文章编号: 2095-6134(2013)04-0528-11

基于栅格地图的分层式机器人路径规划算法*

余 翀[1†], 邱其文[2]

(1 复旦大学信息科学与工程学院, 上海 200433; 2 南京邮电大学自动化学院, 南京 210046)

(2012 年 4 月 11 日收稿; 2012 年 9 月 11 日收修改稿)

Yu C, Qiu Q W. Hierarchical robot path planning algorithm based on grid map [J]. Journal of University of Chinese Academy of Sciences, 2013, 30(4): 528-538, 546.

摘 要 采用分层规划的思想,给出一种基于栅格地图的最优路径规划算法. 分层路径规划算法的第 1 层为拓扑层规划,采用 Voronoi 图起泡生成算法描述全局可行域的拓扑关系; 第 2 层采用广义水平集算法,解决拓扑层的最优路径搜索问题; 第 3 层为栅格层的路径再规划. 在栅格层借鉴窄带水平集的思想,通过拓宽拓扑路径,得到一个机器人安全通行的窄带区域,并在此区域实行局部快速匹配算法,改善了拓扑路径,提高了算法的效率,并提高规划的实时性.

关键词 分层路径规划; 栅格地图; Voronoi 图起泡生成算法; 广义水平集算法; 局部快速匹配算法

中图分类号: TP24 **文献标志码:** A **doi:** 10.7523/j.issn.2095-6134.2013.04.015

Hierarchical robot path planning algorithm based on grid map

YU Chong[1], QIU Qi-Wen[2]

(1 *School of Information Science and Technology, Fudan University, Shanghai* 200433, *China*;
2 *School of Automation, Nanjing University of Posts and Telecommunications, Nanjing* 210046, *China*)

Abstract Utilizing the hierarchical planning idea, we propose an optimal path planning algorithm based on grid map. The first layer of the algorithm is planning in topology layer, and we adopt Voronoi graph construction frothing algorithm to describe topological relationship of global passable regions. In the second layer, we design generalized level-set algorithm to solve the optimal path search problem in topological layer. The third layer is path replanning in grid layer. Utilizing the narrow-band level-set idea, we obtain a narrow band region that robot can safely pass by broadening the topology path in grid layer, and the local fast marching method algorithm is implemented in this region. It improves the topological path and increases the efficiency and real-time capability of the proposed algorithm.

Key words hierarchical path planning; grid map; Voronoi graph construction frothing algorithm; generalized level-set algorithm; local fast marching method algorithm

* 机器人学国家重点实验室基金(R2200703)资助

† 通信作者, E-mail: dxxzdxxz@126.com

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

移动机器人路径规划问题是机器人研究领域的核心内容之一[1]，就是要机器人在障碍物环境中，寻找一条从出发点到目标点的合适路径，保证按照该路径行走时，不发生碰撞，并且满足一定的约束条件[2]，如行程长度、路径累积转向及离开障碍物距离等. 从对环境信息掌握程度及规划结果出发，可将机器人路径规划问题分为局部路径规划和全局路径规划. 全局路径规划可输出全局环境下的最优路径，但需有环境的先验信息，并且计算量相对较大；局部路径规划则依赖不断更新的由所备传感器获取的局部环境信息，自主获取无碰撞最优路径，很难得到全局最优路径. 但因其处理的信息较少且不断更新，所以局部路径规划更具实用性.

局部路径规划方法主要有：传统方法、智能仿生算法、启发式搜索方法、滚动窗口法、基于行为的路径规划算法，以及基于再励学习的路径规划算法等. 其中，传统方法有：模拟退火法、人工势能法和滚动窗口法等. 智能仿生算法有：人工神经网络算法、遗传算法[3]、蚁群优化算法、粒子群算法、回巢算法和“蛙跳”[4]算法等. A^* 算法是启发式搜索方法的最初代表，在其基础上发展出来的 D^* 启发式搜索算法[5]可看成动态的最短路径算法. 滚动窗口法是基于预测控制理论的一种次优方法，其核心思想是依赖实时检测的局部环境信息，以滚动的方式进行在线规划. 在滚动过程中，以启发的方法生成临时最优子目标，并在当前窗口内给出到达子目标的局部最优路径. 反复这一策略，随着窗口的推进以及新检测信息的获取，机器人获取一条由出发点到目标点的可行路径，实现优化与反馈的结合. 基于滚动窗口的机器人路径优化规划算法是解决未知环境下路径规划问题的一种有效、实用的工具[6].

进行全局路径规划的前提是机器人拥有环境全局信息，即全局地图已知. 不同的路径规划方法适用于不同的地图表示方式. 常用的全局路径规划方法有：栅格法、拓扑图法、可视图法、泰森图法(Voronoi 图法)、自由空间法、最优控制法、遗传算法和 Level-set 方法等. 栅格法也称单元分解法，是将机器人环境分解成一系列带权重的网格，权重表示该网格为机器人可达的空闲网格的概率，通常将这一权重二值化形成栅格地图，并用优化搜索算法在栅格地图上获得一个栅格系列，表示最优路径. 该方法计算量大且随栅格粒度减小而快速增长. 相对于栅格法，拓扑图法是基于拓扑地图进行路径规划，它不能得到一条具体精确的行进路径. 但它降低了计算空间的维度，减小了计算量，算法复杂度仅与拓扑节点数有关且相关理论完备. 最初的可视图法通过组合连接各障碍物顶点且非穿过障碍物的线段，得到通向目标点的可达路径. Voronoi 图法则弥补了可视图法生成路径过于接近障碍物的缺点，当机器人沿 Voronoi 图中的边界行走时，到两边障碍物距离相等且为最远，与两边障碍物发生碰撞的概率最小，行走更安全. 由于具有能处理拓扑改变、数值稳定和独立于参数的优势，Level-set 方法在图像处理及计算机图形学中得到广泛应用[7].

1　分层式路径规划算法的结构

在面向大范围、动态环境下的路径规划应用时，以往单一路径规划方法均显现出它们的不足之处. 单一局部路径规划算法得出的规划结果往往难以达到全局的最优化路径. 单一的全局路径规划算法需要对于环境地图具有一定的先验知识，这就需要较大的存储空间. 另外更为明显的缺陷是，在道路环境复杂度提高，计算距离增长的情况下，单一路径规划算法的效率会大幅下降. 为了平衡各种单一路径规划算法的缺陷，最大程度吸取其优势，本文采用分层次多步骤的路径规划思想，在保证算法能够得到适于机器人运行的最优全局路径的基础上，结合局部路径规划方法，大幅度降低算法运算时间和存储空间，得出一种更加高效优越的分层式机器人路径规划算法. 算法的层次结构如图 1 所示.

