



南京大學

本科毕业论文

院 系 地理与海洋科学学院

专 业 地理信息科学

题 目 基于体素模型的监控摄像自动化布点算法研究

年 级 2014 学 号 141160033

学生姓名 廖祥森

指导老师 余江峰 职 称 副教授

论文提交日期 2018 年 5 月 20 日

南京大学本科毕业生毕业论文（设计、作品）中文摘要

题目： 基于体素模型的监控摄像自动化布点算法研究

地理与海洋科学学院 院系 地理信息科学 专业

2014 级本科生姓名： 廖祥森

指导教师（姓名、职称）： 余江峰 副教授

摘要：

随着城市规模的扩大，城市安全问题受到越来越多的关注。监控摄像系统具有信息量大、还原性高、实时性强等特点，是获取目标区域信息的有效手段，对城市安全的保障提供了有力支持。为了在预算有限，即监控摄像机数目固定的情况下增强监控摄像系统的布点效果，本文提出一种基于体素模型的监控摄像自动化布点算法。算法以 CityGML 城市模型作为输入，首先将城市模型体素化，初始化生成位于道路上且符合泊松圆盘分布的监控摄像候选点，使用局部搜索算法对候选点的位置和朝向参数进行优化调整，得到局部最优的监控摄像候选点，最后通过改进的贪心算法在监控摄像机数目一定的限制条件下从调整后的候选点中生成最优的布点方案，使得布点方案覆盖的目标区域最大。实验结果表明，算法的适用性较高，可以在较短的运行时间内取得较好的布点效果。

关键词：

监控摄像布点；体素模型；局部搜索算法；贪心算法

南京大学本科生毕业论文（设计、作品）英文摘要

THESIS: An Algorithm on Automatic Monitor Placement

Using Voxel-Based Model

DEPARTMENT: School of Geography and Ocean Science

SPECIALIZATION: Geographic Information Science

UNDERGRADUATE: Liao Xiang-sen

MENTOR: Associate Professor She Jiang-feng

ABSTRACT:

With the expansion of the city, the issue of urban security has received more and more attention. As the surveillance monitor system having characteristics of great information, strong facticity and high real-time performance. It is an effective means for obtaining information of target area and plays a significant role in the protection of urban security. In order to enhance the coverage rate of monitor placement with limited budget, that is, the number of monitor is fixed, we present a Voxel-Based monitor automated place algorithm. The algorithm takes CityGML city model as input, first voxelizes the city model, initializes and generates the set of candidate monitor points on the road network which conforms to the Poisson disk distribution. Then we use local search algorithm to optimize and adjust the position and orientation parameters of the candidate points which are local optimal. Finally, the improved greedy algorithm has been used to select limited number of monitors. An optimal monitor placement plan is generated among all candidate monitor points so that the target area covered by the monitor placement plan is the largest. The experimental results show that the algorithm has good compatibility and high operating efficiency, and can achieve better results in a shorter running time.

KEY WORDS: monitor placement; Voxel-Based model; local search algorithm; greedy algorithm

目 录

1. 绪论.....	1
1.1 研究目的和意义.....	1
1.2 研究现状.....	1
1.3 研究内容.....	3
2. 数据生成与预处理.....	6
2.1 生成 CityGML 城市模型.....	6
2.2 生成体素模型.....	6
2.2.1 提取三角面片模型.....	7
2.2.2 三角面片体素化.....	9
2.3 生成监控摄像候选点.....	11
3. 布点方案生成.....	14
3.1 监控目标可见性检测.....	14
3.1.1 FoV 检测.....	14
3.1.2 视线遮挡检测.....	15
3.2 调整摄像机参数的局部搜索算法.....	16
3.3 改进的贪心算法.....	17
4. 实验结果与讨论.....	19
4.1 实验环境.....	19
4.2 实验流程与结果.....	19
4.3 监控摄像布点效果的影响因素.....	20
4.3.1 摄像机数目.....	20
4.3.2 体素分辨率.....	21
4.3.3 生成候选点的随机算法.....	22
4.3.4 摄像机间最小距离.....	23
5. 结论与展望.....	24
参考文献.....	25
本科期间相关的科研成果.....	28
致谢.....	29

1. 绪论

1.1 研究目的和意义

随着城市规模的扩大以及城市人口的增长，城市安全问题受到越来越多的关注^[1-2]。城市重点目标(主干道路、商业中心、火车站、机场等人流密集区域)内的监控摄像系统是城市安全防范系统中的重要组成部分。监控摄像系统具有信息量大、还原性高、实时性强等特点，是获取目标区域信息的有效手段，对城市安全的保障提供了有力支持^[3-4]。

近年来，很多城市都建立了城市公共安全监控摄像系统，在城市的公共安全领域发挥出越来越重要的作用。随着“平安城市”、“智慧城市”等目标的提出，对城市监控摄像系统的覆盖率也提出了更高的要求。此外，室内场景监控、地下监控、移动监控等多样化的监控场景也对监控摄像系统方案设计的通用性提出了更高的要求。

监控摄像系统的覆盖率与监控摄像的布点有着十分密切的关系^[3]。然而，传统的监控摄像布点通常是基于人的直觉和设计经验来进行的^[5]，受主观因素的影响较大，缺乏整体规划，监控盲区大，监控效率不高，耗时耗力，效果也不理想。监控摄像的布点需要综合考虑到摄像机的空间位置、朝向(方位角和俯仰角)、摄像机内部参数(分辨率、可视角度、镜头焦距等)、使用成本等因素，影响因素复杂，也难以仅凭人工来完成监控摄像的布点方案设计。

因此，如何使用一种自动化的方法，优化生成监控摄像的布点方案，在成本有限的限制条件下，最大程度地提高监控摄像的覆盖范围成为了一个大家普遍关注的问题^[3]。

1.2 研究现状

预算有限（即给定监控摄像机数目）条件下的监控区域最大范围覆盖，是一种监控摄像的布局优化问题，属于多传感器网络（Multi Sensor Networks）的布局优化问题^[6]。关于多传感器网络的布局优化问题已开展了许多研究，在算法层面已取得了诸多研究成果。但监控摄像的传感模型也有自己的特点，与其他传感

器的全方向（omnidirectional）传感模型不同，监控摄像的传感模型是有方向（directional）和有边界（bounded）的^[9]。因此，多传感器网络的布局优化算法不能直接应用于监控摄像的布局优化问题上。

国内对于监控摄像系统的研究大多从信号传输或嵌入式设备设计的角度展开，对于监控摄像的布局优化问题研究较少，国外学者对此的研究相对比较成熟。关于这个问题的早期研究主要基于计算几何中的可见性问题展开，如：美术馆问题^[7]（Art Gallery Problem，如何使用最少的守卫看守美术馆，并使得美术馆的每个角落都落在守卫的视野内）、探照灯问题^[8]（The Floodlight Problem，给定一个二维线段 $[l, r]$ 和 n 个探照灯，求解这些探照灯能照到线段的最大范围）。但这些问题认为传感器具有全方位的可视角度和无限的可视距离，并且将研究区域抽象为二维平面，与实际问题不符。

监控摄像的布点问题可以抽象为三维空间内的可见性问题（即如何分布若干个传感器，使得这些传感器可以观察到一个多面体的各个区域）。Cole 和 Sharir 等学者证明了三维空间内的可见性问题为 NP 困难（non-deterministic polynomial-time hard，简称 NP-Hard）问题^[6]。NP-Hard 问题解的精确性在有限的时间内很难去验证，因此这类问题求精确解的时间复杂度高，通常使用近似算法（Approximate Algorithm）来求解，在可以接受的时间复杂度内找到一个局部最优的解。

Debaque 和 Jedidi 等人认为监控摄像的布点问题是一个多目标全局优化问题，并综合考虑了最大化覆盖问题、目标冗余覆盖问题和最小化目标覆盖误差问题，通过蒙特卡洛采样的方法遍历研究区域寻找最优解^[10]；Chakrabarty 和 Iyengar 等人认为监控摄像的布点问题是一个 k 度覆盖问题（ k -coverage problem），即每一个目标需要被至少 k ($k \geq 1$) 个摄像机所覆盖，研究将每个摄像机的覆盖范围抽象成不同半径的圆，使用整型线性规划的方法求解^[11]；考虑到对于同一个物体在不同角度观察的结果不同，Kit Yee 等人提出了全角度覆盖的概念，即监控目标的边缘需要被传感器完整覆盖，建立覆盖关系无向图，将问题转化为最短路径问题，使用 Dijkstra 算法求解^[12-13]；Yaagoubi 等人根据场景建筑物的基底（footprint）生成 Voronoi 图，摄像机的候选位置位于 Voronoi 边上，根据场景的数字表面模型来计算摄像机的视域，基于代理人模型求解^[14]。

