面向平板显示设计的悬浮随机行走电容求解算法

（申请清华大学工学硕士学位论文）

培 养 单 位 ： 计算机科学与技术系

学 科 ： 计算机科学与技术

研 究 生 ： 徐 哲 钊

指 导 教 师 ： 喻 文 健 副 教 授

二〇一七年五月

Floating Random Walk Algorithm for the Capacitance Calculation Problem in Flat Panel Display Design

Thesis Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Master of Science

in

Computer Science and Technology

by

Xu Zhezhao

Thesis Supervisor: Associate Professor Yu Wenjian

May, 2017

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：_____ 导师签名：_____

日　　期：_____ 日　　期：_____

摘 要

面向平板显示设计的电容求解问题与大规模集成电路中的电容提取问题比较类似，但又存在很多不同之处，将面向集成电路电容提取的悬浮随机行走算法应用到平板显示设计电容求解问题中遇到了很多新的挑战。针对这些挑战，本文作出了如下贡献。

1. 考虑到平板显示结构包括非曼哈顿型导体的情况，首先提出一种用额外空间网格结构管理非曼哈顿导体的改进空间管理技术，它不需考虑非曼哈顿导体间的遮挡关系，实现简单；之后提出分组遮挡的策略改进非曼哈顿导体遮挡关系的判断算法，从而实现可处理非曼哈顿导体的更高效率空间管理。最后，将它们与曼哈顿转移立方体和旋转转移立方体两种技术结合，开发出四种高效率随机行走电容求解算法，它们比未使用空间管理技术的算法均快几十倍。最后通过更多实验比较，也反映出所提出算法相对于快速边界元法在运算时间和内存用量上的巨大优势。
2. 针对实际工艺中存在的悬浮导体，本文实现了一种针对曼哈顿型悬浮导体的快速计算方法，与已有工作相比，其计算速度在同等准确度情况下快了 35%。进一步还将算法扩展到可以处理非曼哈顿型悬浮导体，并提出了一种自动确定积分面距离参数的策略来平衡电容结果准确度和程序运行时间。
3. 针对平板显示设计中多介质工艺多样性的特点，提出了一种统一的多介质预刻画技术，与已有的两种方法相比，这种方法具有预刻画数据量少、精度高、内存消耗低和不依赖特定多介质工艺的优点。
4. 为了提高程序的计算速度，本文基于 MPI 实现了可以在大规模集群上并行的悬浮随机行走电容求解算法，并提出了一种有效的任务分割策略，实验表明其并行加速比与 MPI 进程数近似呈线形增长关系。

基于本文提出的各种算法，我们开发了专门针对平板显示设计的电容求解软件包 FPDCap，并已经在华大九天公司试用。

关键词：悬浮随机行走；电容求解；平板显示设计

Abstract

The capacitance calculation problem in flat panel display design is similar to the capacitance extraction problem in the design of vary large scale integrated circuits. However, there are distinct differences between them, which bring new challenges to the floating random walk. In this thesis, we made the following contributions.

1. For the non-Manhattan conductor structures, this thesis first proposed an approach with an additional grid structure for non-Manhattan conductors. This approach doesn't consider the domination relationship between non-Manhattan conductors and can be easily implemented. Then this thesis proposed a group domination relationship judgment algorithm which improved the efficiency of space management. Combined with rotated transition cube, four fast capacitance calculation algorithms were developed and had huge advantages over fast boundary element method.
2. This thesis implemented an algorithm for handling the Manhattan floating conductors and extended it to be able to handle non-Manhattan floating conductors. At last this thesis proposed a self-adaptive method to determine the distance parameter between integration surface and floating conductor.
3. For the arbitrary dielectric configuration in flat panel display design, this thesis proposed a unified and accurate dielectric pre-characterization method. Compared with two existing methods, it has the advantages of less pre-characterization data, high accuracy, less memory and independent of specific dielectric configuration.
4. To accelerate the calculation speed, this thesis implemented the parallel FRW algorithm on a Cluster Environment with MPI and proposed an effectively task dividing policy. The speedup is approximately propositional to the number of MPI processes.

Based on the algorithms in this thesis, we developed software FPDCap for flat panel display design and now it's on trial in Empyrean software company.

Key words: Floating random walk; Capacitance calculation; Flat panel display design

目 录

主要符号对照表

FPD	平板显示
CAD	计算机辅助设计
VLSI	大规模集成
BEM	边界元方法
FEM	有限元方法
FRW	悬浮随机行走
MC	蒙特卡洛
IS	重要性采样
SS	分层采样
TSV	硅通孔

第1章　引言

1.1　研究背景

平板显示（Flat Panel Display，FPD）技术在日常生活中的应用越来越广泛，如手机、平板电脑等电子产品。近年来，平板显示与触摸屏技术的结合大大加强了各种电子产品的交互性和用户体验。这种触控平板显示设备一般包含显示和触摸传感两部分，这使得其内部结构更加复杂[1, 2]。触控平板根据传感器工作原理，大致分为电容式、电阻式、红外线式和声波式。电阻式触控平板需要触摸笔输入，不精确，而且灵敏度不高，易损坏。目前市场上的多数触控平板都是采用电容式，其具有耐久度高、可靠性好和适用性强等优点[3]。电容式触控平板主要是利用电容触摸传感器（如图 1.1）来工作，为了提高屏幕的显示分辨率以及触控灵敏度和精度，在设计电容式触控平板时需要精确分析、计算静电场电容。因此，相关电容的计算变得越来越重要和频繁。

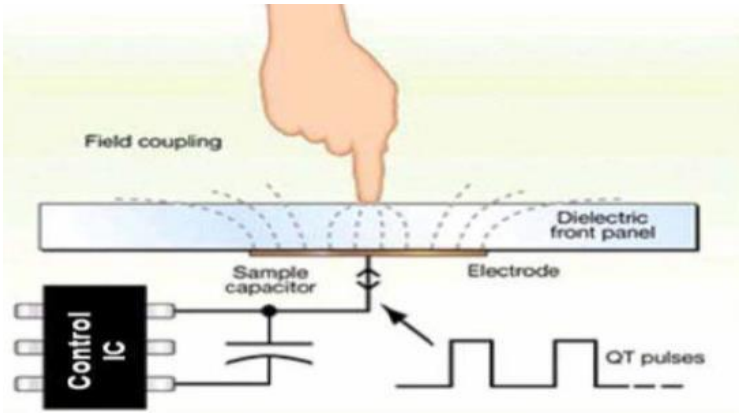


图 1.1 电容式触控平板

高质量的 FPD 设计[4, 5]需要专门的计算机辅助设计（Computer-aided Design，CAD）工具，基本的设计和验证流程包含方案设计、电路模拟、像素设计、版图设计、版图验证和掩膜版设计。版图验证包含设计规则检查、电学规则检查和寄生电阻电容、电压降的计算。为了验证信号时延和获得高显示质量，需要精确地计算 FPD 结构中互连线间的电阻和电容。目前，FPD 呈现了高精度和小型化趋势，其布线区越来越小，而互连线的数量却越来越多，这些互连线的几何结构复杂，使得其电阻和电容提取都变得非常复杂。快速精确的计算互连线之间的电阻、电容已经成为了高质量 FPD 产品成功的关键[9]。