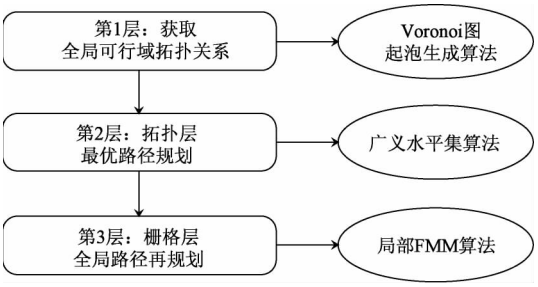


图 1　分层式路径规划算法层次结构图

Fig. 1　Block diagram of the hierarchical path planning algorithm structure

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

2 分层式路径规划算法的具体实现

2.1 生成全局可行域拓扑关系

Voronoi 图是俄国数学家 Voronoi 最早提出的. Voronoi 图具有独特的几何特性,如果将障碍物近似成质点,那么机器人沿着环境障碍物的 Voronoi 边前进时,碰到障碍物的概率最小; 如果障碍物不能近似成质点,可以采用扩展的 Voronoi 图理论进行机器人的路径规划. 因此,Voronoi 图在移动机器人路径规划问题中得到广泛应用.

Voronoi 图又被称为 Dirichlet 图或泰森多边形,是由一系列连接 2 个相邻站点直线的垂直平分线构成的多边形组成. 就二维平面而言,按照最邻近原则绘制 2 个相邻站点的垂直平分线,所得到的中垂线将平面划分成的多边形集合图形称为 Voronoi 图. 这些多边形的边被称为 Voronoi 边,顶点称为 Voronoi 节点. 从二维平面推广到三维空间,则可以定义广义 Voronoi 图,本文下面所述的 Voronoi 图即指广义 Voronoi 图.

2.1.1 广义 Voronoi 图

在二维平面内,如果将点集的概念扩展成几何体的集合,如多边形集合或者多线段集合,那么 Voronoi 图的概念也就扩展成广义 Voronoi 图: GVG(generalized Voronoi graph) ,GVG 中的站点扩展成多边形.

定义点 x 到一个凸集 C_i 的距离 d 及其梯度 ∇d分别为

$$d_i(x) = \min_{c_j \in C_i} \| x - c_j \|, \quad \nabla \boldsymbol{d}_i(x) = \frac{x - c_0}{\| x - c_0 \|},$$

其中,$c_0 = \underset{c_j \in C_i}{\operatorname{argmin}} \| x - c_j \|$,为凸集 C_i 上到 x 距离最短的点,对于凸集来说,距离 x 最近的点是唯一确定的. 梯度是一个单位向量,由 c_0 指向 x.

在 Voronoi 图中,某个站点的 Voronoi 域是同该站点的距离与同其他所有站点之间的距离相比为最小的点所组成的集合. 这个概念可以扩展为广义 Voronoi 域,即某个障碍物的广义 Voronoi 域 F_i 是由到该障碍物的距离比到其他所有障碍物的距离都小的自由空间点所组成的集合,写成集合的形式为

$$F_i = \{ x \mid x \in FS, d_i(x) \leqslant d_j(x), \forall i \neq j \},$$

其中,FS 表示自由空间. 广义 Voronoi 图的基本构成就是到集合 C_i 和 C_j 距离相等的点集,定义为二等距面,它到 2 个障碍物的距离相等,将定义写成集合的形式为

$$S_{ij} = \{ x \mid x \in FS, d_i(x) - d_j(x) = 0, \forall i \neq j \}.$$

注意到二等距面是 2 个相邻的广义 Voronoi 域的边界,即 $S_{ij} = F_i \cap F_j$,所有二等距面就形成了广义 Voronoi 图,即 $\mathrm{GVG} = \cup_{i=1}^{n-1} \cup_{j=i+1}^{n} S_{ij}$. 在二维平面的情况中,到 2 个障碍物距离相等的广义二等距面就是广义等距边,构成了广义 Voronoi 边,广义等距边和广义等距边的交点称为广义交汇点,构成了广义 Voronoi 边顶点.

目前,针对矢量 Voronoi 图生成方法的研究较多,较典型的生成方法有增量法、分治法和间接法等,文献 [8] 给出基于 Level-set 思想的 Voronoi 图生成方法. 但基于栅格地图生成 Voronoi 图的研究较少,目前 2 种主要思路是: 根据栅格距离与矢量距离之间的变换关系来生成 Voronoi 图; 根据栅格地图拓扑关系来生成 Voronoi 图.

本文从第 2 种思路出发,设计基于栅格间拓扑关系的 Voronoi 图生成算法: 起泡算法. 从 Voronoi 图的定义进行反推,经过任意一条 Voronoi 边上的任意一点,绘制一条与该 Voronoi 边相垂直的直线,该直线将与两边障碍物相交得到一条线段,那么该点必为线段的中点. 起泡算法正是采取障碍物主动等速生长的办法,来找出所有这样的中点,从而生成 Voronoi 图. 进一步根据所得 Voronoi 图的特征,设计数据表述和存储结构,获得全局可行路径的拓扑结构图,从而方便后续的全局最优路径查找、路径拼接以及在地图中进行多目标点的路径规划.

2.1.2 起泡算法思路

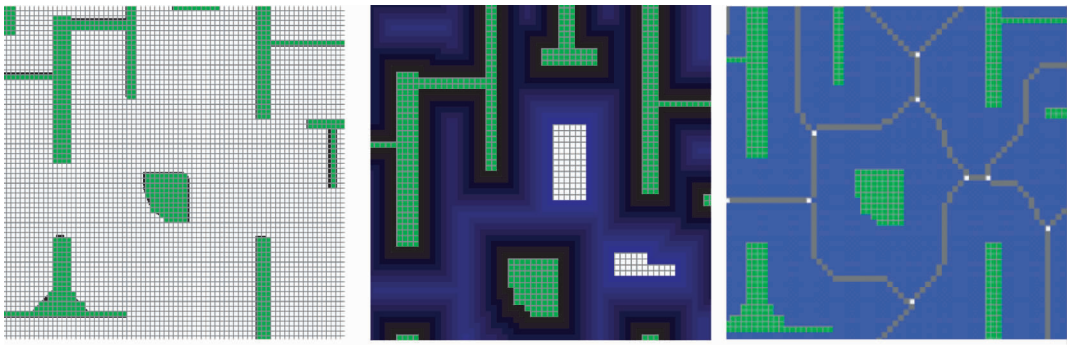
本文提出的算法将栅格地图中的障碍栅格看成热源,其周围自由栅格也会因受热而膨胀起泡,并传递热量,所有栅格受热起泡速度相同. 当某一栅格接受到来自 2 个或 2 个以上热源的热量时,该栅格会因过热成焦灼状态而无法起泡,从而形成 Tough 栅格,也就是最后构成 Voronoi 边的栅格.

起泡算法将栅格地图中的每一个栅格看成膨胀起泡算法关注的胞元,每个栅格在算法初始化和起泡过程中可能呈现出 4 种不同状态,即占据状态、空闲状态、起泡状态以及 Tough 状态. 占据状态和空闲状态是在算法初始化过程中赋予栅格的,其中占据状态表示栅格被障碍物所占据,机器人不可到达; 空闲状态表示栅格为自由栅格,机器

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

人可安全驶过；起泡状态和 Tough 状态是在起泡过程中的更新状态，其中起泡状态表示自由栅格受加热，由空闲状态转换为起泡状态；Tough 状态表示栅格过热焦灼，并且 Tough 栅格无法再次传递热量.