也有些学者使用元启发式算法来求解监控摄像的布点问题。Al-Hmouz 和 Challa 等人在二维的环境下，基于多目标（覆盖区域最大和覆盖区域存在一定的重叠）优化的角度，定义目标函数，使用多目标遗传算法求解^[15]；Morsly 和 Aouf 等人提出了一种二进制粒子群优化算法，在二维和三维的监控环境中都可以取得较好的效果^[16]；Gupta 等人提出了一种基于非优势排序的遗传算法，算法同时兼顾精英保留策略和多样性保留策略^[17]。

以上这些研究大多是基于场景的矢量面片(Mesh)模型来求解，计算量较大，并且较少考虑到摄像头视域被遮挡的问题，很难适用于大规模真实三维城市场景，而且很多研究对监控目标的三维模型也有特殊的要求，难以处理复杂三维监控目标。

1.3 研究内容

本文提出一种基于体素 (Volume Pixel，简称为 Voxel，类似二维空间的最小单位—像素，是三维空间分割上的最小单位) 模型的三维城市场景监控摄像自动化布点算法。算法首先将研究区域体素化，初始化生成符合泊松圆盘分布的监控摄像候选点，使用局部搜索算法对初始点进行优化调整，最后通过改进的贪心算法在监控摄像机数目一定的限制条件下从所有候选点中生成最优的布点方案，使得布点方案覆盖的目标区域最大。

本文的技术路线如图（图 1-1）所示，主要包含数据生成与预处理、布点方案生成两大步骤。



图 1-1 技术路线图

数据的生成和预处理: 首先使用 Biljecki Flip 设计的程序化建模引擎 Random 3DCity^[18]生成 CityGML (City Geography Markup Language, 是一种基于 XML 格式的用于存储三维城市模型的开放数据模型)格式的 LOD3.1 层次(Level of Detail, 细节层次, 用于描述模型的精细程度, 从 LOD0 到 LOD4, 建筑物模型越来越精细) 的三维城市模型; 接着将 CityGML 模型按语义提取转化为表示建筑物不同构件的 Mesh 模型, 将 Mesh 模型体素化, 按体素所属的建筑物构件类型给体素赋予不同的标识值; 提取体素模型中特定类型(如建筑物入口等)的体素作为待监控目标, 也可任意指定待监控目标; 根据 CityGML 模型提取 AABB (Axis-

aligned Minimum Bounding Box, 轴对齐最小包围盒, 含义为包含目标对象且平行于坐标轴的最小六面体), 在其 XOY 平面上生成符合泊松圆盘分布 (Poisson Disk Distribution) 的监控摄像候选点, 并使用道路网基底对候选点进行裁剪, 得到位于道路网上方的监控摄像候选点。

布点方案生成: 首先对于监控目标进行可见性检测生成监控摄像候选点与待监控目标之间的可见性矩阵; 接着使用局部搜索算法 (Local Search Algorithm) 对预先生成的监控摄像候选点的位置、水平角以及俯仰角进行调整, 找到局部的最优点位, 并更新可见性矩阵; 接着根据可见性矩阵使用改进的贪心算法 (Greedy Algorithm) 在监控摄像机数目一定的限制条件下, 从所有候选点中生成最优的布点方案, 使得布点方案覆盖的目标区域最大。

实验结果表明, 该方法基于体素模型实现, 适用性较高, 可以在较短的运行时间内取得较好的布点效果。

2. 数据生成与预处理

2.1 生成 CityGML 城市模型

研究的初始数据是使用 Biljecki Flip 设计的程序化建模引擎 Random 3DCity 生成的 CityGML 格式 LOD3.1 层次的三维城市模型^[18]。

Random 3DCity 使用 python 语言编写而成，包含“randomiseCity”和“generateCityGML”两个工具。“randomiseCity”能按一定的规则随机生成建筑物的各种特征信息（空间位置、基底尺寸、屋顶高度、屋顶类型、建筑层数、入口位置等信息）并将其以 XML 的格式存储在文件中。“generateCityGML”根据不同级别建筑物 LOD 模型的定义，从“randomiseCity”生成的 XML 格式的建筑物特征信息文件中读取相应 LOD 级别应包含的建筑物特征信息，由此生成建筑物模型，并给建筑物赋予 UUID（unique identifier，唯一标识符），最后将结果以 CityGML 的格式存储在文件中。使用 Random 3DCity 生成的 CityGML 城市模型如图（图 2-1）所示。



图 2-1 CityGML 城市模型

2.2 生成体素模型

将待监控目标、摄像机候选点以及整个三维城市模型按照一定的空间分辨率离散化存储于三维的体素格网中，可大大减少整个监控摄像布点方案的计算时间，也可避免由于数据规模过大引起的运行过程中内存空间溢出等问题。因此，需要对 CityGML 城市模型进行体素化。

在 CityGML 城市模型中，三维建筑物由建筑物的不同组件（如：GroundSurface 地面、RoofSurface 屋顶、WallSurface 墙体、Window 窗户、Door 门等）组合而成。每个组件都以一个有孔多边形（如组件较为复杂，则拆分为多个可用有孔多边形表示的简单组件）来表示，整个 CityGML 城市模型可由若干个有孔多边形组合而成。

直接从有孔多边形生成体素模型较为困难，而对三角面片进行体素化则较为简单。因此从 CityGML 城市模型生成体素模型包含两个步骤：从 CityGML 有孔多边形中提取三角面片模型；对三角面片进行体素化。

2.2.1 提取三角面片模型

CityGML 是一种 XML 类型的建筑物文件格式，本文使用 XML 解析工具库 Xerces（Apache 基金会支持下的一个用于提取、验证、序列化、修改 XML 格式文件的工具库）将 CityGML 文件按照 XML 语法规则解析生成一个包含建筑物语义信息的树形数据结构，再提取每个建筑物组件的 gml::Polygon 有孔多边形。CityGML 中的有孔多边形表示形式如下图（图 2-2）所示，由一个外轮廓 gml::exterior 和若干个内轮廓 gml::interior 组成，轮廓又由一系列点构成，内轮廓之内的部分表示多边形内的孔洞，外轮廓与内轮廓之间的部分即为多边形。

```
<gml:Polygon gml:id="f4a8105d-2424-45af-9a13-ba97fccca2774">
  <gml:exterior>
    <gml:LinearRing>
      <gml:posList>-6.0 -6.0 0.0 245.0 -6.0 0.0 245.0 225.0 0.0 -6.0 225.0 0.0 -6.0 -6.0 0.0</
      gml:posList>
    </gml:LinearRing>
  </gml:exterior>
  <gml:interior>
    <gml:LinearRing>
      <gml:posList>-1.0 -1.0 0.0 -1.0 34.0 0.0 34.0 34.0 0.0 34.0 -1.0 0.0 -1.0 -1.0 0.0</
      gml:posList>
    </gml:LinearRing>
  </gml:interior>
</gml:Polygon>
```

图 2-2 CityGML 中的有孔多边形

对这个有孔多边形进行三角化 (triangulate) 即可得到最终所需的三角面片模型。本文使用 Jonathan 等人开发的二维三角格网化工具 Triangle^[19] 对多边形进行三角化，从 CityGML 有孔多边形中提取三角面片的算法细节如算法 1 所示。

算法 1 从有孔多边形中提取三角面片

输入: 有孔多边形外轮廓 Exterior; 有孔多边形内轮廓 Interiors

输出: 三角面片 Triangles

```
1: vertices  $\leftarrow$  {Exterior 与 Interiors 中所有点}
2: holes  $\leftarrow$  { 每个 Interiors 代表的多边形内的任意一点 }
3: segments  $\leftarrow$  { 多边形边的端点在 vertices 中的索引 }
4: Triangles  $\leftarrow$  {}
5: if 多边形与 YOZ 平面平行 then
6:   for each point  $\in$  vertices, point  $\in$  holes do
7:     point.x, point.y, point.z = point.y, point.z, point.x
8:   end for
9: else if 多边形与 XOY 平面垂直 then
10:   for each point  $\in$  vertices, point  $\in$  holes do
11:     point.y = point.z
12:   end for
13: end if
14: plane  $\leftarrow$  Plane(vertices)
15: Triangulate 只能对二维的多边形进行三角化, 将 vertices 与 holes 投影到 XOY 平面上
16: resTris  $\leftarrow$  Triangulate(verticesProj, segments, holesProj)
17: // 将三角化结果投影回原多边形所在平面
18: for each tri  $\in$  resTris do
19:   triAdj  $\leftarrow$  {}
20:   for each vert  $\in$  tri do
21:     if 多边形与 YOZ 平面平行 then
22:       triAdj.add(Point(vert.z, vert.x, vert.y))
23:     else if 多边形与 XOY 平面垂直 then
24:       triAdj.add(Point(vert.x, Proj(plane, vert.x, vert.y), vert.y))
25:     else
26:       triAdj.add(Point(vert.x, vert.y, Proj(plane, vert.x, vert.y)))
27:     end if
28:   end for
29:   Triangles.add(triAdj)
30: end for
31: return Triangles
```