FPD 设计中的电容求解问题包含对 FPD 结构的静电场模拟，这需要精确且有效的场求解器，这一问题与大规模集成（Vary Large Scale Integrated，VLSI）电路设计中的电容提取问题非常类似，然而，这两者之间还是有很多的不同之处（如表 1.1 所示），这些不同之处使得我们不能将应用在 VLSI 电路设计中的场求解器直接应用在 FPD 电容求解问题中。

表 1.1 VLSI 电路电容提取与 FPD 电容求解的不同之处

	VLSI 电路电容提取	**FPD 电容求解**
导体几何形状	大部分是曼哈顿型；长宽比适中	普遍为非曼哈顿型；长宽比很大
介质环境	完全处于芯片内部；介质配置相对固定	设备内介质和设备外的空气介质；任意的介质配置
电容精确度需求	主要需要主导体自电容来进行精确的延迟计算，对耦合电容精确度要求不高	需要精确的耦合电容

针对 VLSI 电路的电容提取问题有很多成熟的技术，主要包括几何模式匹配算法和场求解器方法。几何模式匹配算法需要预计算大量基本模式，然后查找近似模式，通过插值、拟合的方法来得到结果，这种方法虽然效率高，但其准确度低，无法满足目前的需求。场求解器方法可以分为两类，一类是传统的确定性算法[6-14]，以边界元方法（Boundary Element Method，BEM）[6-12]和有限元方法（Finite Difference Method，FEM）[13, 14]为代表，它们通常需要离散化问题空间，然后建立并求解线性方程组，这类算法准确度高，但很难在 VLSI 电路设计中应用，因为其所需内存和时间随着问题规模的增长速度非常快，一般是应用在小型测试结构中，用来验证其他算法的准确度。另一类是基于悬浮随机行走（Floating Random Walk，FRW）的随机型算法[15-31]，它通过蒙特卡洛（Monte Carlo，MC）随机采样的方式来进行电容提取，与确定性算法相比，它具有局部性强、精度可控、并行性好、计算代价随问题规模增长缓慢和使用内存少等优点[21-23]。近年来关于圆柱形硅通孔（Through-silicon via，TSV）结构[24]的研究也证明了基于 FRW 算法的电容求解器比基于 BEM 的电容求解器具有更可靠的准确度。

高效的 FRW 算法主要依赖于一个假设——所有考虑的导体几何形体都是曼哈顿（Manhattan）型的，“曼哈顿型”是表示形体的每个面都平行于 *xoy*、*yoz* 和 *zox* 平面中的一个，这一假设在 VLSI 电路设计领域中成立。对于 FPD 设计中的电容求解问题，上述假设并不成立（见表 1.1），而且几何形体的长宽比可能很大（超

过 1000），这使得基于离散化的 FEM 和 BEM 算法不太适合。FPD 的制造工艺非常多样，不像集成电路中那样固定，一个适用于 FPD 设计的电容求解器需要能够适用于多种多介质工艺环境，因此，事先针对某些多介质工艺进行预刻画的方案并不可行[16, 26]，而且，由于空气介质的存在，还需要考虑更大的相邻介质比（超过 2）。另外，FPD 结构中通常还存在大量工业生产中加入的悬浮导体（如图 1.2），若将悬浮导体当做普通导体进行处理，电容结果会有较大误差。最后，由于触摸传感器是通过检测耦合电容的变化来确定触摸点的位置，所以电容求解器应该能够提供非常精确的耦合电容值。

目前存在一些关于 FPD 电容求解问题的研究[4, 5]，但其主要是基于模式匹配或一种未知的场求解器技术，而且存在精确度不够或计算速度慢的问题，不能满足实际的需求。

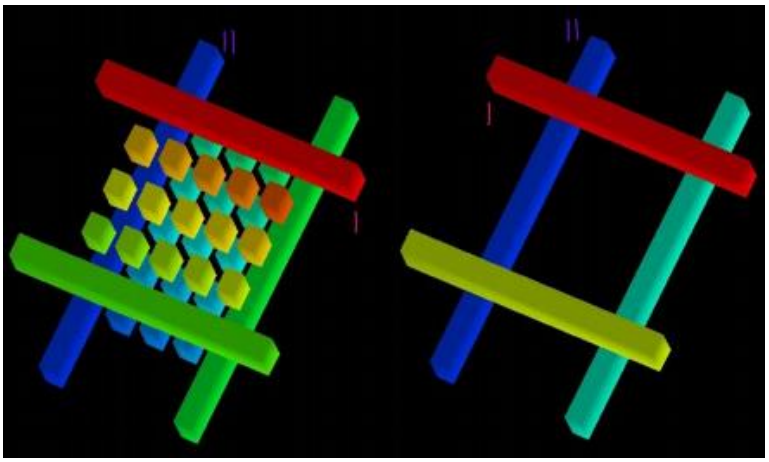


图 1.2 包含悬浮导体（左） 不包含悬浮导体（右）

1.2 本文贡献

总结起来，直接将 VLSI 设计中的电容提取技术应用到 FPD 设计中时，存在如下问题：1) 不能处理包含非曼哈顿型导体的结构；2) 不能处理包含悬浮导体的结构；3) 无法适应 FPD 设计中的多介质工艺环境，需要改进多介质预刻画技术；4) 耦合电容精确度不够；本文主要贡献在于针对这四项挑战，提出了有效的解决办法。

本文首先根据本课题组原有工作[30, 31]补充并完善了处理非曼哈顿型导体[27]和悬浮导体的技术。然后针对 FPD 设计中的多介质工艺环境问题，提出了一种统一的多介质预刻画技术，其具有预刻画数据量少、精度高、内存消耗低和不

依赖特定多介质工艺的优点[28]。

FRW 电容求解算法可以很好的在精确度和效率间进行取舍，更高的耦合电容精确度要求必然导致计算时间的增加，为了解决这一问题，本文将 FRW 电容求解算法由原本的单机并行扩展到了基于 MPI 的大规模集群并行，使得计算时间大大减少[28]。

基于以上这些工作，我们开发了专门针对 FPD 设计的电容求解软件包 FPDCap，并在华大九天软件公司试用，目前反馈效果良好。

第2章 悬浮随机行走电容求解算法

本章 2.1 节首先描述了利用蒙特卡洛方法计算积分的原理，2.2 节介绍了基于蒙特卡洛方法的悬浮随机行走（Floating Random Walk，FRW）电容求解算法，2.3 节介绍了加速 FRW 算法收敛速度的重要性采样和分层采样技术，2.4 节介绍了 FRW 算法需要用到的空间管理技术。

2.1 蒙特卡洛方法计算积分

蒙特卡洛方法[34]，又称计算机随机模拟方法，是一种基于“随机数”的计算方法，它诞生于上个世纪 40 年代美国的"曼哈顿计划"，名字来源于赌城蒙特卡罗。

为了计算积分

$$I = \int_{\Omega} f(x)\,dx, \tag{2-1}$$

首先选择采样概率密度函数 $q(x)$，其满足 $\int_{\Omega} q(x)\,dx = 1$，把积分（2-1）改写为下面的形式

$$I = \int_{\Omega} q(x)\frac{f(x)}{q(x)}dx, \tag{2-2}$$

根据蒙特卡洛方法，上述积分可以利用求和近似得到

$$I \approx I_n = \frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{q(x_i)}, \tag{2-3}$$

