改进，尤其是在障碍物数量较少时。
2. 空间复杂度优化：算法的空间复杂度降低到了 O(n)，这通过使用特定的数据结构和优化技术实现，减少了内存的使用。
3. 波阵面扩展：算法采用了波阵面扩展的方法，这是一种高效的路径搜索技术，可以快速确定从源点到目标点的最短路径。
4. 三角剖分的应用：算法利用了三角剖分来简化自由空间的表示，使得后续的路径搜索更加直接和高效。

5. 最短路径图（SPM）的构建：算法能够构建一个最短路径图，这是一个强大的数据结构，可以用于快速响应最短路径查询。

5.2 技术细节

1. 算法结构：算法采用了结构化的方法，首先处理凸多边形障碍物，然后扩展到一般情况，包括非凸障碍物。
2. 数据结构：算法使用了平衡二叉搜索树来维护波阵面和生成器，这支持了高效的插入、删除和查找操作。
3. 波阵面合并：算法详细描述了波阵面合并的过程，这是处理多个波阵面相遇时的关键步骤。
4. 障碍物边界处理：算法特别处理了波阵面与障碍物边界的交互，确保了路径搜索的正确性。
5. 空间优化技术：通过在算法的各个阶段使用空间优化技术，如空间重置和数据结构优化，减少了额外的空间开销。

5.3 优化空间复杂度

作者采取了一些特殊的方法对该算法的时间复杂度进行优化。

在本文的算法中，初始实现可能需要 $O(n + h \log h)$的空间复杂度，其中 n 是所有障碍物顶点的总数，h 是障碍物的数量。这种空间需求主要来自于以下几个方面：

1 波阵面数据结构的存储

2 持久化二叉搜索树用于维护波阵面和生成器

3 中间结果和辅助数据结构

为了降低空间复杂度，作者采取了空间重置，数据结构简化，延迟计算等技术优化算法空间复杂度

通过上述优化策略，作者成功将算法的空间复杂度降低到 $O(n)$，这是一个显著的改进，因为它使得算法能够在更有限的内存资源下运行，同时保持了原有的时间效率。

优化空间复杂度不仅仅是为了减少内存使用，还有助于提高算法的实用性和可扩展性。在资源受限的系统或大规模数据处理场景中，降低空间需求可以使算法更加高效和可靠。

5.3 性能分析

1. 时间效率：算法的时间复杂度得到了显著降低，为 $O(n + h \log h)$，这使得它在处理大规模数据集时更为高效。
2. 空间效率：通过优化空间复杂度，算法能够在保持时间效率的同时，减少对内存的需求，仅有 $O(n)$的空间复杂度。
3. 可扩展性：算法的设计允许它容易地扩展到更复杂的情况，如动态变化的环境或更高维度的空间。

4. 实用性：算法不仅在有理论价值，而且在实际应用中也具有很高的价值，例如地图的路径规划，机器人在复杂环境下的寻路等等。

5.4 分析结论

作者的新算法在时间复杂度上为 O(n + h log h)，空间复杂度为 O(n)，其中 n 是所有障碍物顶点的总数，h 是障碍物的数量。当障碍物相对较少时，该算法相比于先前的工作具有显著的优势。

6 举例说明

假设我们有一个由三个矩形障碍物组成的平面环境，源点 s 位于这些障碍物的左侧，目标点 t 位于右侧。我们的任务是找到从 s 到 t 的最短路径，同时避开障碍物。

1 首先，对自由空间（即没有障碍物的空间）进行三角剖分，将自由空间划分为若干个三角形区域。

2 接着，基于障碍物的顶点构建一个符合子划分，这个划分将平面划分为多个单元，每个单元对应一个或多个障碍物的顶点。

3 从源点 s 开始，创建一个初始的波阵面，这个波阵面以 s 为中心，并且逐步向外扩展。

4 波阵面开始扩展，覆盖自由空间中的点。在每个单元内，波阵面会根据透明边（不包含障碍物的边）和障碍物边缘来更新其状态。

5 当波阵面到达障碍物边界时，算法会在边界上创建新的生成器，这些生成器代表波阵面在障碍物上的新起点。

6 如果波阵面从不同方向到达同一个单元，算法需要合并这些波阵面，确定最终的波阵面状态。

7 随着波阵面的扩展，算法会构建 SPM，这是一个数据结构，它记录了从源点 s 到自由空间中每个点的最短路径信息。

8 一旦 SPM 构建完成，就可以快速查询从 s 到任意点（包括目标点 t）的最短路径。

在三角剖分后，波阵面首先在没有障碍物的自由空间中扩展，形成一个圆形区域。当波阵面遇到第一个矩形障碍物时，它会在障碍物的顶点处创建一个新的生成器，并沿着障碍物的边缘扩展。