算法 1 CityGML 有孔多边形三角化算法

算法的输入为有孔多边形的外轮廓 Exterior 和内轮廓 Interiors，输出为该多边形三角化后的结果 Triangles。Triangle 工具需要的输入参数有三个：多边形内外轮廓顶点坐标 vertices、多边形边的端点在 vertices 中的索引 segments，多边形每个内部孔洞中的任意一点 holes。由于 Triangle 工具只能对二维平面上的多边形进行三角化，因此需要先将 vertices 投影到 XOY 平面上，使用 Triangle 工具对 XOY 平面上的有孔多边形三角化后，再将三角化后的结果投影回原本所在平面。

对于和 XOY 平面垂直的多边形，直接投影会使得多边形投影为一条线段，因此先将多边形旋转到与 XOY 平面平行后（交换多边形的 X, Y, Z 值），再投影到 XOY 平面上，三角化后，将三角形按相同的规则旋转回原本所在平面。

平面有孔多边形使用 Triangle 工具三角化后的结果如图（图 2-3）所示。其中顶点 A、B、C、D 为多边形的外轮廓 Exterior，顶点 E、F、G、H 为多边形的内

轮廓 Interior，顶点 O 为多边形内部孔洞内的任意一点，三角化后生成 8 个三角形。

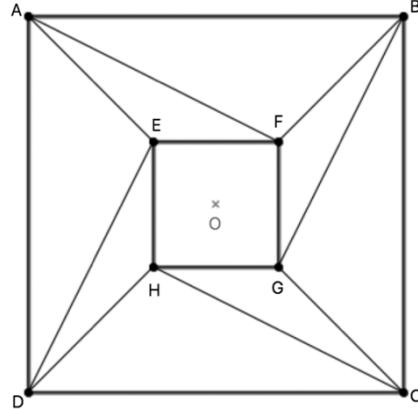


图 2-3 有孔多边形三角化结果

2.2.2 三角面片体素化

对上一步生成的三角面片模型进行体素化的算法细节如下（算法 2）所示。

算法 2 三角面片体素化

输入：三角面片模型 Triangles; 体素分辨率 Res

输出：体素模型 Voxels

```

1: Voxels  $\leftarrow \{\}$ , ResH  $\leftarrow 0.5 * Res$ 
2: for each triangle(p1, p2, p3)  $\in$  Triangles do
3:   if Area(triangle)  $< 1e - 6$  then
4:     continue
5:   end if
6:   // 计算三角形 AABB 的两个顶点 lowerP, upperP
7:   // 以体素中心的 Hash 值为 Key, 建立 HashMap, 避免邻近体素被重复提取
8:   voxelMap  $\leftarrow newHashMap()$ 
9:   // 遍历 AABB 内的所有体素
10:  for x = lowerP.x; x < upperP.x; x += Res do
11:    for y = lowerP.y; y < upperP.y; y += Res do
12:      for z = lowerP.z; z < upperP.z; z += Res do
13:        sVoxel.lowerP  $\leftarrow (x - ResH, y - ResH, z - ResH)$ , sVoxel.upperP  $\leftarrow (x + ResH, y + ResH, z + ResH)$ 
14:        sVoxel.center  $\leftarrow (sVoxel.lowerP + sVoxel.upperP) / 2$ 
15:        // 判断体素与三角面片是否相交, 通过 Hash 值判断是否存在邻近体素
16:        if Overlap(sVoxel, triangle) and voxelMap.exist(Hash(sVoxel.center)) then
17:          Voxels.insert(sVoxel.center)
18:          voxelMap.add(Hash(sVoxel.center), sVoxel.center)
19:        end if
20:      end for
21:    end for
22:  end for
23: end for
24: return Voxels

```

算法 2 三角面片体素化算法

算法的输入是三角面片数组 Triangles(不同的三角面片间会存在共享的顶点，每个三角形单独用三个顶点坐标来表示会造成大量的数据冗余。在实际应用中，

使用一个顶点数组来存储所有顶点，使用一个二维索引数组来代表相应三角形三个顶点在顶点数组中的索引）以及预先设定的体素空间分辨率 Res，输出是体素数组 Voxels。算法提取面片模型中的各个三角形的 AABB，将 AABB 划分为三维体素格网，遍历三维体素格网，判断体素与对应的三角形是否存在重叠（使用 Akenine-Mfoller 提出的“三角形与长方体快速重叠检测算法”^[20]），如重叠则将该体素插入算法的输出体素数组中。为防止相邻体素的重叠，使用 Teschner 等人提出的 Hash 函数^[21]对体素的中心点求 hash 值，将结果存储在 HashMap 中，如果新生成的体素的 hash 值与 HashMap 中的值冲突，则认为该体素和已有体素重叠，丢弃该体素。

使用该体素化算法生成的城市体素模型效果如图（图 2-4）所示。

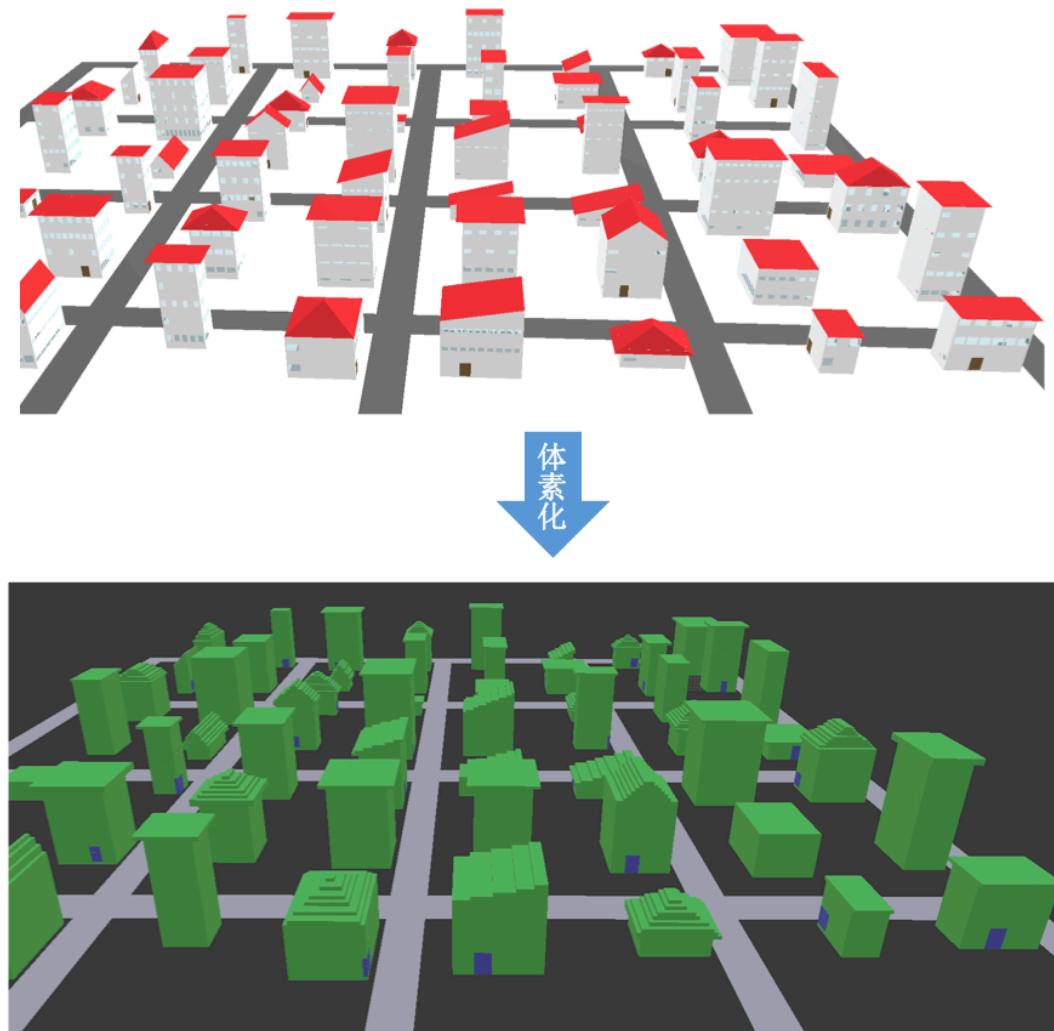


图 2-4 城市模型体素化效果

算法输出的体素由体素中心的三维坐标来表示，坐标需占用三个浮点型的存储空间。为减少数据存储的压力，需将输出体素数据进行处理，将体素坐标根据

分辨率取整，只需占用三个整型 (x_i, y_i, z_i) 的存储空间，体素的中心坐标 $P(x, y, z)$ 可通过场景的 AABB 的顶点 $\text{lowerP}(x_0, y_0, z_0)$ 和分辨率 Res 计算得出：

$$x = x_0 + x_i * \text{Res}; \quad y = y_0 + y_i * \text{Res}; \quad z = z_0 + z_i * \text{Res} \quad (2-1)$$