本文基于正方形栅格地图，在 VC6.0 环境下完成 Voronoi 图的生成，地图面积为 1100 × 750 像素. 实验中认为单个正方形栅格与它周围的 8 个栅格相邻，在膨胀起泡过程中，8 个相邻格将同时起泡. 图 2 是 Voronoi 图生成全过程中栅格状态的变化. (a) 中暗色栅格为占据栅格，白色栅格为空闲栅格；(b) 中黑色栅格为由空闲状态转为起泡状态的栅格；(c) 中灰暗色栅格为已经被判定为焦灼的 Tough 栅格.



(a) 初始化　　(b) 起泡过程中　　(c) 生成*Voronoi*图

图 2　Voronoi 图生成全过程

Fig. 2　Whole process of Voronoi graph construction

2.1.3　起泡算法具体流程

按照所给地图栅格的规模分配一个整形数组，用于存放每一个栅格所处状态的标志量 cell_value_flag，算法关注与起泡栅格相邻的空闲栅格. 在气泡向外膨胀推进的过程中，当前被考虑的空闲栅格可能被加热成为起泡栅格，也可能因受多热源加热，过热而成为 Tough 栅格. 新的起泡栅格壮大了原有的气泡，受同一热源加热的所有栅格自成一簇，而 Tough 栅格则将不同的气泡进行分隔.

第 1 判据　由栅格拓扑关系可以看出，在单一热源情况下，当前栅格周围的 8 个相邻栅格中可能有 1 个或者多个起泡栅格给它传递热量. 但即使是有多个起泡栅格给当前栅格传递热量，所有这些起泡栅格在位置上都连成一片，未有断续的情况，即保证是单一热源. 所以判断当前空闲栅格是否受多热源加热的可行判据是，考察当前栅格的 8 个相邻栅格中，能够传递热量的已起泡栅格在位置上是否相连.

第 2 判据　注意到当 2 个气泡相距为偶数栅格时，若只按第 1 判据进行空闲状态栅格的判断，则会出现气泡相互融合，无法找出分割 2 个气泡的 Tough 栅格，也得不到所需要的 Voronoi 边. 即在此种情况下，2 个起泡栅格相互靠近，之间并无 Tough 栅格形成，2 个气泡已经融为一体. 为此，起泡算法需要增加另外一个辅助判据，在第 1 判据将空闲栅格判定为起泡栅格后，再对该栅格进行检验；第 2 判据检测该栅格的外圈 12 个栅格，如果这 12 个栅格中，出现位置上断续的起泡栅格，那么该栅格被判定为 Tough 栅格.

起泡栅格是否相连判据　第 1、第 2 判据中都要对栅格位置是否连续进行判断，判断方法是顺时针扫描 8 个或 12 个栅格，并读取其状态标志变量. 依次将前后 2 个变量值进行异或，如果为 1 则表示出现一次断续；如果出现 4 次或更多次异或结果为 1，则表示这一系列的栅格中，同类栅格出现位置上不连续的情况.

Voronoi 边的修饰　起泡过程中会有多个气泡在同一区域内相遇的情况，从而导致这一区域 Voronoi 边上的 Tough 栅格冗余，如图 3 所示为 3 个气泡相遇，其中浅色栅格为所求 Tough 栅格，而深色栅格则可视为冗余的. 为了方便栅格拓扑地图的构建，有必要对这一区域的 Voronoi 边进行修饰. 修饰方法为：扫描所有的 Tough 栅格，如果某个 Tough 栅格同时有 3 个边邻接的是 Tough 栅格，则将该 Tough 栅格还原为起泡栅格，这样最终被判定为 Voronoi 节点的就只有 3 种点.

起泡算法总流程如图 4 所示，该算法直接通

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

过热源数来实现对不同障碍物产生的气泡的分隔，从而避免了障碍物栅格聚类的判断. 起泡算法作用于少量满足条件的空闲栅格，判断条件较少，且实现方法简单. 因此总体计算量较少，适用于障碍物密集的栅格地图.

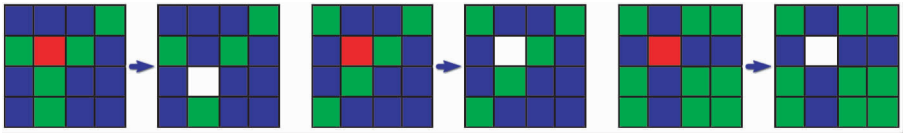


图 3 Voronoi 节点类型

Fig. 3 Voronoi node types

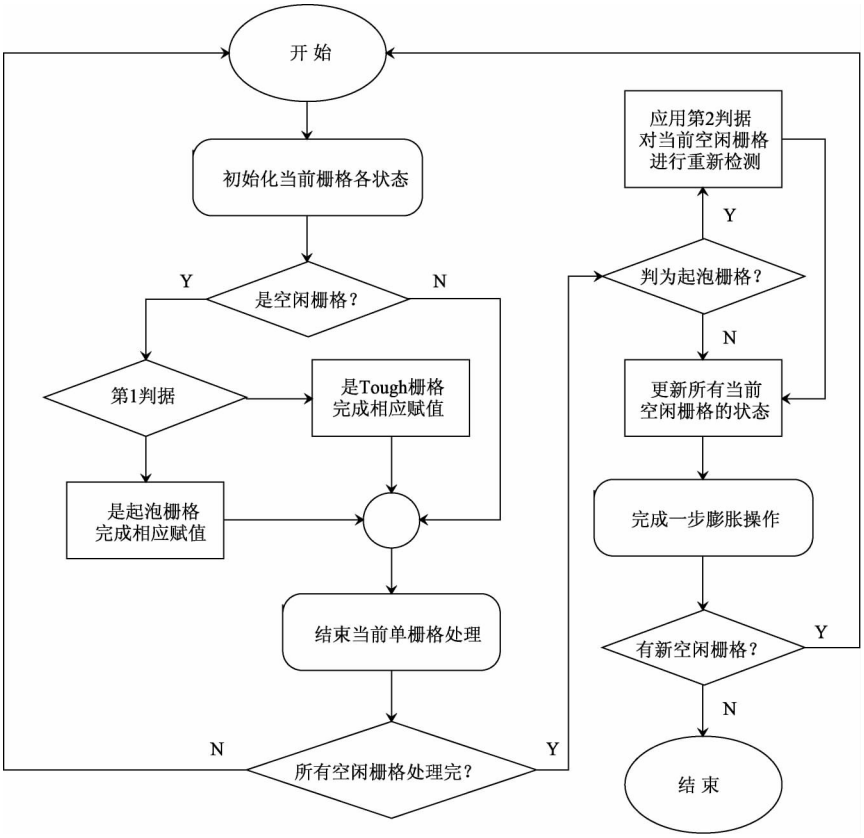


图 4 起泡算法生成 Voronoi 图的流程

Fig. 4 Flowchart of Voronoi graph construction generated by adopting frothing algorithm