随着波阵面的继续扩展，它会在遇到第二个和第三个障碍物时重复这个过程。在每个障碍物的边界上，波阵面都会创建新的生成器，并更新 SPM。

最终，当波阵面覆盖到目标点 t 时，算法会利用 SPM 来确定从 s 到 t 的最短路径。这条路径会避开所有障碍物，并可能是一条折线，表示在障碍物边界上的转折点。

四、不足之处

1　算法实现复杂：该算法涉及到复杂的数据结构（如平衡二叉搜索树）和精细的操作（如波阵面合并、双曲线事件处理等）。这可能导致算法实现起来相对困难，调试和维护成本较高。

2　特殊情况下的性能：虽然算法在一般情况下提供了时间复杂度的改进，但在某些特殊情况下（例如，障碍物数量极少或极多，或者障碍物的布局非常特殊），算法的性能可能不会得到显著提升，甚至可能不如其他专门设计的算法。

3　动态环境适应性：算法对于静态环境的最短路径问题进行了优化，但算法依赖于事先给定的三角剖分，在动态环境中（即障碍物位置或形状随时间变化），需要频繁地更新三角剖分，从而增加了额外的计算负担，适应性和效率可能需要进一步研究。

4　实际应用的考量：在实际应用中，除了计算最短路径之外，可能还需要考虑其他因素，如路径的平滑性、避障策略的鲁棒性等。本文的算法可能需要与其他算法结合使用，以满足这些额外的需求。

5　算法的通用性：算法针对的是二维平面上的多边形障碍物，对于更高维度的空间或不同形状的障碍物，算法可能需要进行相应的调整或优化。

6　计算资源的限制：尽管算法在理论上具有较好的时间复杂度，但在实际执行时，由于涉及大量的数据结构操作和几何计算，可能需要较多的计算资源，这在资源受限的系统上可能是一个问题。

7　对输入数据的敏感性：算法依赖于精细的输入数据，例如障碍物的大小、形状和分布。在真实环境下，无法得到足够精细的数据，可能导致算法效率降低或规划的路径与障碍物产生碰撞。

五、新算法

针对现有算法的局限性，我们提出了一种对原算法的修改方案，旨在提高最短路径算法在动态环境、大规模数据处理和内存使用效率方面的性能。

5.1 修改方案

1　自适应三角剖分

可以采用自适应三角剖分技术[5]，该技术能够根据障碍物的动态变化实时更新三角网格。与传统的静态三角剖分相比，自适应三角剖分减少了预处理的计算成本，因为它仅在障碍物实际发生变化时才进行局部更新。

2　简化数据结构

为了降低实现复杂性，该方案使用简化的数据结构，如优先队列，来维护和更新波阵面。这种简化不仅减少了内存占用，而且提高了算法的执行效率。

3　动态障碍物处理

为避免因环境变化产生过高的计算代价或发生意外，可以引入了基于机器学习的预测模型来预测障碍物的未来位置和运动。这些预测信息被用于提前调整波阵面，从而减少在动态环境中重新计算最短路径的需要，并提前为动态环境的变化做出准备。

5.2 分析

该修改方案通过自适应更新和简化的数据结构，有效地解决了现有算法在预处理和实现复杂性方面的不足。动态障碍物处理机制使得算法能够快速响应环境变化此外，该方案保持时间效率的同时，显著降低了空间需求，使其成为一种在多种环境下都具有竞争力的最短路径算法。

参考文献

[1] [中国]王仁宏等. 计算几何教程[M]. 北京： 科学出版社， 2008：

[2] Joseph S. B. Mitchell. Shortest paths among obstacles in the plane.

[3] J. Hershberger, S. Suri. An optimal algorithm for Euclidean shortest path

[4] 平面中多边形障碍下最短路径的求解[EB/OL] （2011）2011f-d.pdf (tsinghua.edu.cn)

[5] 蒋恒恒;汤宝平;李奇敏. 自适应三角剖分算法及其关键技术研究 [D].重庆：重庆大学，2012：82.