体素的分辨率高低会极大地影响到整个问题的计算规模。体素越小，待监控目标体素化后的数量越大，监控摄像候选点也越多，也就导致可见性测试的计算越加复杂，相应地导致了内存使用率和计算时间的增加，但这样也可以得到更加精确的解，选择合适的体素分辨率需要在计算复杂度和解的精确度上做出权衡。

2.3 生成监控摄像候选点

CityGML 模型文件中的 Envelope 字段可以表示城市的 AABB（轴对齐最小包围盒）。在 AABB 的 XOY 平面上随机生成点，并用从 CityGML 模型文件中提取的道路网基底对其进行裁剪，就可以得出二维平面上的监控摄像候选点位置 (X, Y) 。给候选点赋予一定的高度值 Z ，再加上初始水平角 ϕ 以及初始俯仰角 Θ ，就生成了监控摄像候选点 (X, Y, Z, ϕ, Θ) 。初始化生成的监控摄像候选点随机性较大，在之后的局部搜索算法中，还需对监控摄像候选点的参数进行调整。

点的随机生成算法会对监控摄像候选点的质量造成很大的影响。如图（图 2-5）所示，普通的随机算法和网格随机抖动（将整个区域划分为网格，每个网格中随机生成一个点）生成的点会存在相邻点过近的现象，使得监控摄像候选点在整个研究区域的分布不均匀。更糟糕的是，这样还会使最后生成的布点方案的随机性过大，在研究区域和算法参数相同的情况下，会产生监控目标覆盖率差异非常大的监控摄像布点方案，算法的收敛性不佳。

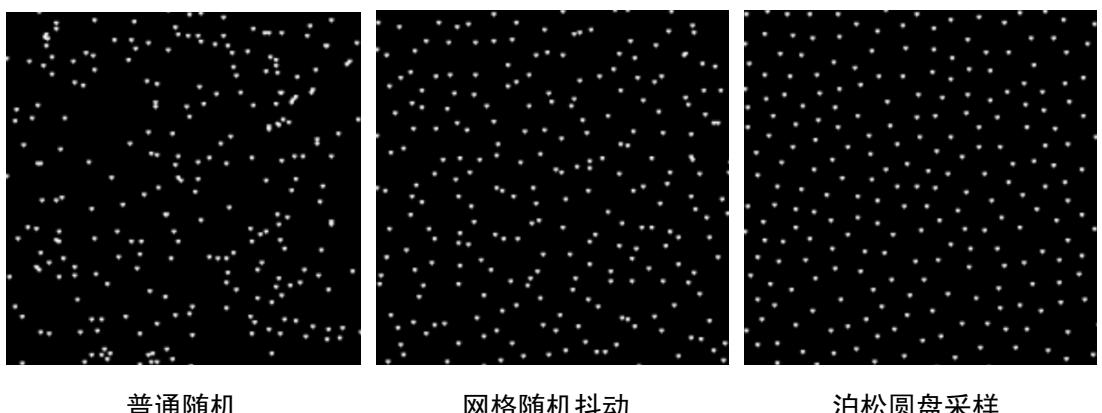


图 2-5 随机算法生成点效果对比

因此，本方法采用泊松圆盘采样的方法来生成随机点^[22]，在整个研究区域内可以生成较为均匀的采样点。泊松圆盘采样是一种图形学中用于反走样（anti-aliasing）的超采样（super sampling）方法，在研究区域中随机地生成点，但是要保证任意两点间的最小距离 Dis。使用泊松圆盘采样算法生成点的效果如图（图 2-5）所示，有效地避免了相邻点过近的现象。

常规的泊松圆盘采样方法，每生成一个点，都需要计算一次和所有已生成的点之间的距离，如果和任意一个点的距离小于 d，则丢弃这个点，在最好情况下，也有 $O(n^2)$ 的时间复杂度，最坏情况下，会花费大量的时间，并且计算的复杂性上限无法预估，实用性不高。

算法 3 泊松圆盘分布点生成算法

输入：宽度 Width, 高度 Height, 最小距离 Dis
输出：符合泊松圆盘分布的点集 Points

```

1: function RandomPointAround(p)
2:   radius ← Dis * Random(1, 2), angle ← Random(0, 360)
3:   return Point(p.x + radius * cos(angle), p.y + radius * sin(angle))
4: end function
5: function InNearCell(grid, point)
6:   xi, yi ← grid.getIndex(point)
7:   if  $\exists i \in [x_i - 2, x_i + 2], \exists j \in [y_i - 2, y_i + 2], grid.exist(i, j)$  且  $distance(grid[i, j], point) < Dis$  then
8:     return true
9:   end if
10:  return false
11: end function
12: Points ← {}, cellSize ← Dis/ $\sqrt{2}$ 
13: // 将整个区域 (Width, Height) 按边长 cellSize 划分为格网 grid
14: processList ← {}
15: firstPoint ← Point(random(Width), random(Height))
16: processList.add(firstPoint), Points.add(firstPoint), grid.add(firstPoint)
17: while processList not empty do
18:   point ← processList 中的一个成员
19:   for i = 1 → 30 do
20:     newPoint ← RandomPointAround(point)
21:     if not InNearCell(grid, newPoint) then
22:       processList.add(newPoint), Points.add(newPoint), grid.add(newPoint)
23:     end if
24:   end for
25: end while
26: return Points

```

算法 3 泊松圆盘分布点生成算法

本文使用一种改进的泊松圆盘分布点生成算法，可以在 $O(n)$ 的时间复杂度下点。算法每次从已生成的点中随机取出一个点，在其周围随机生成 30 个候选点

点，判断候选点所在网格的 24 领域格网内是否已经存在点，如不存在，则可以保证该点与所有点之间的距离大于 Dis。

3. 布点方案生成

3.1 监控目标可见性检测

布点方案生成算法的运行过程中，需要频繁检测一个摄像头是否可以观察到一个监控目标点，即监控目标的可见性检测。可见性检测的算法复杂度会极大地影响到布点方案生成算法的运行效率。

可见性检测方法包含两部分：检测监控目标是否位于摄像头的视域（Field of View，简称 FoV）中的 FoV 检测、检测摄像头到监控目标的视线是否被遮挡的视线遮挡测试。

3.1.1 FoV检测

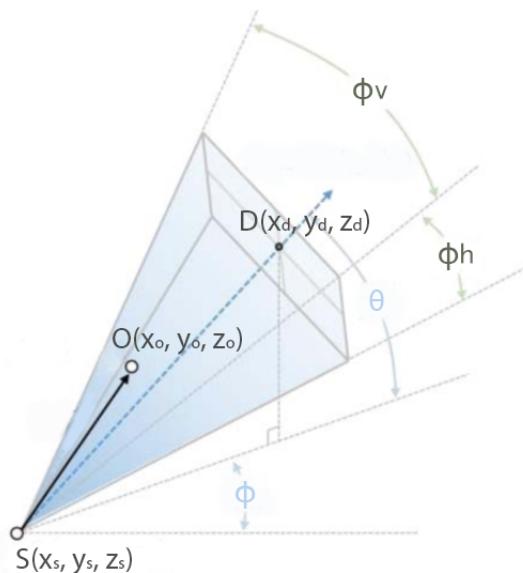


图 3-1 FoV 覆盖区域

摄像头的 FoV (Field of View) 是指摄像头可以观察到的整个可视区域，也可以称为视锥体 (Viewing Frustum)，如图 (图 3-1) 所示。在图形学问题中，摄像机的视锥体还需要用近平面 (Near Plane) 对视锥体进行裁剪，与视点之间距离小于近平面距离的目标认为不可见。但在本问题模型中，由于视点与近平面的距离远小于体素的分辨率，因此可以不考虑近平面对 FoV 检测的影响。可视区域 FoV 取决于摄像头的空间位置 (x_s, y_s, z_s)、水平角 ϕ 、俯仰角 Θ 、相机垂直方向可视角度 ϕ_v 、相机水平方向可视角度 ϕ_h 以及摄像机最远可视距离 d (即摄像