其中 x_i 是在被积区间Ω上按照采样概率密度函数 $q(x)$随机得到的第 i 个采样，n 是采样次数，可以看出，采样次数越多，近似结果就越精确。令$A = \int_{\Omega} dx$，若取$q(x) = \frac{1}{A}$，即为均匀采样，此时求和（2-3）式可以简化为

$$I \approx I_n = A \cdot \frac{\sum_{i=1}^{n} f(x_i)}{n}, \tag{2-4}$$

式（2-4）是最常见的利用蒙特卡洛随机采样方法计算积分的公式。

2.2 用于求解电容的悬浮随机行走算法

FRW 算法的基本公式[15, 16]如下：

$$\phi(\boldsymbol{r}) = \oint_S P(\boldsymbol{r}, \boldsymbol{r}^{(1)})\phi(\boldsymbol{r}^{(1)})d\boldsymbol{r}^{(1)}, \tag{2-5}$$

其中$\phi(\boldsymbol{r})$为点 $\boldsymbol{r}$ 处的电势，S 是一个包围点 $\boldsymbol{r}$ 封闭曲面,，$P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$被称为表面格林函数（surface Green's function），它只与曲面 S 的形状和介质有关，与电势无关。对于 $P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$，有

$$\oint_S P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) d\boldsymbol{r}^{(1)} = 1, \tag{2-6}$$

因此，对于固定点 $\boldsymbol{r}$，$P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$可以认为是在曲面 S 随机选择一点 $\boldsymbol{r}^{(1)}$的概率密度函数，因此，可以在曲面 S 上选取大量的随机点，并用这些随机点电势的平均值来估算点 $\boldsymbol{r}$ 处的电势$\phi(\boldsymbol{r})$。若曲面 S 是一个以点 $\boldsymbol{r}$ 为中心的立方体的表面，则表面格林函数 $P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$仅仅与点 $\boldsymbol{r}^{(1)}$的相对位置和介质有关，与立方体的尺寸无关。由于表面格林函数的这个性质，在单介质情况下，它可以被解析的得到，对于分层的多介质情况，也可以用数值方法得到[16]。因此，可以事先计算表面格林函数 $P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$的值，并且利用它在曲面 S 上进行随机采样。

当采样点 $\boldsymbol{r}^{(1)}$的电势$\phi(\boldsymbol{r}^{(1)})$未知时，可以递归的使用公式（2-5），得到如下的嵌套积分公式：

$$\begin{aligned}\phi(\boldsymbol{r}) = &\oint_{S^{(1)}} P^{(1)}(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \oint_{S^{(2)}} P^{(2)}\left(\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(2)}\right) \cdots \\ &\oint_{S^{(k+1)}} P^{(k+1)}(\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k+1)}) \phi(\boldsymbol{r}^{(k+1)}) d\boldsymbol{r}^{(k+1)} \cdots d\boldsymbol{r}^{(2)} d\boldsymbol{r}^{(1)},\end{aligned} \tag{2-7}$$

其中，$S^{(i)}$, (i=1, …, k+1)分别是以点 $\boldsymbol{r}^{(i-1)}$为中心的立方体的表面，$P^{(i)}(\boldsymbol{r}^{(i-1)}, \boldsymbol{r}^{(i)})$, ($i$=1,…, k+1)分别是 $S^{(i)}$的表面格林函数。这一嵌套积分公式可以对应成一个悬浮随机行走过程：对于第 i 次跳转，首先构造一个不包含导体的以点 $\boldsymbol{r}^{(i-1)}$为中心的最大的转移立方体，然后根据概率密度函数 $P^{(i)}(\boldsymbol{r}^{(i-1)}, \boldsymbol{r}^{(i)})$在这个立方体的表面随机的选取一个点 $\boldsymbol{r}^{(i)}$，需要注意的是，概率密度函数 $P^{(i)}(\boldsymbol{r}^{(i-1)}, \boldsymbol{r}^{(i)})$实际是根据单位立方体计算得到的。当第 k 次跳转的点 $\boldsymbol{r}^{(k)}$处的电势已知（如位于导体表面）时，整个悬浮随机行走过程结束，点 $\boldsymbol{r}^{(k)}$处的电势即为这次随机行走的采样值。由于概率密度函数是事先计算好的，所以随机行走过程的主要开销为几何计算。经过很多次的随机行走后，就可以用这些采样值的均值来近似$\phi(\boldsymbol{r})$。

根据导体电容的定义，有

$$\begin{bmatrix} C_{11} & C_{12} & \cdots \\ C_{21} & C_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \end{bmatrix}, \tag{2-8}$$

考虑第 i 行：

$$\sum_j C_{ij} V_j = Q_i, \tag{2-9}$$

若令公式（2-9）中 V_i = 1V，$V_{j(j \neq i)}$ = 0V，可得 $C_i = C_{ii} = Q_i$，即导体 i 的总电容在数值上等于其电荷量。于是，只需求出电荷量 Q_i 就能得到电容 C_i。

为了计算导体电荷量，首先构造一个包裹导体 i（主导体）的高斯面 G_i，根据高斯定理有

$$Q_i = \oint_{G_i} D(\boldsymbol{r}) \cdot \hat{n}(\boldsymbol{r}) d\boldsymbol{r} = \oint_{G_i} \varepsilon(\boldsymbol{r}) \big(-\nabla\phi(\boldsymbol{r})\big) \cdot \hat{n}(\boldsymbol{r}) d\boldsymbol{r}, \qquad (2\text{-}10)$$

其中，$\varepsilon(\boldsymbol{r})$是点 $\boldsymbol{r}$ 处的介质介电常数，$\hat{n}(\boldsymbol{r})$是高斯面 G_i 在点 $\boldsymbol{r}$ 处的法方向。将公式（2-5）代入公式（2-10）可以得到

$$Q_i = \oint_{G_i} \varepsilon(\boldsymbol{r})\, g \oint_{S^{(1)}} w(\boldsymbol{r}, \boldsymbol{r}^{(1)}) P^{(1)}(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \phi(\boldsymbol{r}^{(1)}) d\boldsymbol{r}^{(1)} d\boldsymbol{r}, \qquad (2\text{-}11)$$

其中

$$w\big(\boldsymbol{r}, \boldsymbol{r}^{(1)}\big) = -\frac{\nabla_r P^{(1)}(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})}{g P^{(1)}(\boldsymbol{r}, \boldsymbol{r}^{(1)})}, \qquad (2\text{-}12)$$

称为权值[16]，∇_r是关于 $\boldsymbol{r}$ 的梯度运算符，常数 g 满足$\oint_{G_i} \varepsilon(\boldsymbol{r}) g d\boldsymbol{r} = 1$。公式（2-11）中的第一个积分可以看成是在 G_i 上的随机采样过程，第二个积分可以用基于公式（2-7）的悬浮随机行走过程来计算。因此，Q_i 的计算通过悬浮随机行走过程与导体电势关联了起来。算法 1 和图 2.1 描述了完整的 FRW 电容求解算法流程，称被求解电容的导体为主导体。

算法 1 FRW 电容求解算法

输入： 互连线导体的三维版图，主导体 i

输出： 电容 C_{ij}，(j=1,2,…)s

1: 加载预计算的转移概率表和权值表
2: 构造包裹主导体 i 的高斯面 G
3：C_{ij} := 0，$\forall i$；n_{path} := 0
4：**Repeat**
5：　n_{path} := n_{path} + 1
6：　在 G 上随机选取一点 $r^{(0)}$，并以 $r^{(0)}$为中心生成转移立方体区域 T，根据转移概率表在区域 T 的表面随机选取一点 $r^{(1)}$，根据权值表计算出权值 w
7:　**While** $r^{(1)}$不在导体表面 **do**
8：　以 $r^{(1)}$为中心生成新的转移立方体区域 T，并在 T 的表面根据转移概率表随机选取一点 $r^{(2)}$
9：　$r^{(1)}$:= $r^{(2)}$
10：**End**
11：C_{ij} := C_{ij} + w
12：**Until** C_i^j 精度满足终止条件
13：C_{ij} := C_{ij} / n_{path}，$\forall i$