2.1.4 栅格化存储拓扑地图

经过修饰后，查找 Voronoi 图中的与 3 个或 4 个 Tough 栅格相邻的 Tough 栅格，将它们作为拓扑图节点. 将 Voronoi 边作为拓扑边，并计算各拓扑节点间的拓扑边所包含的栅格数量，作为该条拓扑边的权重，从而构成栅格拓扑地图. 存储栅格拓扑地图时，以节点作为存储数据单元，每个存储单元包含 9 个子单元，分别用于存放节点自身属性值和 8 个方向上可能存在的拓扑节点的相关信息，包括相邻节点的编号以及到达相邻节点的拓扑边长度. 后续基于栅格拓扑地图的全局路径规划将在该数据结构上进行. 存储单元的数据结构如下:

```
Top_MAP:
{
   Self_ID;
   ( Neighbour_1 ,Len_to_Neighbour_1) ;
          ⋮
   ( Neighbour_8 ,Len_to_Neighbour_8) ;
}
```

在地图中障碍物较松散且分布不均的情况下，栅格拓扑地图中的节点将远离障碍物，从而影

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

响后续规划出的拓扑路径. 为此可以根据障碍物的疏密程度，选取障碍物松散区域的节点进行再次起泡，重构 Voronoi 图，可使拓扑地图更真实地反映全局可行区域的拓扑关系.

2.2　拓扑层最优路径规划

2.2.1　Dijkstra 算法

Dijkstra 算法由起点开始逐步向外探索，在推进的过程中每一步都选取最短路径. 在算法执行的过程中，每个节点都被赋上一个值，这个值可能表示由起点到该节点的最短路径的权值，记作 P 标号；也可能表示由起点到该节点的最短路径权值的上界，记作 T 标号；Dijkstra 算法就是逐步修改 T 标号，使某个满足一定条件的 T 标号变成 P 标号，并且每步只产生一个新的 P 标号. 若起点和终点的最短路径上有 N 个节点，那么最多经过 $N-1$ 次就可以找到从起点到终点的最短路径. Dijkstra 算法具体步骤如下：

第 1 步，给起点以 P 标号$(0,S)$，其中 0 表示起点到自身距离为 0，S 表示该点的上一个节点编号.

第 2 步，用集合 I 和集合 J 分别表示被标号和未被标号的点集，2 类点之间的连接边的集合为$\{(v_i,v_j)\mid v_i\in I,v_j\in J\}$. 当该集合为空时，算法结束.

第 3 步，更新 T 标号为 P 标号. 若节点 v_i 得到 P 标号(l_i,k_i)，寻找$\{(v_i,v_j)\mid v_i\in I,v_j\in J\}$ 中满足条件 l_i+c_{ij} 最小的边，其中 c_{ij} 为该边的权值，将 P 标号给该边的终点，并返回到第 2 步.

重复步骤 2 和 3，直到终点获得 P 标号，或者集合$\{(v_i,v_j)\mid v_i\in I,v_j\in J\}$ 为空. 然后由终点反向回溯即可得到所求的最短路径.

将 Dijkstra 算法应用到上述栅格拓扑地图中，可以得出由 Voronoi 边所组成的最优拓扑路径. 当地图中障碍物密集且分布较均匀时，该算法找到的最优拓扑路径很接近全局最优路径.

仔细分析 Dijkstra 算法可知，该算法基于图的(节点数为 N)结构进行，需要已知图中各节点的关联矩阵、距离矩阵和邻接矩阵等信息，因此进行程序运算求取全局最优路径时，要用到 $N\times N$ 大小的数组. 当地图规模增大时，所需的计算机存储空间剧增，不利于实用，这是 Dijkstra 算法的主要缺点.

2.2.2　广义水平集算法

基于 Dijkstra 算法，并借鉴 Level-set 理论在全局路径规划中的应用思想，本文基于各节点邻接关系的拓扑图，给出内存占用相对较少，算法程序操作更为简便的广义水平集算法.

广义水平集算法将有权图看成各向异质的离散空间，图中各边的权重规定了速度场的方向及其大小，进行 Level-set 操作后，图中各节点将处于不同的 Set 之上，算法可根据 Set 信息获得图中任何节点到起始点的全局最优拓扑路径. 注意到每个节点最多只与 4 节点相邻，算法只需要定义大小为 $4\times N$ 的数组 LevelsetMap，数组单元的定义为{Self_Set_value，Source_node_ID，Len_to_sourcenode，flag_alive}，其中 Self_Set_value 成员存放节点的当前 Set 值，Source_node_ID 记录节点的源节点号.

广义水平集算法具体步骤为

1) 将出发节点的 Self_Set_value、flag_alive 初始化为 1，其他所有节点的 Self_Set_value、flag_alive 初始化为 0，初始算法停止标志 flag_stop 初始化为 0.

2) 将 flag_stop 标志设为 1，然后扫描所有 flag_alive 为 1 的节点，并进行如下操作：

①根据 Top_MAP 查找当前节点的相邻节点，并获取 Len_to_neighbour_i 的值，其中 i 从 1 到 8.

②记录当前节点的 Self_Set_value 为 Set_value_A；扫描当前节点的所有相邻节点，记录其相邻节点的 Self_Set_value 为 Set_value_B；

若

Set_value_B

或

Set_value_A + Len_to_neighbour_i < Set_value_B

则

{

Set_value_B = Set_value_A + Len_to_neighbour_i;

flag_alive = 1;

}

否则

无 Self_Set_value 更新操作

其中，flag_alive_i = 1 将唤醒 Self_Set_value 成员值被更新的相邻栅格.

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

③针对其相邻节点的操作结束后，将当前节点的 flag_alive 置为 0.

④如果有 Self_Set_value 成员进行了更新操作，则将标志 flag_stop 设为 0; 否则无操作.

3) 判断 flag_stop 是否为 1，如果为 1，则进入第 4) 步，否则跳转到第 2) 步.

4) 根据生成的 LevelsetMap 数组，从目标节点出发，回溯读取数组中当前节点的成员 Source_node_ID，形成一个节点 list，并累加 Len_to_sourcenode 得到沿途边长的总和.

广义水平集算法实验结果 广义水平集算法执行一次，就可以给出从出发节点到图中所有其他节点的全局最短拓扑路径，并将该信息存储于 LevelsetMap 数据结构中. 基于该数据结构，从不同目标节点开始回溯，便可得出相应的最优路径，图 5 给出广义水平集算法得出的多条最优路径.