机与视锥体远平面之间的距离，为了降低计算的复杂度，认为摄像机为不可变焦的固定焦距摄像机，即最远可视距离 d 固定)。

当摄像头 $S(x_s, y_s, z_s)$ 、监控目标点 $O(x_o, y_o, z_o)$ 与视锥体远平面中心 $D(x_d, y_d, z_d)$ 之间满足公式 3-2、3-3、3-4 时，则认为目标点 O 位于摄像头 S 的 FoV 内，其中公式 3-4 通过限定视线 SO 在向量 SD 上的投影长度来保证监控目标 O 不会超出摄像机的最远可视距离 d ：

$$dx = x_o - x_s \quad dy = y_o - y_s \quad dz = z_o - z_s \quad (3-1)$$

$$\phi - \phi_h/2 \leq \arctan\left(\frac{dy}{dx}\right) \leq \phi + \phi_h/2 \quad (3-2)$$

$$\theta - \phi_v/2 \leq \arctan\left(\frac{dz}{Dis(S, O)}\right) \leq \theta + \phi_v/2 \quad (3-3)$$

$$0 \leq dx * \cos\theta * \cos\phi - dy * \cos\theta * \sin\phi + dz * \sin\theta \leq d \quad (3-4)$$

3.1.2 视线遮挡检测

除了测试监控目标点 O 是否位于摄像头 S 的 FoV 内，还需判断摄像头到监控目标点的视线是否被场景中的其他物体所遮挡。

对于矢量面片模型来说，判断视线是否被遮挡需要遍历整个场景的三角面片，依次测试视线与三角面片是否相交，每次相交测试都需要使用若干次浮点数的乘除法，当场景所含的三角面片数量过大时，计算量很大。为提高相交测试的效率，Moller 等人提出了“Möller–Trumbore 相交测试算法”^[18]可以加速单次相交测试的速度；也有学者使用 k-d 树（k-dimensional tree）构建分割整个场景^[24]，视线无间复杂度^[25]。

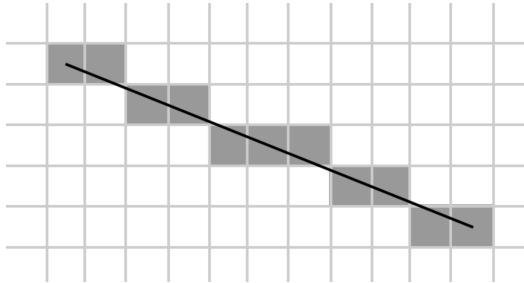


图 3-2 BresenHam 算法二维效果图

对于本方法基于的体素模型来说，遮挡测试有更加高效的实现方法。BresenHam 画线算法是用来快速计算两点所决定直线的栅格化算法，可以得出一条线段在网格中最接近的一系列点，二维下的效果如图所示(图 3-2)。BresenHam

算法的高效实现只需用到整数的加减法，运行效率较高。本方法使用的基于 BresenHam 画线算法的快速遮挡测试算法包含以下几步：将测试目标点 O 与摄像头 S 间的视线使用 BresenHam 算法离散化为网格点（网格的分辨率与体素模型的分辨率相同），判断网格点所对应的体素点集与城市场景的体素点集是否存在交集，如不存在，则认为视线 SO 没有被遮挡。

3.2 调整摄像机参数的局部搜索算法

通过随机算法初始化生成的监控摄像候选点质量具有较高的随机性，而且在初始化时没法给出合理的候选点的初始水平角 ϕ 以及初始俯仰角 Θ ，只有暂且将他们设为同一定值，很显然这是不合理的，候选点的水平角 ϕ 与俯仰角 Θ 应取决于候选点的空间位置 (X, Y, Z) 以及周边待监控目标点的分布。这就导致了初始化生成的监控摄像候选点的监控目标覆盖率较低，如果直接使用贪心算法中的权重计算公式来计算每个候选点的得分，会发现大多数候选点的得分都很低甚至为零，此时只能通过随机的方式挑选最优解，贪心算法就退化为普通的随机算法，效果不佳。

可以通过局部搜索算法调整监控摄像候选点参数，来提高总体的待监控目标覆盖率，调整摄像机参数 (X, Y, Z, ϕ , Θ) 的局部搜索算法如算法 4 所示。

算法 4 调整摄像机参数的局部搜索算法

输入：摄像机参数 Cameras, 待监控目标 Targets, 研究区域 Voxels
输出：调整后的摄像机参数 CamerasAdj

```

1: function EvalCoverage(cameras)
2:   coverCount ← 0
3:   for each target ∈ Targets do
4:     if ∃camera ∈ cameras, CanMonitor(camera, target) then
5:       coverCount = coverCount + 1
6:     end if
7:   end for
8:   return coverCount/Targets.size()
9: end function

10: CamerasAdj ← Cameras
11: coverage ← EvalCoverage(CamerasAdj)
12: for each camera ∈ Cameras do
13:   cameraAdj ← camera
14:   tempCamerasAdj ← CamerasAdj
15:   for each  $dx \in DX, dy \in DY, dz \in DZ, d\phi \in D\phi, d\theta \in D\theta$  do
16:     cameraAdj = camera + ( $dx, dy, dz, d\phi, d\theta$ )
17:     tempCamerasAdj ← CamerasAdj.replace(camera, cameraAdj)
18:     if EvalCoverage(tempCamerasAdj) > coverage then
19:       CamerasAdj ← tempCamerasAdj
20:       coverage ← EvalCoverage(CamerasAdj)
21:     end if
22:   end for
23: end for
24: return CamerasAdj

```

算法 4 调整摄像机参数的局部搜索算法

局部搜索算法^[26]是一种用于解决复杂条件下优化问题的启发式算法。局部搜索算法从一个初始化的候选解开始，搜索解的邻域（search place），如果结果更优，则从这个新产生解的邻域开始搜索，直到邻域中不存在更优解或者算法运行时间超过某个阈值。

调整摄像机参数的局部搜索算法遍历输入的监控摄像机参数 Cameras，根据预先定义的离散常量值数组 ($DX = \{ dx = -2, 0, -2 \}$, $DY = \{ dy = -2, 0, -2 \}$, $DZ = \{ dz = -2, 0, -2 \}$, $D\phi = \{ d_\phi = 0^\circ \sim 330^\circ, \Delta = 30^\circ \}$, $D\Theta = \{ d_\Theta = -30^\circ \sim 30^\circ, \Delta = 10^\circ \}$) 调整监控摄像候选点 camera 的参数 (X, Y, Z, ϕ, Θ)，调整值 cameraAdj 的各项参数计算公式如下所示：

$$x += dx, y += dy, z += dz, \phi += d_\phi, \Theta += d_\Theta \quad (3-5)$$

对于生成的每一个调整值，将原值 camera 替换为调整值 cameraAdj，重新计算整体监控摄像候选点 tempCamerasAdj 的待监控目标覆盖率 EvalCoverage，如果比调整前好，则替换原候选点 camera 为调整值 cameraAdj。

为减少计算的复杂度，在局部搜索候选点的最优水平角 ϕ 和最优俯仰角 Θ 的过程中，不必完整搜索整个 360 度的连续角度空间，可以认为水平角 ϕ 和俯仰角 Θ 为若干个离散的可能值。在本方法中，将水平角 ϕ 离散为从 0 度到 330 度的间隔为 30 度的 12 个离散值，将俯仰角 Θ 离散为从 -30 度到 30 度的间隔为 10 度的 7 个离散值（实际应用中，俯仰角过大或过小也不利于监控摄像头的安装）。

3.3 改进的贪心算法

贪心算法遵循启发式的思想，在算法迭代的每个阶段做出当前局部最优的选择，以期望找到全局的最优解。贪心算法的核心是如何做出当前的最优选择，该选择只取决于之前做出的选择，也不会修改之前做过的选择，可以在较短的时间内收敛到最终解，保证得出最终的解决方案。

在本研究中，所有的待监控目标点都被离散化为若干个体素来表示，由于体素的尺寸远小于待监控目标的原始尺寸，因此无需考虑待监控目标的全角度覆盖问题，待监控目标点只需被至少一个监控摄像头所覆盖即可。