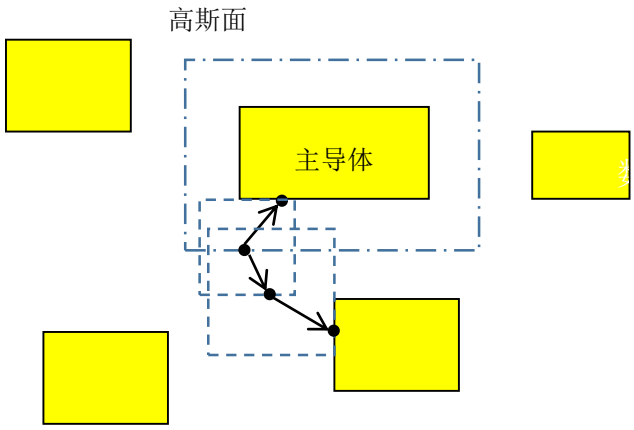


图 2.1 FRW 电容求解算法示例

2.3 重要性采样与分层采样

通过算法 1，可以发现 FRW 算法的主要耗时大致可以表示为

$$T_{total} = N_{walk} \cdot N_{hop} \cdot T_{hop}, \tag{2-13}$$

其中，N_{walk} 是采样次数，N_{hop} 是单次采样的平均跳转次数，T_{hop} 是单次跳转所需的平均时间。本节讨论减小 N_{walk} 的相关技术。

回顾公式（2-4），可以发现 I_n 是 $A \cdot f(x_i)$的平均值，由于 x_i 是采样值，所以 I_n 是一个随机量。函数 $A \cdot f(x)$的方差可以近似用下式表示，

$$Var_n = \frac{A^2 \sum_{i=1}^{n} (f(x_i))^2 - n \cdot {I_n}^2}{n}, \tag{2-14}$$

根据中心极限定理[35]，I_n 满足正态分布，其标准差可以近似表示为

$$err_n \approx \frac{\sqrt{Var(A \cdot f(x))}}{\sqrt{n}} \approx \frac{\sqrt{Var_n}}{\sqrt{n}}, \tag{2-15}$$

其中，err_n 也被称作 I_n 的 1–σ误差，由于 I_n 满足正态分布，1–σ误差意味着该误差区间的置信水平为 68%，且 $3 \cdot err_n$ 误差区间的置信水平为 99.7%。

根据上面的讨论可知，当采样数一定时，减小被积函数 $f(x)$的方差可以减小 err_n，使得估计值 I_n 的精度更高。回顾公式（2-2），一般情况下，蒙特卡洛方法的被积函数为$\frac{f(x)}{q(x)}$，因此需要减小函数$\frac{f(x)}{q(x)}$的方差，函数 $f(x)$是不可变的，因此需要选择合适

的 $q(x)$，$q(x)$与 $f(x)$越接近，被积函数$\frac{f(x)}{q(x)}$的方差就越小。这种根据待计算函数来选取采样概率函数从而减小方差的方法称为重要性采样（Importance Sampling，IS）。

重要性采样方法可以用在 FRW 算法中，首先计算

$$K = \oint_S \left|\nabla_r P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})\right| d\boldsymbol{r}^{(1)}, \tag{2-16}$$

其中，S 和$P(r, r^{(1)})$分别为单位立方体的表面和表面格林函数。因为 $\boldsymbol{r}$ 是立方体的中心点，所以 K 为一个常数。将公式（2-11）改写成如下的形式

$$\begin{aligned} Q_i &= \oint_{G_i} \varepsilon(r) g \oint_S -\frac{\nabla_r P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})}{L \cdot g} \phi\left(r^{(1)}\right) dr^{(1)} dr \\ &= \oint_{G_i} \varepsilon(r) g \oint_S -\frac{K \cdot \nabla_r P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})}{L \cdot g \left|\nabla_r P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})\right|} \cdot \frac{\left|\nabla_r P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})\right|}{K} \phi\left(r^{(1)}\right) dr^{(1)} dr, \end{aligned} \tag{2-17}$$

其中，L 为第一个转移立方体的边长，并且积分区域由原来的 $S^{(1)}$转化成了单位立方体的表面 S。

定义函数

$$q\left(r, r^{(1)}\right) = \frac{\left|\nabla_r P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})\right|}{K}, \tag{2-18}$$

根据式（2-16），$q(r, r^{(1)})$可以作为 S 上的概率密度函数。于是有

$$Q_i = \oint_{G_i} \varepsilon(r) g \oint_S -\frac{K \cdot \nabla_r P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})}{L \cdot g \left|\nabla_r P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \cdot \hat{n}(\boldsymbol{r})\right|} \cdot q\left(r, r^{(1)}\right) \phi\left(r^{(1)}\right) dr^{(1)} dr, \tag{2-19}$$

可以将公式（2-19）看成是一种新的采样策略，其中权值为

$$\tilde{w}\left(r, r^{(1)}\right) = \begin{cases} -\frac{K}{L \cdot g}, & if\ \nabla_r P\left(\boldsymbol{r}, \boldsymbol{r}^{(1)}\right) \cdot \hat{n}(\boldsymbol{r}) > 0 \\ \frac{K}{L \cdot g}, & if\ \nabla_r P\left(\boldsymbol{r}, \boldsymbol{r}^{(1)}\right) \cdot \hat{n}(\boldsymbol{r}) < 0 \end{cases}, \tag{2-20}$$

这种新的采样策略的权值几乎为常数，这使得在采样数相同的情况下方差更小，也就是结果更精确。

除重要性采样外，还有一种叫做分层采样（Stratified Sampling，SS）的方差约减技术，分层采样的核心思想是将被积区间划分成若干个子区间分别进行积分，例如，将公式（2-1）中的被积区间Ω划分成Ω_a和Ω_b两个大小相同的子区间，于是有

$$I = I_a + I_b = \int_{\Omega_a} f(x) dx + \int_{\Omega_b} f(x) dx, \tag{2-21}$$

分别对 I_a、I_b 使用蒙特卡洛方法计算，可以得到

$$I_n' = I_{na}^{(a)} + I_{nb}^{(b)} = \frac{A}{2} \cdot \frac{\sum_{i=1}^{n_a} f(x_{a,i})}{n_a} + \frac{A}{2} \cdot \frac{\sum_{i=1}^{n_b} f(x_{b,i})}{n_b}, \quad (2\text{-}22)$$

其中，n_a 和 n_b 分别为在区间Ω_a 和Ω_b 上的采样数，由于$I_{na}^{(a)}$和$I_{nb}^{(b)}$相互独立，I_n'的方差为$I_{na}^{(a)}$和$I_{nb}^{(b)}$的方差之和，可以证明[35]

$$Var(I_n') < Var(I_n), \quad (2\text{-}23)$$

当 $f(x)$在区间Ω_a 和Ω_b 上采样的平均值不相等时，上式就成立。

根据公式（2-20）可以看到，权值只有两个非零数值，所以可以将立方体表面划分成两部分，一部分是权值为正的区域，另一部分是权值为负的区域。 另外，对于式（2-19）的第一个积分，将高斯面 G_i 分成 6 个朝向不同的方向进行分层采样也可以有效的减小方差。