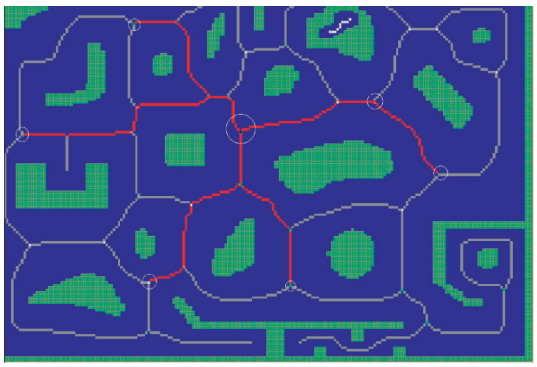


图 5 广义水平集算法得到全局最优拓扑路径

Fig. 5 Global optimal topological path obtained by adopting generalized level-set algorithm

2.3 栅格层全局路径再规划

上一节中，通过 Voronoi 图构建拓扑地图后，利用广义水平集搜索算法求得了一条全局路径. 根据 Voronoi 图的特点可知，由 Voronoi 边所组成的路径拥有较高的安全系数，但前进途中转折较多，且路径长度不是最优的.

本小节基于上一节中得到的全局路径，依据窄带水平集的思想，进行栅格层的全局路径再规划. 当采用 FMM(fast marching method) 方法在栅格地图上进行全局路径规划时，所得结果比传统的基于栅格搜索方法路径更为光滑. 但是随着地图规模的增加，采用水平集方法或者 FMM 方法进行全局路径规划时，都存在计算量随地图的规模迅速增加的问题. 窄带水平集思想就是把构建水平集的任务限定在某一特定区域内，在给定区域外，构建算法不作任何处理，以减小计算量. 这一思想同样适用于 FMM 算法，只要对 FMM 中的代价势函数的有效区域进行限定即可.

2.3.1 水平集方法

Level-set 方法的原始思想比较简单，是由 Osher 和 Sethian 于 1987 年提出的，希望将其用于分析和计算空间界面 Γ 上速度场 v 下的连续运动，其中 v 可能随位置、时间、界面几何特性，以及外界物理作用等而变化. 当时只定义了一个光滑函数 $\varphi(x,t)$，表示 n 维空间的一个开放区域，$\varphi(x,t)=0$ 表示包含该开放区域的曲面. 该函数具有以下性质: 1) 当 $x\in\Omega$ 时，$\varphi(x,t)>0$; 2) 当 $x\notin\Omega$ 时，$\varphi(x,t)<0$; 3) 当 $x\in\partial\Omega=\Gamma(t)$ 时，$\varphi(x,t)=0$. $\varphi(x,t)$ 的值被称为 level，而 $\Gamma(t)$ 被称为 set. 仅在 $\varphi(x,t)$ 消失时，才定位 $\Gamma(t)$，此时界面被捕捉，被隐式定义成函数 $\varphi(x,t)$ 的零等轮廓线，这对数值计算具有重大意义. 隐式函数 $\varphi(x,t)$ 的发展受同 v 相关的对流方程控制，即 $\frac{\partial\varphi}{\partial t}+v\cdot\varphi=0$，其中，$v$ 是界面上的期望速度，在其他地方是任意的.

Ron Kimmel 等最早应用 Level-set 方法，在光滑三维曲面上寻找 2 点间的最短路径[8]. 直观地说，Level-set 方法是通过回溯等距轮廓线来实现优化路径的查找，它能够有效地模拟动态过程. 应用于图像处理时，Level-set 方法最大的优点在于能够用于任何凸凹、分裂和合并的图像处理，能够发现局部和全局极小值，当存在多个 Level-set 函数时，在分割过程中，能完成自动分裂和合并的多项处理任务. Level-set 方法也存在缺点，当一些目标嵌在别的目标中时，它无法捕捉所有的兴趣目标; 另外如果目标中存有间隙，会出现遗漏的情况，而且激波也可能带来振荡解的问题.

Level-set 方法在路径规划中得到较多关注. 其中 Bin Xu 等[9] 提出一种高效的 Level-set 计算方法，应用于在静态但不精确的全局地图中进行路径规划. 文中指出，水平集方法的主要计算量来自于水平集的构造，新方法在机器人探测到新障碍物时，根据水平集的特性，只进行部分水平集的重构，极大地提高了 Level-set 方法的效率，使其更加适用于移动机器人的导航路径规划.

2.3.2 FMM 方法

FMM 是基于 Level-set 理论的界面演变跟踪

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

算法,是一种基于逆风策略求解 Eikonal 方程的高效数值方法,能有效解决静态环境中最短几何路径求解问题. R. Philippsen、Cheng-Yan Wu 等基于 FMM 给出一种动态环境中的路径规划方法. 在格网总数为 N 的矩形直角网格中求解 Eikonal 方程的算法时间复杂度为 $O(N\log N)$. 在二维网格空间中进行规划时,假定给出一个各向异性场 C,寻找平面上 2 点之间的最优路径时,可遵循最小代价条件 $\min\int_{\text{start}}^{\text{goal}} C\mathrm{d}s$. 有势场函数 $u(x)$ 满足方程: $\| \nabla \boldsymbol{u}(x) \| = C, C>0$ 和 $u(\text{goal}) = 0$,其中 x 表示在平面上的位置. 那么只要按最陡梯度下降法,由势函数 $u(x)$ 搜寻一条从出发点到目标点的轮廓线,就可以得到所要求的最优路径[10]. 所述方程中,如果 $C=1$,则势函数可看成各向同性、均匀分布,依此求解的最优路径可看成是简单的基于欧式路径的寻优结果,求解过程可简化. 以单位速度拓展边界曲线,从而形成由出发点开始的一系列等高线,当目标点被等高线所涵盖后,按最陡梯度下降法可回溯出最优路径. 如果 C 不为 1,势函数的各向异性的特性将导致边界曲线在发展过程中和自己交叉,形成多值"燕尾解"[11].

本文的路径规划算法基于栅格地图进行,地图中的栅格分为占据栅格和自由栅格 2 种,机器人在自由栅格中行走,各向同性,从而简化了 FMM 方法在栅格地图中的应用.

2.3.3　窄带生成

在障碍分布密集且较均匀的环境中,采用广义水平集算法得到的全局路径是拓扑意义上的最优,并且已经较接近全局最优路径. 本小节基于该全局路径,进行栅格层的全局路径再规划. 我们仍然从 Voronoi 图的定义出发,以得到的拓扑路径为路基,向两边进行拓宽,这是 Voronoi 边生成的逆过程. 拓宽路基保证了新的路径与两边障碍物存在一定距离,这一约束条件是考虑到机器人体积以及机器人的运动特性而给出的,可以保证机器人按照拓宽后所得路径前进时,与障碍物保持一定的安全距离.

路基拓宽以单条 Voronoi 边作为操作对象,如图 6(a) 所示. 图片左下角的框内为一段 Voronoi 边拓宽后的情形. 路径拓宽过程受到其他 Voronoi 边的限制,这样可以保证新得到的路径是以原拓扑路径为骨架. 图 6(b) 为整条拓扑路径拓宽后的结果.