在监控摄像布点问题中，常见的贪心策略是选择那些覆盖的待监控目标最多的监控摄像候选点，但这样有可能会导致选择的监控摄像候选点间的覆盖区域存

在过多的重合，也会导致一些候选点很难被覆盖到。本文使用改进的贪心算法能有效地提高待监控目标覆盖率，算法的详细逻辑如算法 5 所示。

算法 5 选择布点方案的改进贪心算法

输入：算法 4 调整后的摄像机点位 Cameras, 待监控目标 Targets, 研究区域 Voxels, 摄像机布点方案点数 Count

输出：布点方案 CamerasChosen

```

1: CamerasChosen  $\leftarrow \{\}$ 
2: repeat
3:   bestCamera  $\leftarrow \text{null}$ , bestScore  $\leftarrow 0$ 
4:   for each camera  $\in$  Cameras do
5:     if CameraScore(camera)  $>$  bestScore then
6:       bestCamera  $\leftarrow$  camera, bestScore  $\leftarrow$  CameraScore(camera)
7:     end if
8:   end for
9:   if bestCamera 为空 then
10:    break
11:   else
12:     CamerasChosen.add(bestCamera)
13:     Count = Count - 1
14:     Cameras.pop(bestCamera)
15:   end if
16: until Count  $> 0$ 
17: return CamerasChosen

```

算法 5 生成布点方案的改进贪心算法

改进的贪心策略优先选取的监控摄像机可以覆盖到更多的其他监控摄像机所不能覆盖到的待监控目标点。如果摄像机覆盖到了一些只有它才能覆盖到的待监控目标，那么就算其总共覆盖的待监控目标较少，也有可能具有很高的权重。算法 5 中的权重计算函数 *CameraScore* 可以计算得出对应摄像机 j_0 的权重 *score*，计算公式如下所示（其中 *m* 代表待监控目标体素个数、*n* 代表剩余的监控摄像点个数、*view* 代表监控目标体素与监控摄像间的可见性矩阵）：

$$score_{j_0} = \sum_{i=1}^m (view[i][j_0] * (n - \sum_{j=1}^n view[i][j])) / n \quad (3-6)$$

4. 实验结果与讨论

4.1 实验环境

实验代码使用 C++语言编写，遵循 C++11 规范，采用 GNU make（make 是一个工具程序，读取“makefile”文件中记录的源代码之间的构建依赖关系，自动化构建程序）作为构建工具，使用 GCC（GNU Compiler Collection，GNU 编译器套装）作为编译环境，编译时统一开启 O3 级别的优化选项，提高编译后代码的执行速度。

程序的运行环境为：CPU Intel Core i7 4770HQ，内存 16GB DDR3 1600MHz，操作系统 macOS High Sierra 10.13.1。

4.2 实验流程与结果

使用 Random 3DCity 生成包含 50 幢建筑物和相应道路网的 CityGML 城市模型；以 0.5m 的空间分辨率将整个研究区域体素化；提取建筑物入口作为待监控目标区域并体素化；使用泊松圆盘采样（生成点之间的最小距离为 10m，即相邻监控摄像机之间的最小距离为 10m）初始化生成且位于道路上方的监控摄像候选点共 61 个。其中，监控摄像头的垂直方向可视角度为 60 度，水平方向可视角度为 80 度，最远可视距离为 22.5 米。

初始化生成的监控摄像候选点的监控目标覆盖率为 5.88%；使用局部搜索算法调整监控摄像候选点参数，调整后的监控目标覆盖率为 75.59%；使用改进的贪心算法从监控摄像候选点中最终生成包含 20 个监控摄像头的监控摄像布点方案，最终生成的布点方案的监控目标覆盖率为 58.11%，数据预处理与算法运行共耗时 55.48s。

布点方案效果如图（图 4-1）所示，其中蓝色的区域为待监控目标建筑物入口，深绿色方格为布点方案中的监控摄像头，淡绿色区域表示对应监控摄像头的可视区域。在图 4-1 中，未表现出摄像头可视区域被建筑物遮挡后的效果，但布点方案的计算中已考虑了遮挡问题。

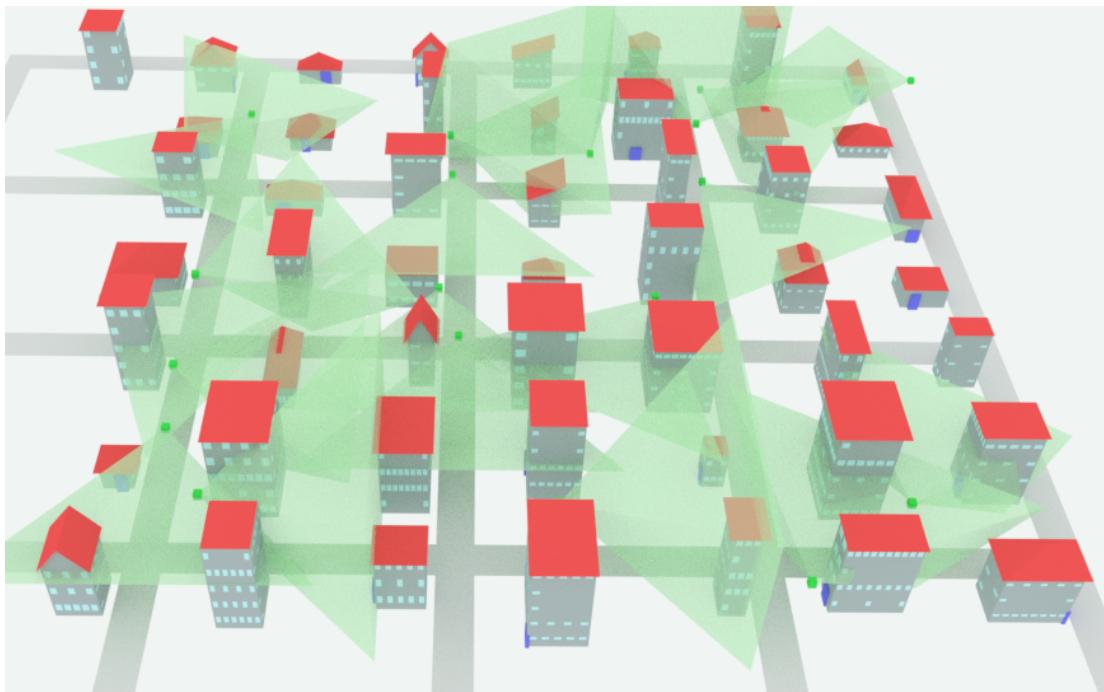


图 4-1 监控摄像布点方案效果图

最终的布点方案选择了 35% 的监控摄像候选点，最终的布点效果是使用所有监控摄像候选点时布点效果的 77.3%。实验结果证明，本方法在较短的运行时间内可以取得较好的布点效果。

4.3 监控摄像布点效果的影响因素

为进一步验证方法的适用性，并探讨影响监控摄像布点效果的各种因素，分别改变算法的各项输入或预设参数（布点方案的监控摄像机数目、体素模型的空间分辨率、相邻监控摄像点间的最小距离），通过布点方案的监控目标覆盖率和算法整体运行时间来研究监控摄像布点效果。

4.3.1 摄像机数目

使用 Random 3DCity 分别生成包含 50 幢、75 幢、100 幢、125 幢的 CityGML 城市场景，每种城市场景设置布点方案包含的摄像机数目分别为 10 个、20 个、40 个，共 12 种测试参数。每种测试参数下，算法运行 10 次，最终生成的布点方案的监控目标覆盖率如图（图 4-2）所示。每张图共有 30 个数据点，分别表示每种监控摄像点数目下算法运行 10 次的结果。