2.4 空间管理技术

在随机行走的跳转过程中，一个非常重要的步骤是构造一个不包含导体块的最大转移立方体（如图 2.1），为了达到这一目的，需要首先找到距离当前点最近的导体块，当前点到最近导体块的距离就是要求的最大转移立方体的半边长。由于通常会有几百万次跳转过程，所以构造转移区域的过程应当尽可能的快，若每次查询最近导体时都使用遍历所有导体块的方法，则当导体块的数目很大时，效率会非常低，为了解决这个问题，需要引入空间管理技术。空间管理的主要思想是将整个三维空间区域划分为有组织的互不相交的子区域（称作空间单元），通过存储与各个空间单元有关的邻近导体信息，就可以快速计算空间中任一点到最近导体块的距离。

针对只包含曼哈顿型导体块的结构，已经有了很多相关的空间管理技术[16, 17, 25, 22]。主要思想是对于每个空间单元，为其维护一个候选导体列表，使得该空间单元中任意一点的最近邻导体都能在这个候选导体列表中找到，这样的话，查询最近邻导体时不需要遍历所有的导体块，只需要遍历候选导体列表就行，这使得查找速度大大提升。

空间管理包含构造空间数据结构和基于空间数据结构查询最近邻导体两个步骤，前者会引入一些时间和内存开销，但后者可以大大减少 FRW 算法的运行时间。八叉树和空间网格结构是两种应用非常广泛的空间数据结构，[17]还提出了一种网格-八叉树的混合结构。

在构造空间结构过程中，一个主要的步骤的生成空间单元的候选导体列表。我们需要逐个检查导体以确定其是否需要加入到候选导体列表中，在这一过程中，需要考虑导体之间的遮挡关系。

定义 2.1：T 是一个空间单元，B_1、B_2 是两个曼哈顿型导体块。若对于任意点 $\boldsymbol{P}\in$T，且 $\boldsymbol{P}\notin B_1\cup B_2$，$d(\boldsymbol{P}, B_1)\le d(\boldsymbol{P}, B_2)$，则称对于 T，$B_1$ 遮挡 B_2，如图 2.2 所示。

若对于空间单元 T，B_1 遮挡 B_2，且 B_1 已经在 T 的候选导体列表中，则 B_2 不能加入到该列表中。其中，$d(\ ,\ )$表示点到长方体的距离，即以该点为中心，且与长方体接触的曼哈顿型立方体的半边长，当点在长方体内部时，该距离为负值。

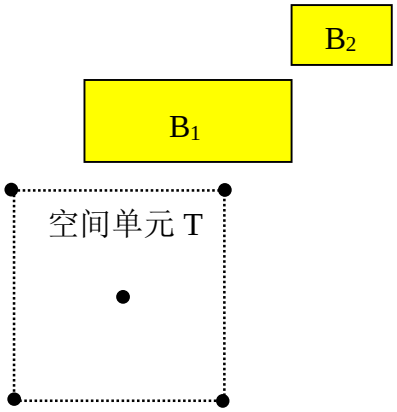


图 2.2 遮挡关系示例，对于空间单元 T，B_1 遮挡 B_2

为了加速候选导体列表的生成，引入门限距离 L(T)的概念，L(T)表示空间单元 T 中任意点到最近导体块的最大值[17]。算法 2 描述了尝试将导体 B 加入到空间单元 T 的候选导体列表的过程。

对于两个曼哈顿型导体来说，它们之间的遮挡关系判断并不困难，但如果其中一个导体变为非曼哈顿型导体，则遮挡关系的判断将变得非常复杂。

算法 2 尝试将新导体加入到空间单元的候选列表

输入：导体 B，空间单元 T，T 的候选导体列表 $list_T$
输出：添加成功返回 ture，否则返回 false

```
1：d := d(B, T); l 为 T 的边长
2：If d ≥ L(T) then return false
3：For b ∈ list_T do
4：   If b 遮挡 B then return false
5：   Elseif B 遮挡 b then
6：     将 b 从 list_T 中移除
7：   End
8：End
9：将 B 加入到 list_T
10：If (d + l) < L(T) then L(T) := d + l
11：return true
```

第3章 非曼哈顿型导体处理

平板显示（Flat Panel Display，FPD）结构中存在大量的倾斜金属导线和一般形状的非长方形导体块，为了设计高质量的 FPD 设备，悬浮随机行走（Floating Random Walk，FRW）算法能够处理一般非曼哈顿几何形体的需求越来越迫切。

关于处理非曼哈顿型导体的 FRW 算法研究非常少，在[21]中，提出了一种使用球形转移区域的方法，但这种方法只能用在单介质结构中。基于[21]的方法，[23]中提出了一种利用距离映射来辅助球形转移区域构造的改进方法，但生成足够精度的距离映射非常耗时，所以也不是一种有效的方法。[24]中提出了一种可以处理圆柱形硅通孔结构的 FRW 算法，然而该算法只考虑了圆柱形非曼哈顿型导体，并不能推广到一般情况。

[30]中提出了一种处理非曼哈顿型导体的 FRW 算法，以此为基础，本章进一步提出了改进算法，包括使用额外的网格结构来管理非曼哈顿型导体和改进的非曼哈顿型导体块遮挡关系判断算法。最后将各种不同的策略整理分类为 6 个算法，并进行了综合实验比较，另外还与快速边界元方法作了对比。

3.1 基本距离计算和高斯面生成

[30]中已经提出了基本的距离计算方法和高斯面生成算法，但为了论文完整性，本节将对这部分内容作一个简要介绍。

非曼哈顿型导体的底面和顶面平行于 *xoy* 坐标平面，侧面垂直于 *xoy* 平面，且其在 *xoy* 平面的投影（或称为顶视图）为一个任意的凸多边形。在 FRW 算法中，需要计算点与导体块或导体块与导体块之间的距离，由于非曼哈顿型导体的缘故，这些距离的计算变得更为复杂。首先作一些关于距离的基本定义。

定义 3.1：二维点 *P* 到凸多边形 A 的对齐距离（aligned-box distance）$dist_a(P, \mathrm{A})$为以 P 为中心点与 A 接触的轴对齐正方形边长的一半。

图 3.1(a)展示了多边形 A 周围的一些典型点位置，图 3.1(b)展示了计算 $dist_a(P, \mathrm{A})$的基本思路：首先找到 A 对于点 *P* 的可见边（若以 *P* 为中心的曼哈顿正方形接触 A 的某条边，则该条边为 A 对于点 *P* 的可见边），方法是对 A 的每条边$\overrightarrow{A_\iota A_{\iota+1}}$，计算$\overrightarrow{A_\iota P}$与$\overrightarrow{A_\iota A_{\iota+1}}$的叉积，若结果为正数，则边$\overrightarrow{A_\iota A_{\iota+1}}$为可见边，且该结果 1/2 即为为三角形 PA_iA_{i+1} 的面积，如图 3.1(b)所示，三角形 PA_iA_{i+1} 可以看成是由三角形 PA_iR、PRA_i、RA_iS 和 RSA_{i+1} 组成，其中，点 *S* 是转移立方体与 A 的接触点，点 *R* 是正方