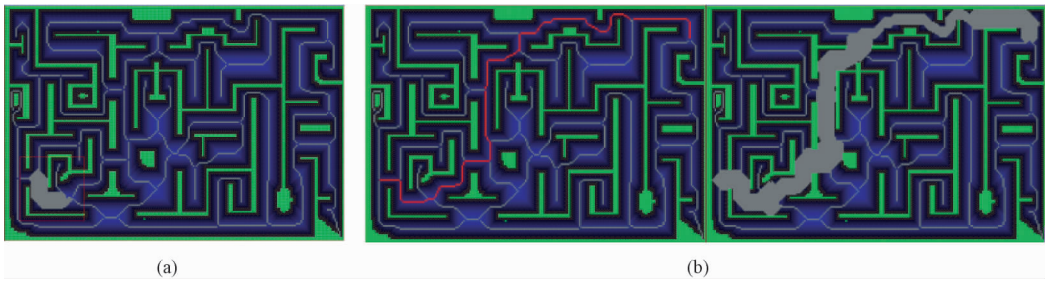


图 6　拓扑路基拓展

Fig. 6　Topology roadbed broadening

2.3.4　局部 FMM 方法

通过路基拓宽得到一个窄带区域,从图 6(b) 中可以明显看出,新生成的窄带路径区域的面积远小于地图总体面积. 所以在进行 FMM 算法路径再规划时,算法所需要覆盖的栅格数也明显小于地图栅格总数,因而在此基础上应用 FMM 算法的计算量也将远小于全局 FMM 路径规划.

由于仿真地图采用正方形网格进行建模,所以在进行局部 FMM 算法时,需要对距离代价进行重新定义. 如图 7(a) 所示,假设任何一个栅格到其相邻的 8 个栅格的距离代价均为 1,并且假定栅格地图中,自由栅格各向同性. 那么由起始点以单位速度向外扩展的边界曲线将是一系列的内切正方形,如图 7(b) 所示. 在图 7(b) 中,灰色斑马线即为等速扩展曲线.

在构造了斑马线式的等速扩展曲线后,从终点出发,按最陡梯度下降法反向回溯到出发点,就可以获得栅格层的最优路径. 但是栅格拓扑距离定义时,某一栅格到其周围 8 个栅格的距离均为 1,为简化回溯过程,回溯算法只检查当前栅格的 8 个相邻栅格. 如图 7(d) 所示,若从 1 号栅格开始回溯,则同时有 3 个相邻栅格距离代价为 1,最

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

陡梯度下降法失效. 本文通过给出默认的优先次序,以解决此种最陡梯度下降法失效的问题. 如图 7(a) 所示对相邻栅格进行编码,在进行路径回溯时,首先判断 0 号相邻栅格与当前栅格是否存在距离代价梯度,如果存在,则将 0 号栅格选定为路径中的下一个栅格; 如果不存在距离代价梯度,则转而判断 1 号栅格,以此类推,如图 7(d) 所示为 2 条回溯路径. 这种方法有效保证了回溯所得路径的方向稳定性,减少了路径的累积转向次数.

局部 FMM 算法最终规划所得的全局路径如图 8 中的条纹栅格序列所示,其中地图栅格单元大小为 5 ×5 pix.

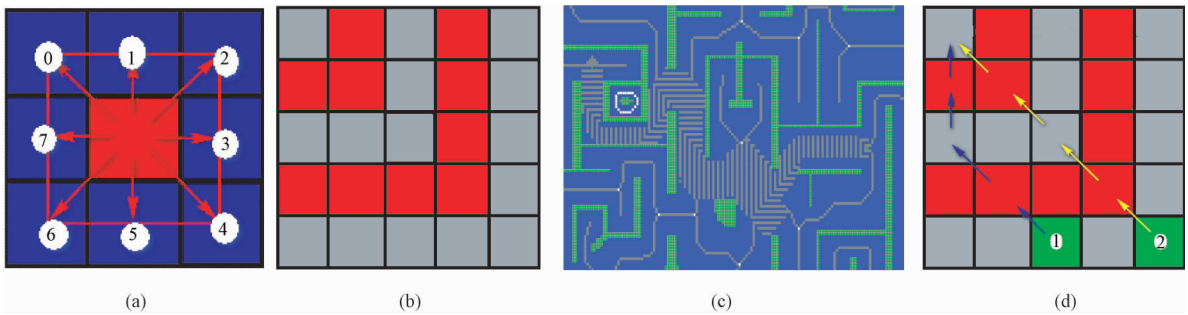


图 7 局部 FMM 算法的等速扩展和路径回溯示意图

Fig. 7 Illustration of constant-velocity-expansion and path backtracking process in local FMM algorithm

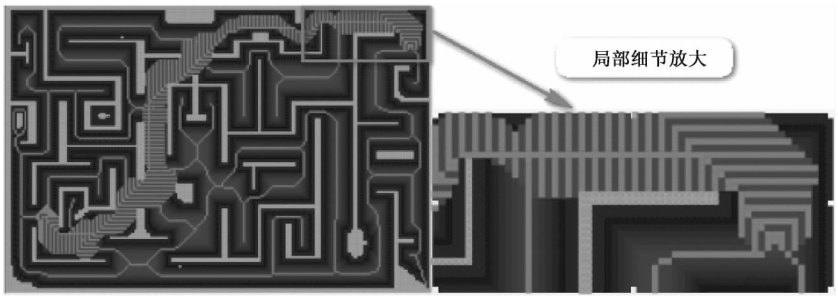


图 8 局部 FMM 算法产生回溯路径

Fig. 8 Backtracking path generated by local FMM algorithm

2.3.5 动态路径规划

由局部 FMM 算法给出栅格层的最优路径后,机器人可沿所得栅格序列前进. 当环境中有动态障碍物占据路径序列栅格时,机器人可以采用经典的滚动窗法,实时避开运动的障碍物. 另外由于已经将路基拓宽生成窄带路径,所以也可以在拓宽得到的窄带路面上采用局部 FMM 方法,实现实时的动态路径规划.

当动态障碍物完全阻碍了规划路径时,可以将所对应的拓扑边设定为断开状态,然后重新启动全局路径规划算法,寻求另一条次优路径.

3 算法分析与对比

为了说明本文提出的分层机器人路径规划算法的效果,在相同的实验条件下,我们先分别和一些经典的单一路径规划算法进行对比,再和一种前人提出的分层路径规划算法进行性能对比分析.