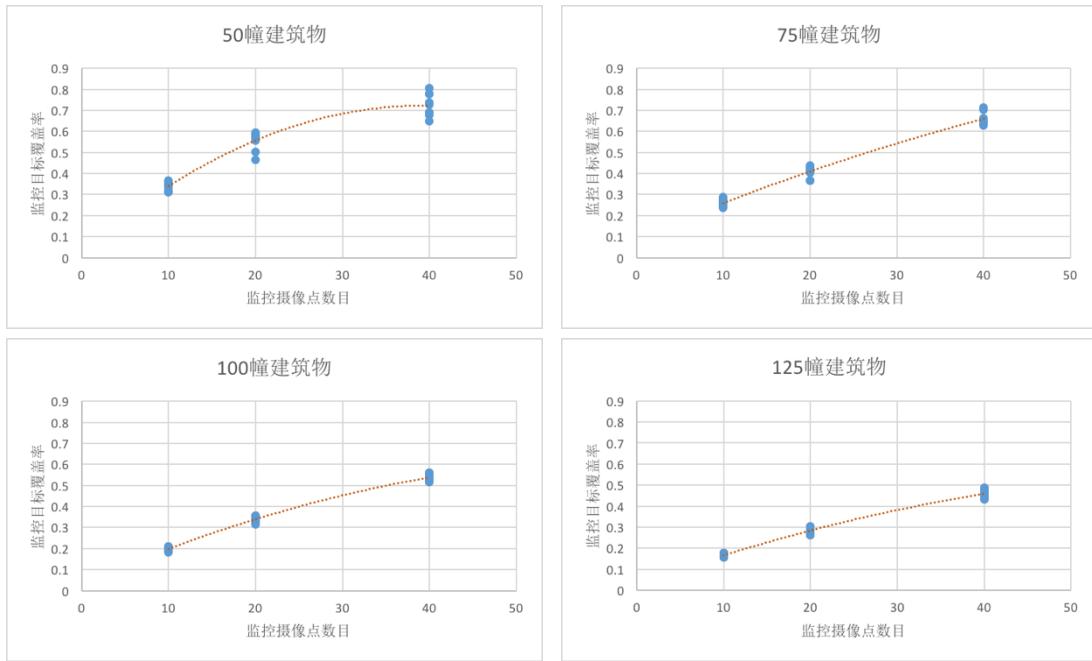


图 4-2 监控摄像点数目-监控目标覆盖率散点图

实验结果表明：在场景建筑物数目一定时，监控目标数目一定，布点方案所含的监控摄像点数目越多，监控目标覆盖率越高；在布点方案所含的监控摄像点数目一定时，场景建筑物数目越多，监控目标也就越多，监控目标覆盖率越低。

对于 50 幢建筑物的场景来说，当布点方案所含监控摄像点数目接近 40 时，可以看出监控目标覆盖率的上升趋势明显变缓，监控目标覆盖率达到一个阈值。这是由于这种情况下生成的监控摄像候选点，经过局部搜索算法调整后，所有候选点的监控目标覆盖率也最多只有 80% 左右，使用贪心从其中挑选出一定数量的点，这些点的监控目标覆盖率也不可能超过这个阈值。

因此，并不是所有情况下，通过增加布点方案所含的监控摄像点数，都可以提高监控目标覆盖率，在到达一定阈值后，监控目标覆盖率还受到其他因素的影响。

4.3.2 体素分辨率

在 50 幢建筑物的城市场景中，分别以 0.25m、0.5m、1.0m 作为体素空间分辨率对城市场景进行体素化，生成含 20 个监控摄像头的监控摄像布点方案。每种情况下，算法运行 10 次。布点方案的监控目标覆盖率和算法运行时间如下图（图 4-3）所示。

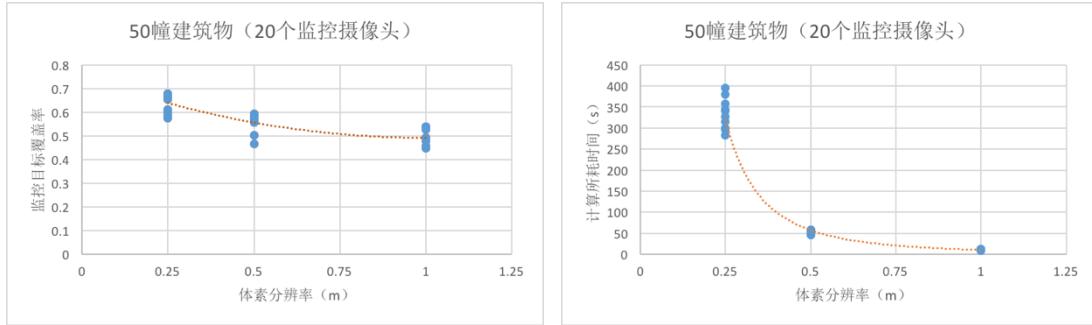


图 4-3 不同体素分辨率下的布点效果及运行效率

实验结果表明：将体素分辨率提高一倍，可以初始化生成更多的监控摄像候选点，产生更高质量的监控摄像布点方案，监控目标覆盖率可以提高 20%左右，但算法的运行时间是之前的 7 倍左右。通过使用更高空间分辨率的体素模型，可以生成更高质量的监控摄像布点方案，但算法的运行时间也会以指数级的速度迅速增长。

使用更高的体素空间分辨率，可以有效提高监控目标覆盖率，但如果使用过高的体素空间分辨率，会导致算法运行时间暴涨，同时也会导致算法运行时所占内存空间的增加。因此，选择合适的空间分辨率需要在监控摄像布点效果和算法运行效率上做出权衡。

4.3.3 生成候选点的随机算法

在 50 幢建筑物的城市场景中，分别使用普通随机算法、网格随机抖动算法、泊松圆盘采样算法初始化生成监控摄像候选点（约为 200 个），挑选出含 20 个监控摄像头的监控摄像布点方案。每种情况下，算法运行 10 次。布点方案的监控目标覆盖率和算法运行时间如下图（图 4-4）所示。

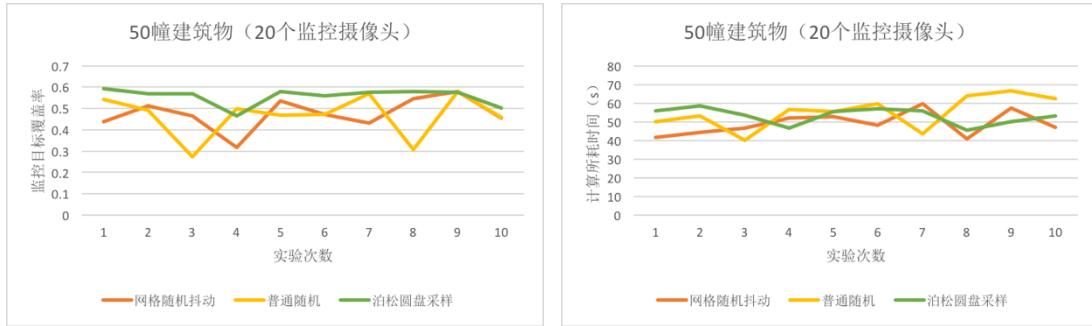


图 4-4 不同类型随机算法对布点效果的影响

实验结果表明：在生成相同数量的监控摄像候选点的情况下，普通随机算法、网格随机抖动算法和泊松圆盘采样算法的运行效率大致相同，但前两者会造成最

终结果监控目标覆盖率的不稳定。其中网格随机抖动算法下的监控目标覆盖率方差为 0.0056，普通随机算法下的监控目标覆盖率方差为 0.0103，泊松圆盘采样算法下的监控目标覆盖率方差为 0.0016。从算法结果的收敛性角度来看，泊松圆盘采样算法是一个较为合适的监控摄像候选点生成随机算法。

4.3.4 摄像机间最小距离

在 50 幢建筑物的城市场景中，分别以 5m、10m、15m 作为泊松分布采样点间的最小距离（即监控摄像头间的最小距离）初始化生成监控摄像候选点，挑选出含 20 个监控摄像头的监控摄像布点方案。每种情况下，算法运行 10 次。布点方案的监控目标覆盖率和算法运行时间如下图（图 4-5）所示。

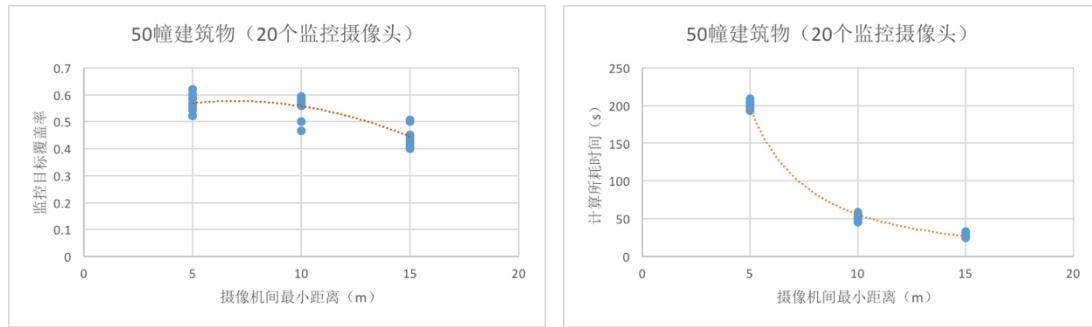


图 4-5 不同摄像间最小距离下的布点效果及运行效率

实验结果表明：缩小泊松分布采样点间的最小距离，可以初始化生成更多的监控摄像候选点，在一定程度上可以提高最终布点方案的监控目标覆盖率，但算法的运行时间也会迅速的增长。当摄像机间的最小距离到达一定的阈值时，再继续缩小最小距离，基本不能提高监控目标覆盖率。