形边上的中点，这四个三角形都是以转移立方体的半边长作为其底边，且它们的高合起来就是点 A_i 到 A_{i+1} 的 x 轴距离和 y 轴距离之和。所以，$\overrightarrow{A_iP}$与$\overrightarrow{A_iA_{i+1}}$的叉积除以点 A_i 到 A_{i+1} 的 x 轴距离和 y 轴距离之和即为转移立方体的半边长。图 3.1(a)中的点 P_3，其转移立方体与 A 的顶点接触，这种情况下 $dist_a(P, \mathrm{A})$为 P 到 A 的曼哈顿边界盒子（Manhattan bounding box）的距离。于是可以得到定理 3.1。

定理 3.1：假设多边形 A 的顶点为 A_1，A_2，A_3，…，A_n，且为逆时针序排列，其中，A_i 的坐标为（x_i，y_i），i=1,2，…，n。假设点 P 坐标为（x，y），则有

$$dist_a(P,\mathrm{A}) = \max\{\max_{1\le i\le n} \frac{(x-x_i)(y_{i+1}-y_i)-(y-y_i)(x_{i+1}-x_i)}{|x_{i+1}-x_i|+|y_{i+1}-y_i|}, dist_a(P,\mathrm{B_A})\}, \quad (3.1)$$

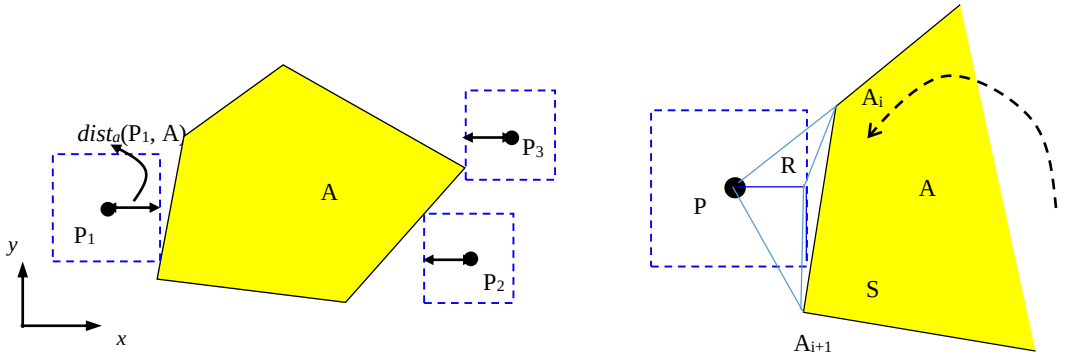
其中 $\mathrm{B_A}$ 是多边形 A 的曼哈顿边界盒子。

点到导体块的竖直距离定义如下。

定义 3.2：点 P（x, y, z）到导体块 A 的竖直距离为

$$dist_v(P,\mathrm{A}) = \max\{z - z_{max}(\mathrm{A}), z_{min}(\mathrm{A}) - z\}, \quad (3.2)$$

其中，z_{min}(A)和 z_{max}(A)分别为导体块 A 的所有顶点 z 坐标的最小值和最大值。



(a) (b)

图 3.1 (a) 多边形 A 与点的二维对齐距离. (b) 计算二维对齐距离方法

下面简要介绍高斯面生成算法。首先考虑主导体只包含一个导体块的情况，图 3.2 展示了一种非曼哈顿型结构的典型顶视图（“A”标记为主导体），从图 3.2(a)中可以看到主导体的边界盒子与其他导体相交了，因此，[24]中为边界盒子生成高斯面的方法在这里不适用了，这里必须考虑主导体的实际几何形状。

对于两个不同的多边形，定义它们之间的对齐距离如下。

定义 3.3: 两个多边形 A 和 B 之间的对齐距离 $dist_a$(A, B)为同时接触 A 和 B 的轴对齐正方形边长的最小值。

基于点到多边形的对齐距离（定义 3.1），可以推导出两个多边形[见图 3.2(b)]之间的对齐距离的公式，

$$dist_a(\mathrm{A},\mathrm{B}) = \min\{\min_{1\le i\le n} dist_a(A_i,\mathrm{B}), \min_{1\le i\le m} dist_a(B_i,\mathrm{A})\}, \quad (3\text{-}3)$$

其中，A_i（i=1,2,…,n）为多边形 A 的顶点，B_i（i=1,2,…,m）为多边形 B 的顶点。两个导体块之间的水平距离定义如下。

定义 3.4：两个导体块 A 和 B 之间的水平距离为

$$dist_v(\mathrm{A},\mathrm{B}) = \max\{z_{min}(\mathrm{A}) - z_{max}(\mathrm{B}), z_{min}(\mathrm{B}) - z_{max}(\mathrm{A})\}, \quad (3\text{-}4)$$

因此，可以定义两个导体块之间的距离。

定义 3.5：两个导体块 A 和 B 之间的距离为

$$dist(\mathrm{A},\mathrm{B}) = \max\{dist_v(\mathrm{A},\mathrm{B}), dist_a(\mathrm{P_A},\mathrm{P_B})\}, \quad (3\text{-}5)$$

其中，$\mathrm{P_A}$ 和 $\mathrm{P_B}$ 分别为多边形 A 和 B 在 *xoy* 平面的投影。

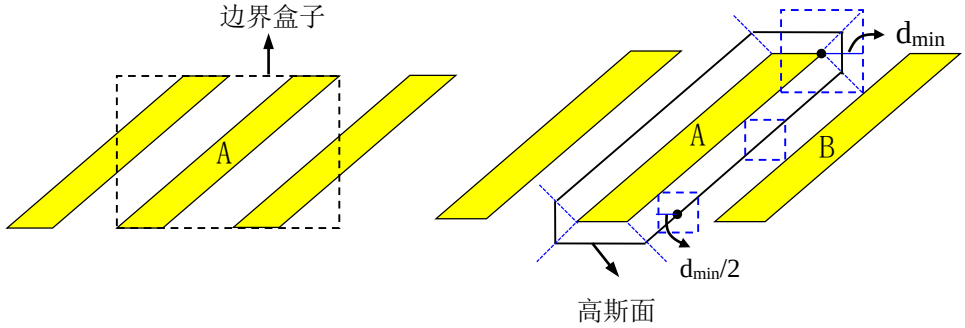


图 3.2 非曼哈顿型导体顶视图 (a) 主导体边界盒子. (b) 有效的高斯面生成算法

生成主导体 A 的高斯面 $\mathrm{G_A}$ 的主要思想为：首先计算 A 到其相邻导体块的最小距离 $d_{\min}$，然后令高斯面 $\mathrm{G_A}$ 到 A 的距离为 $d_{\min}/2$，这样可以保证高斯面不与其他导体相交，且这个包裹主导体的高斯面同样会是一个凸直棱柱，图 3.2(b)展示了高斯面在 *xoy* 平面的投影，主导体 A 的每一条边都向外膨胀一段距离得到一条新的边，使得新的边上的每个点距 A 的对齐距离为 $d_{\min}/2$，然后，将膨胀得到的边连接起来，这样就得到了高斯面在 *xoy* 平面的投影 $\mathrm{G_A}$。若 A 的投影有 n 条边，则 $\mathrm{G_A}$ 边的数目在 n 到 $2n$ 之间。

对于主导体包含多个导体块的情况，可以用[19]中的虚拟高斯面采样技术，这种方法不需要计算各个导体块高斯面的包络面，可以直接对每个导体块的高斯面单独采样，对于非曼哈顿型导体块的顶部和底部，其对应的高斯面为一般二维多边形，可以用拒绝采样的方法在其上进行采样[35]。