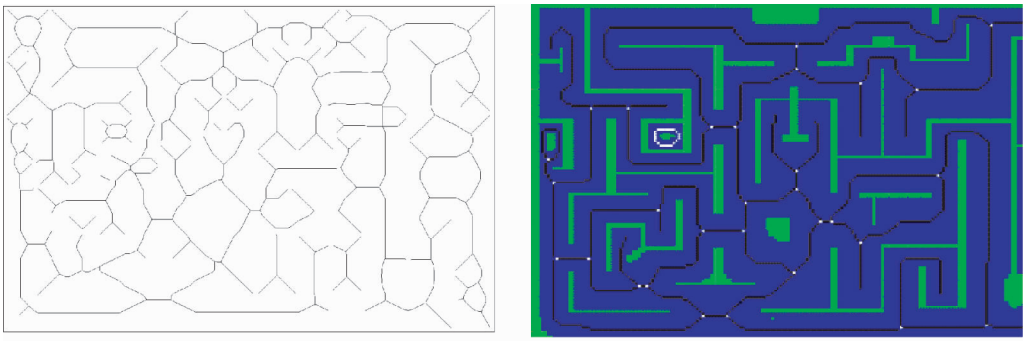
3.1 起泡生成算法的分析与对比

根据栅格地图拓扑关系来生成 Voronoi 图的算法中,比较有代表性的是利用图像处理方法的腐蚀算法,图 9(a) 、9(b) 分别给出采用腐蚀算法和本文提出的起泡算法,对室内环境生成 Voronoi 图的结果. 对比 2 张图可以看出,采用腐蚀算法生成的 Voronoi 图虚假分支较多,不利于对环境几何特征进行描述,这点在文献 [12] 中也有提到. 本文基于栅格地图自行设计 Voronoi 图的起泡生成算法,生成的 Voronoi 图虚假分支很少,Voronoi 节点数量适中,能很好地表示出地图的拓扑结构,更利于对环境几何特征进行描述,所得结果更加有效.

进一步分析可知,腐蚀算法是针对像素点进行运算,时间代价较大. 而本文提出的起泡算法是基于栅格地图进行运算,对于障碍栅格采用并

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

行生长的方法，速度较快，计算耗时很小. 图 9 是一张 1 100 × 750 像素的地图，在 VC6. 0 环境下完成 Voronoi 图的生成过程，采用腐蚀算法耗时 13. 4 s；而采用起泡算法，耗时仅为 1. 1 s.



(a) 腐蚀法生成的Voronoi图　　(b) 起泡法生成的Voronoi图(图中黑线所示)

图 9　Voronoi 图生成算法对比

Fig. 9　Comparison between Voronoi graph construction algorithms: (a) Voronoi graph generated by adopting corrosion algorithm and (b) Voronoi graph generated by adopting frothing algorithm

3. 2　广义水平集算法的分析与对比

通过分析经典的 Dijkstra 算法可知，该算法基于图的(节点数为 N) 结构进行，需要已知图中各节点的关联矩阵、距离矩阵和邻接矩阵等信息. 因此进行程序运算求取全局最优路径时，要用到 $N \times N$ 大小的数组. 当地图规模增大时，所需的计算机存储空间剧增，不利于实用，这是 Dijkstra 算法的主要缺点.

而广义水平集算法可以明显节约数据存储空间，注意到每个节点最多只与 4 节点相邻，算法只需要定义大小为 $4 \times N$ 的数组 LevelsetMap，因此所需存储空间与拓扑地图节点规模呈线性关系.

广义水平集算法执行过程中，针对每个节点都需要访问它的 4 个相邻节点，并做出相应的判断和赋值操作. 假设一次操作一个节点的时间消耗为单位 I，当节点状态被更新时，该节点会被唤醒，增加算法时间复杂度. 理想情况下，如果图中每条边的权重相同，那么所有遍历过的节点都没有被再次唤醒，此时算法的时间复杂度为 $N \times I$. 通过实验可知，针对给定的拓扑带权图，该算法是有限封闭的，并且每个节点最多只影响其相邻的 4 个节点(在本文的拓扑图中，每个节点只有 3 个相邻节点)，在最差的情况下，所有节点都被唤醒，则算法渐近时间复杂度为 $O(N \times N)$.

3. 3　局部 FMM 算法的分析与对比

在进行栅格层的全局路径再规划过程中，如果采用 FMM 方法在栅格地图上进行全局路径规划，所得结果比传统的基于栅格搜索方法路径更为光滑. 但是随着地图规模的增加，采用水平集方法或者 FMM 方法进行全局路径规划时，都存在计算量随地图的规模迅速增加的问题.

我们采用的局部 FMM 方法，保留了全局 FMM 方法的优点，得到的结果依然比传统的基于栅格搜索方法路径更为光滑. 同时对 FMM 方法中代价势函数的有效区域进行限定，将任务限定在局部窄带区域内，而在此区域外不作处理，能够大大减小计算量.

通过图 6(b) 可以看出，路基拓宽得到的窄带区域面积远小于地图总体面积，所以在进行路径再规划时，局部 FMM 算法所需要覆盖的栅格数也明显小于地图栅格总数，因而计算量也将远小于全局 FMM 路径规划.

我们对不同大小和障碍物分布情况下的地图进行实验，得出局部 FMM 算法、全局 FMM 算法和 Level-set 3 种算法消耗时间的曲线图，如图 10 所示.

通过 3 种算法消耗时间的曲线图可以看出，对应相同的地图情况下，局部 FMM 算法耗费的时间远小于另外 2 种算法，说明局部 FMM 算法的计算量相对全局 FMM 算法和 Level-set 算法更小，更加符合实时性的应用要求.

3. 4　整体分层式路径规划算法的分析与对比

为进一步比较说明我们提出的分层式路径规划算法的性能，我们和前人提出的一种分层算法进行比较. 文献 [3] 提出一种在复杂动态环境

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

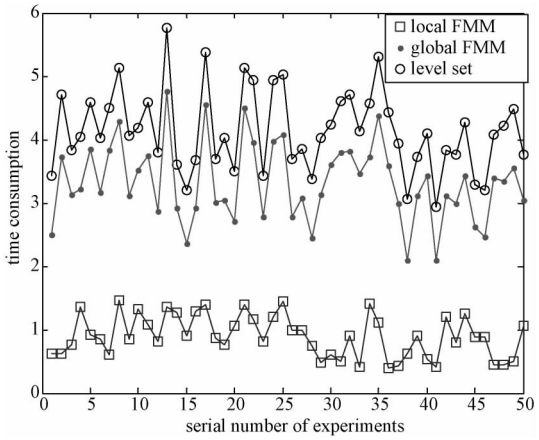


图 10 局部 FMM、全局 FMM、Level-set 算法耗时曲线图

Fig. 10 Time-consuming curves of local FMM algorithm, global FMM algorithm, and level-set algorithm

中,在满足给定时间约束条件下的分层路径规划算法. 算法分为 3 层,第 1 层利用 A^* 算法产生一条全局路径; 第 2 层在满足机器人最大运行速度和时间约束条件情况下,求解最优化问题; 第 3 层结合动态障碍物信息,修正在第 2 层所得到的最优路径. 我们在仿真环境中,在相同条件下分别采用 2 种算法进行 100 次试验,统计结果如表 1 所示.