限定监控摄像间最小距离是为了防止最终布点方案中监控摄像点过于集中，但改进的贪心算法可以有效避免这种现象的发生。在使用改进的贪心算法计算各个监控摄像候选点分数的过程中，如果一个候选点已经被挑选进入最终的布点方案中，此时就不再考虑该候选点所覆盖的待监控目标（监控目标只需被一个监控摄像覆盖即可），该候选点附近候选点所覆盖的监控目标就会减少，分数就会降低，这也就有效避免了布点方案中监控摄像点的过于集中。

5. 结论与展望

本文提出了一种基于体素模型的监控摄像自动化布点算法：使用泊松圆盘采样算法初始化生成监控摄像候选点，使用局部搜索算法对候选点参数进行调整，最后使用改进的贪心算法在监控摄像点有限的情况下挑选生成监控摄像布点方案。实验结果证明，算法可以在较短的运行时间内取得较好的布点效果。该算法的布点效果主要受体素分辨率、摄像机数目、摄像机间最小距离的影响。其中体素分辨率的影响较大，摄像机间最小距离的影响较小。来算法运行提高体素分辨率可以有效地提高监控摄像布点效果，但与此同时也会带时间的迅速增长，合适的体素分辨率需要在监控摄像布点效果和算法运行效率上做出权衡。

本文提出的算法也有一些局限性，例如：摄像机类型单一，没有考虑到多种类型摄像机组合下的监控摄像布点方案；不同监控目标的重要性存在差异，监控目标对应的体素应赋予相应的权重信息，在计算监控目标覆盖率时应使覆盖目标体素的权重和最大。

在后续的研究中为提高算法的运行效率，考虑对于不同复杂程度的城市场景，建立不同空间分辨率的体素模型，并使用空间八叉树对整个场景进行划分，减少监控目标与摄像机间可见性检测的计算次数。

参考文献

- [1] 樊亚文, 张重阳, 郑世宝, 等. 城市图像监控系统科学布点布局问题研究. 视频应用与工程, 2011; 1(35) : 107—110.
- [2] 王安, 魏建. 城市化质量与刑事犯罪[J]. 山东大学学报: 哲学社会科学版, 2013, (3): 72-83.
- [3] Costa D G, Guedes L A. A Survey on Multimedia-based Cross-layer Optimization in Visual Sensor Networks[J]. Sensors, 2011, 11(5): 5439-5468.
- [4] 魏浩, 陈华锋, 陈军. 基于路径覆盖的城市监控摄像网络优化部署方法[J]. 计算机工程, 2016, 42(5): 269-274.
- [5] 金伟. 多镜头无缝拼接成像系统的设计与研究[D]. 浙江: 浙江大学, 2006.
- [6] Cole, R. & Sharir, M. Visibility problems for polyedral terrain[J]. Journal of Symbolic Computation, 1989, 7: 11-30.
- [7] Joseph O' Rourke. Art Gallery Theorems and Algorithms. New York, Oxford University Press, 1987.
- [8] P. Bose, Leonidas J. Guibas, A. Lubiw, M. Overmars, D. Souvaine, J. Urrutia. The floodlight problem. In Proc. 5th Canad. Conf. Comput. Geom., 1993: 399 – 404.
- [9] Wang C, Qi F, Shi G M. Nodes placement for optimizing coverage of visual sensor networks[C]. Springer Berlin Heidelberg, 2009, 5879: 1144—1149.
- [10] B. Debaque, R. Jedidi, and D. Prevost. Optimal video camera network deployment to support security monitoring[C]. Proc. Inf. Fusion, 2009: 1730 - 1736.
- [11] K. Chakrabarty, S.S. Iyengar, H. Qi, E. Cho. Grid coverage for surveillance and target location in distributed sensor networks[C]. IEEE Transactions on Computers 51 (2002): 1448 – 1453.
- [12] C. Kit Yee. Angle coverage in wireless sensor networks[D]. Electrical Engineering Department, University of HongKong, 2007.
- [13] K.-Y. Chow, K.-S. Lui, E. Lam. Wireless sensor networks scheduling for full angle coverage[J]. Multidimensional Systems and Signal Processing 20 (2009): 101 – 119.

- [14] Yaagoubi, RedaEl Yarmani, MabroukKamel. HybVOR: A Voronoi-Based 3D GIS approach for camera surveillance network placement[C]. ISPRS International Journal of GEO-Information, JUN 2015: p754-p782.
- [15] R. Al-Hmouz and S. Challa. Optimal placement for opportunistic cameras using genetic algorithm[C]. Proc. Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. Conf, 2005: 337 – 341.
- [16] Y. Morsly, N. Aouf, M. S. Djouadi and M. Richardson. Particle swarm optimization inspired probability algorithm for optimal camera network placement[C]. IEEE Sensors J., vol. 12, May 2012: 1402 – 1412.
- [17] A. Gupta, K. A. Pati, V. K. Subramanian. A NSGA-II based approach for camera placement problem in large scale surveillance application. Proc. 4th Int. Conf. Intell. Adv. Syst. (ICIAS), 2012: 347 – 352.
- [18] Biljecki, Filip and Ledoux, Hugo and Stoter, Jantien. Generation of multi-LOD 3D city models in CityGML with the procedural modelling engine Random3Dcity[J]. ISPRS, 2016, IV-4/W1: 51-59.
- [19] <http://www.cs.cmu.edu/~quake/triangle.html>.
- [20] Tomas Akenine-Moller. Fast 3D Triangle-Box Overlap Testing[C]. ACM SIGGRAPH 2005 Courses, New York.
- [21] M Teschner, B Heidelberger, M Müller, D Pomeranets, M Gross. Optimized spatial hashing for collision detection of deformable objects[J]. Proceedings of Vision, Modeling, Visualization VMV' 03: 47-54.
- [22] DUNBAR, D., AND HUMPHREYS. A spatial data structure for fast poisson-disk sample generation. ACM Trans. Graph. 2006, 25(3): 503 – 508.
- [23] A. T. Murray, K. Kim, J. W. Davis, R. Machiraju, R. Parent. Coverage optimization to support security monitoring[J]. Computer Environment Urban System, 2007, 31: 133 – 147.
- [24] Havran V, Bittner J. On improving k-d trees for ray shooting. Proceedings of the WSCG, 2002: 209 – 216.

- [25] Brown RA. Building a balanced k-d tree in $O(kn \log n)$ time. Journal of Computer Graphics Techniques, 2015, 4 (1): 50 - 68.
- [26] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala , Vinayaka Pandit. Local Search Heuristics for k-Median and Facility Location Problems. Siam Journal of Computing, 2004, 33(3).

本科期间相关的科研成果

一、科研项目

大学生创新训练计划项目国家级立项（G201710284013）基于开放数据的南京市住宅价格时空演变研究

二、所获奖项

第一届江苏省高校测绘地理信息创新创业大赛 创新开发组 特等奖

第二届江苏省高校测绘地理信息创新创业大赛 创业计划组 一等奖

致谢

时光如白驹过隙，转眼大学四年学习生活即将画上一个句号，回首大学四年，是迄今为止人生中成长最快的一段时期。我很幸运来到了南京大学，她让我在其中渡过了如此丰富多彩的四年，她给了我一个广阔的平台，她让我结识了许许多多优秀的人。

首先我要感谢我的导师余江峰副教授，大二开始有幸“地理数据库”这门课上结识了余老师。在此之后，我不仅在学术研究方面得到了余老师的帮助，余老师认真的做事态度也深深地影响了我。在之后参加的 ESRI 开发大赛、江苏省高校测绘地理信息创新创业大赛等竞赛中我也得到了余老师和虚拟地理环境实验室的各位师兄师姐的大力帮助，竞赛的获奖离不开老师和师兄师姐们的支持。参与实验室的项目，也给我在编程方面大量的锻炼机会，没有这些编程方面的训练，我也无法完成这个毕业论文。

我还要感谢实验室的各位师兄师姐，实验室每周的文献汇报增长了我的研究视野，也给了我毕业论文的灵感。感谢李鑫驰和张利维两位师兄，感谢他们和我在论文成型过程中的一次次讨论，给了我许多宝贵的意见。

廖祥森

2018 年 5 月