3.2 使用曼哈顿转移立方体的算法

事实上，对于非曼哈顿型结构，曼哈顿转移立方体仍然是适用的。在随机行走过程中，需要构造一个不包含导体的最大的曼哈顿转移立方体，于是需要计算

当前点 $\boldsymbol{P}$ 到最近导体块的距离，当导体块为曼哈顿导体时，距离计算很容易，当导体块为非曼哈顿导体 A 时，距离计算公式为：

$$dist_a(\boldsymbol{P},\mathrm{A}) = \max\{dist_v(\boldsymbol{P},\mathrm{A}), dist_a(\boldsymbol{P},\mathrm{P}_A), \tag{3-6}$$

其中，P_A 表示 A 在 *xoy* 平面的投影。

一般情况下，随机行走过程中会涉及到到几百万次的跳转，而每一次的跳转都需要计算点到最近导体块的距离，这个计算过程应该越快越好。使用空间管理技术可以显著提高这个计算过程，但以往的空间管理技术只考虑了曼哈顿型导体，对于非曼哈顿型导体是不适用的，所以需要设计新的空间管理技术来适应非曼哈顿型导体的情况。我们提出了三种方法来解决这个问题。

1) 采样遍历所有非曼哈顿型导体块的方法

若大部分导体都是曼哈顿型，一种简单的办法是首先计算出只考虑曼哈顿型导体作为遮挡物的最大转移立方体，然后逐个遍历非曼哈顿型导体来调整转移立方体的大小。这种方法仅仅利用了已有的考虑曼哈顿型导体的空间管理技术，非常容易实现，在非曼哈顿型导体数目比较少时非常有效，但当导体块的数目很大时，计算速度会非常慢。

2) 添加一种额外的结构来处理非曼哈顿导体

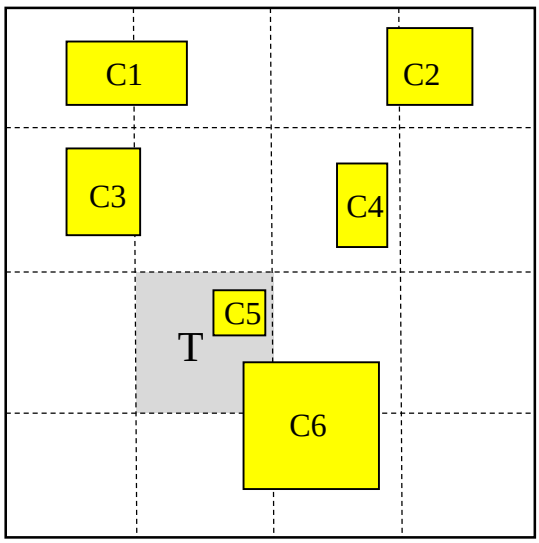


图 3.3 添加额外的空间网格结构来处理非曼哈顿型导体

这种方法主要思想是为非曼哈顿型导体块建立一个额外的空间结构，使得计算最大转移立方体时不需要遍历所有的非曼哈顿型导体。为了避免涉及到非曼哈顿型导体的复杂遮挡关系判断，可以使用一种简单的空间网格结构来处理非曼哈

顿型导体。主要思想是将整个三维空间均匀的划分为等大小的立方体网格单元，对每个网格单元，记录下与自身和相邻空间单元相交的非曼哈顿型导体信息，如图 3.3 所示，空间单元 T 需要记录的导体信息为 C3、C4、C5 和 C6，其中 C3 和 C4 与和 T 相邻的空间单元相交，C5 和 C6 与 T 相交。在随机行走过程中，首先计算出只考虑以曼哈顿型导体为遮挡物的转移立方体大小，然后考虑当前网格单元中记录的非曼哈顿型导体，调整转移立方体的大小使其不与任何导体块相交。利用这种方法，可以避免每次跳转都遍历非曼哈顿型导体，然而，由于没有考虑非曼哈顿型导体块之间的遮挡关系，这种方法并不是最优的，会引入一些额外的内存开销。

3) 修改已有的空间管理技术以使其适应非曼哈顿型导体

一种更为复杂的解决方法是扩展已有的空间管理技术，使其同样适用于非曼哈顿型导体存在的情况。这种方法的难点主要在于两个导体块之间的遮挡关系判断。我们不能直接套用定义 2.1 来实现遮挡关系的判断，因为无法枚举空间单元 T 中的所有点。但对于两个曼哈顿型长方体，遮挡关系的判断只需要检查 T 中很小的一个点集即可。

定理 3.2：假设导体块 A、B 和空间单元 T 都是曼哈顿型形体，则“对于 T，A 遮挡 B”的充分必要条件是：

1) $B \cap T = \varnothing$

2) $(B^* \cap T) \subseteq (A^* \cap T)$，其中 A^* 和 B^*分别为 A 和 B 膨胀距离 $dist(B, T)$后得到的几何形体。

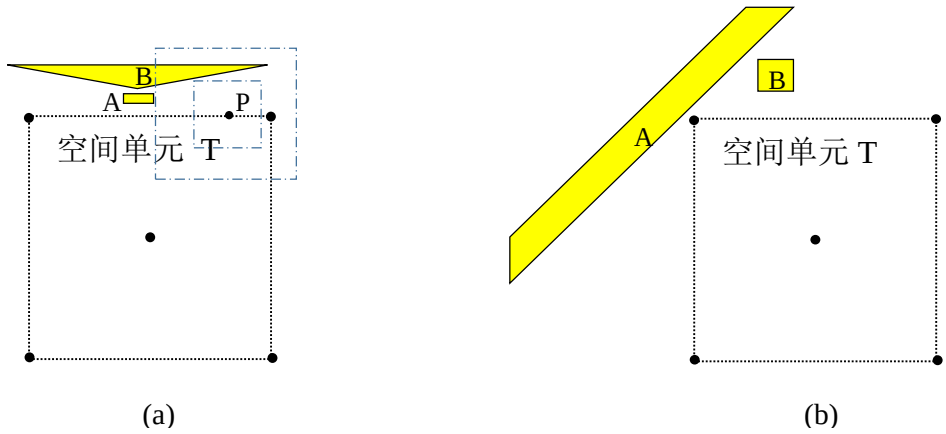
为了解释定理 2 中的条件 2)，考虑属于 $B^* \cap T$ 中的点，由于 B^*是 B 膨胀距离 $d = dist(B, T)$后得到的几何形体，$B^* \cap T$ 实际上是 B^*和 T 表面的一部分。注意到 B^*表面的每一个点到 B 的距离都为 d，这形成了一个等高面。所以$\forall \boldsymbol{P} \in B^* \cap T$，$dist_a(\boldsymbol{P}, B) = d$。$A^*$是 A 膨胀距离 d 得到的，根据条件 2），$\boldsymbol{P} \in A^* \cap T$，所以 $dist_a(\boldsymbol{P}, A) \le d = dist_a(\boldsymbol{P}, B)$。想象将 A 和 B 继续以相同的速度膨胀，这使得它们的等高面逐渐扫过 T 中的剩余点，由于 A 的等高面在 x, y, z 三个方向上的扩张速度都与 B 的等高面相同，所以对于属于 T，但不属于 $B^* \cap T$ 的点 $\boldsymbol{P}$，$dist_a(\boldsymbol{P}, A) \le dist_a(\boldsymbol{P}, B)$仍然成立。这意味着对于 T，A 遮挡 B。因此，定理 3.2 得证。