表 1 本文和文献[13]提出的分层路径规划算法实验结果对比

Table 1 Comparison of experimental results between the hierarchical path planning algorithms proposed in this paper and in reference [13]

	避障成功率/%	平均所用时间/s	平均路径长度/像素
文献算法	100	0.84	1602
本文算法	100	0.89	1315

通过对比文献[13]和本文的分层路径规划算法的实验结果,2 种算法都可以在动态障碍环境下实时完成避障. 文献[13]的算法更多考虑了时间的约束条件,运行的平均时间略低于本文的算法; 但是由于机器人速度的限制,文献[13]的算法得到的平均路径长度大于本文分层路径规划算法,得到的全局路径的优化程度也不如本文算法.

4 结论

在分析和对比几种经典路径规划方法的基础上,本文设计实现了分层式全局最优路径规划算法,解决了任意环境障碍物分布情况下,全局拓扑地图的生成问题. 其中 Voronoi 图起泡生成算法采用并行生长的方法,速度较快,且无需对障碍栅格进行分类,所得结果无虚假骨架,非常适用于描述环境可行域的拓扑结构. 在拓扑层采用广义水平集算法,这种基于无向图的全局最优拓扑路径搜索算法的空间、时间复杂度低,算法运行一次就可以得到所有节点到出发点的全局最优拓扑路径. 栅格层的路径再规划过程很好地弥补了拓扑路径的不足,借鉴了窄带水平集的思想,实现了局部 FMM 算法,使得 FMM 算法可以方便地应用于实时动态路径规划中.

客观地说,本文提出的分层式路径规划算法也存在不足之处,就是在进行路径规划的过程中,对于机器人本身的运动结构考虑较少. 获得的路径虽然在全局环境最优,但是对于实际运动的机器人,可能某些次优路径更适合其行进和控制,这也是今后要进一步研究的课题.

参考文献

[1] Zhu D Q, Yan M Z. Survey on technology of mobile robot path planning [J]. Control and Decision, 2010, 25(7): 961-967(in Chinese).
朱大奇,颜明重. 移动机器人路径规划技术综述 [J]. 控制与决策,2010,25(7):961-967.

[2] Willms A R, Yang S X. An efficient dynamic system for real-time robot-path planning [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2006, 36(4): 755-766.

[3] Zhang Y, Zhang L, Zhang X H. Mobile robot path planning base on the hybrid genetic algorithm in unknown environment [C] // Eighth International Conference on Intelligent Systems Design and Applications (ISDA). China: Taiwan, 2008: 661-665.

[4] Hassanzadeh I, Madani K, Badamchizadeh M A. Mobile robot path planning based on shuffled frog leaping optimization algorithm [C] // Conference on Automation Science and Engineering (CASE). Canada: Ontario, 2010: 680-685.

(下转第 546 页)

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

参考文献

[1] Halpern J, Teague V. Rational secret sharing and multiparty computation [C] // Babai. Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing. Chicago: ACM, 2004: 623-632.

[2] Gordon S, Katz J. Rational secret sharing, revisited [C] // Roberto D P, Moti Y. Security and Cryptography for Networks. Maiori: Springer, 2006: 229-241.

[3] Abraham I, Dolev D, Gonen R, et al. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation [C] // Ruppert. Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing. Denver: ACM, 2006: 53-62.

[4] Lepinski M, Micali S, Peikert C, et al. Completely fair SFE and coalition-safe cheap talk [C] // Chaudhuri. Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing. Newfoundland: ACM, 2004: 1-10.

[5] Izmalkov S, Micali S, Lepinski M. Rational secure computation and ideal mechanism design [C] // 46th Annual IEEE Symposium on. Foundations of Computer Science. Pittsburgh: IEEE, 2005: 585-594.

[6] Kol G, Naor M. Cryptography and game theory: Designing protocols for exchanging information [C] // Ran C. Theory of Cryptography. New York: Springer, 2008: 320-339.

[7] Kreps D M, Wilson R. Sequential equilibria [J]. Econometrica: Journal of the Econometric Society, 1982, 50: 863-894.

[8] Zhang Z, Liu M L. Rational secret sharing as extensive games [J]. Science China Information Sciences, 2010: 1-13.

[9] Herzberg A, Jarecki S, Krawczyk H, et al. Proactive secret sharing or: How to cope with perpetual leakage [C] // Advances in Cryptology-CRYPTO'95. Berlin: Springer, 1995: 339-352.

[10] Shamir A. How to share a secret [J]. Communications of the ACM, 1979, 22(11): 612-613.

[11] Osborne M J, Rubinstein A. A course in game theory [M]. Cambridge: the MIT Press, 1994.

[12] Pedersen T. Non-interactive and information-theoretic secure verifiable secret sharing [C] // Feigenbaum J. Advances in Cryptology-CRYPTO'91. Berlin: Springer, 1992: 129-140.

[13] Hendon E, Jacobsen H J, Sloth B. The one-shot-deviation principle for sequential rationality [J]. Games and Economic Behavior, 1996, 12(2): 274-282.

(上接第538页)

[5] Guo J M, Liu L, Liu Q, et al. An improvement of D* algorithm for mobile robot path planning in partial unknown environment [C] // Second International Conference on Intelligent Computation Technology and Automation (ICICTA). China: Hunan, 2009: 394-397.

[6] Zhang C G, Xi Y G. Robot rolling path planning based on locally detected information [J]. Acta Automatica Sinica, 2003, 29(1): 38-44(in Chinese).
张纯刚,席裕庚. 基于局部探测信息的机器人滚动路径规划 [J]. 自动化学报, 2003, 29(1): 38-44.

[7] Yang Y W, Yang J Y, Gong L. The solution for robot path planning based on Level Set method [J]. Journal of Image and Graphics, 2005, 10(9): 1139-1145(in Chinese).
杨余旺,杨静宇,龚璐. Level Set 方法求解机器人路径规划的探讨 [J]. 中国图象图形学报, 2005, 10(9): 1139-1145.

[8] Kimmel R, Amir A, Bruckstein A M. Finding shortest paths on surfaces using Level Sets propagation [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995, 17(6): 635-640.

[9] Xu B, Stilwell D J, Kurdila A. Efficient computation of Level Sets for path planning [C] // International Conference on Intelligent Robots and Systems (IROS). USA: Louis, 2009: 4414-4419.

[10] Philippsen R, Siegwart R. An interpolated dynamic navigation function [C] // Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Spain: Barcelona, 2005: 3782-3789.

[11] Scheding S, Dissanayake G, Nebot E M, et al. An experiment in autonomous navigation of an underground mining vehicle [J]. IEEE Transactions on Robotics and Automation, 1999, 15(1): 85-95.

[12] Li M, Wang J, Zhu M Q. On skeleton extraction algorithm for path planning of mobile robots in complex planar maps [C] // 29th Chinese Control Conference (CCC). China: Beijing, 2010: 3704-3708.

[13] Minhyeok Kwon, Heonyoung Lim, Yeonsik Kang, et al. Hierarchical optimal time path planning method for an autonomous mobile robot using A* algorithm [C] // 2010 International Conference on Control Automation and Systems (ICCAS). Korea: Gyeonggi-do, 2010: 1997-2001.

?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net