基于定理 3.2，在判断 A 是否遮挡 B 时，可以只考虑 $B^* \cap T$ 中的点（二维长方形），[16, 17]中提出了基于此的遮挡关系判断算法。

然而，对于非曼哈顿型形体来说，仅仅检查 T 和 B^*交集点是不够的。这是因为形体等高面扩张时扫过的区域与形体的形状有关。若 A 和 B 为任意形状的形体，当 A 距 $B^* \cap T$ 中的点更近时无法保证 A 比 B 距离 T 中其他的点更近。图 3.4（a）

展示了一个这样的例子。类似的，仅仅检查空间单元 T 的顶点对于遮挡关系的判断来说也是不够的[见图 3.4(b)]。

不同于曼哈顿型导体块，对于一般的非曼哈顿型导体块，我们无法找到一种



(a) (b)

图 3.4 对于 T，A 不遮挡 B 的例子 (a) A 距 T∩B*中点更近
(b) A 距 T 的四个顶点更近

有效的方法来判断它们的遮挡关系。但我们很容易发现两个明显的判断遮挡关系的充分条件，[30]中提出了下面这种判断非曼哈顿导体的遮挡关系的方法。

定理 3.3：对于两个导体块 A 和 B，In(A)表示 A 的曼哈顿内切盒子，Ex(B)表示 B 的曼哈顿外切盒子。则

1) 若对于空间单元 T，A 遮挡 Ex(B)，则对于 T，A 遮挡 B

2) 若对于空间单元 T，In(A)遮挡 B，则对于 T，A 遮挡 B

定理 3.3 的一个推论为：若 In(A)遮挡 Ex(B)，则 A 遮挡 B。因此，通过使用判断两个曼哈顿型导体遮挡关系的算法（基于定理 3.2），我们可以部分的判断两个任意形状导体块的遮挡关系。图 3.5 展示了涉及非曼哈顿型导体块的遮挡关系判断算法。

需要注意的是，对于某些非曼哈顿型导体（如长斜线），其曼哈顿内切盒子和外切盒子与其自身的差别非常大，因此，基于定理 3.3 的遮挡关系判断方法经常会失效，例如，图 3.5 中，无法判断出 A 遮挡 C。[30]中并没有很好的解决这个问题，为了增大遮挡关系判断算法的成功率，我们进一步提出了定理 3.4。

定理 3.4：对于两个导体块 A 和 B，曼哈顿型形体块$\{B_i\}$满足$\cup B_i \supseteq B$，曼哈顿型形体块$\{A_i\}$满足$\cup A_i \supseteq A$。则

1) 若对于任意 i 都有对于空间单元 T，A 遮挡 B_i，则对于 T，A 遮挡 B；

2) 若对于空间单元 T，$\cup A_i$ 遮挡 B，则对于 T，A 遮挡 B。

定理 3.4 是定理 3.3 的扩展版本，且很容易证明。当导体块 A 为曼哈顿型时，定理 3.4 中的 1)很容易判断，当导体块 B 为曼哈顿型时，定理 3.4 中的 2)同样可以

用基于定理 3.2 的方法判断。

利用定理 3.4，我们可以改进之前的对于非曼哈顿型导体块的遮挡关系判断算

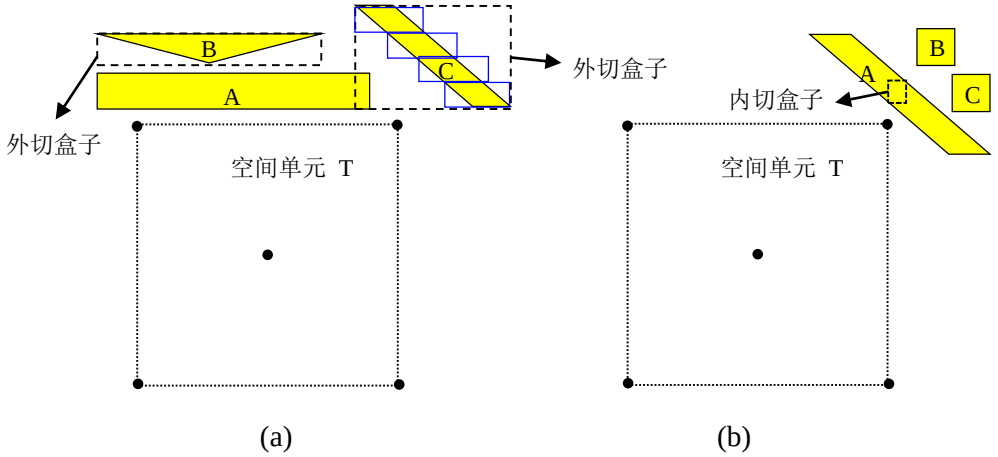


图 3.5 通过定理 3.3，可以判断 (a) A 遮挡 B (b) A 遮挡 B, A 与 C 的遮挡关系不容易判断

法，主要思想是将一个非曼哈顿型导体块用多个曼哈顿导体块来替代，然后通过定理 3.4 来判断曼哈顿导体块集之间的遮挡关系，我们称这种方法为“分组遮挡判断法”。如图 3.5(a)中，导体 C 用 4 个曼哈顿导体块（蓝线框）替代，然后通过定理 3.4 可以判断出导体块 A 遮挡导体块 C。如何选取非曼哈顿导体块的替代曼哈顿导体块集会影响遮挡关系判断的成功率和计算时间，当判断两个非曼哈顿导体块之间遮挡关系时，若替代的曼哈顿导体块的数目很大时，计算遮挡关系所花费的时间比遮挡关系正确判断所减少时间还要大，得不偿失。所以，我们应当限制替代非曼哈顿导体块的曼哈顿导体块数量。

在尝试将新导体加入到空间单元的候选列表算法（算法 2）中，空间单元 T 与导体 B 之间的距离 $d(\mathrm{B},\ \mathrm{T})$与定义 3.5 的意义是相同的，由于空间单元 T 是立方体形状，于是有

$$dist(\mathrm{B},\mathrm{T}) = dist_a(\boldsymbol{P},\mathrm{B}) - l/2, \qquad (3\text{-}7)$$

其中，l 是空间单元 T 的边长，$\boldsymbol{P}$ 是空间单元 T 的中心点，通过公式（3-7），可以快速计算空间单元 T 与导体块 B 之间的距离。

由于我们提出的判断非曼哈顿型导体块遮挡关系的算法使用的是充分条件，所以有时候仍然无法无法正确判断某些导体块的遮挡关系，特别是对于长斜导线这种情况，为了克服这个缺点，可以在预处理阶段将长斜导线切成多个独立的小导体块，这会使得导体块的数目增加，但由于使用了“分组遮挡判断法”，这同样提升了空间管理的效率。

3.3 使用旋转转移立方体的算法

[30]中还提出了使用旋转转移立方体的策略，同样为了论文的完整性，本节将对这种策略作一个简要介绍。

使用旋转转移立方体的好处是可以使得转移区域和导体的接触区域更大，从而增大随机行走过程的终止概率，如图 3.6(a)所示。在圆柱形 TSV 结构电容提取问题中，用到了旋转转移立方体的思想，并且带来了 2 倍的加速[24]。为了将旋转转移立方体应用到我们的问题中，首先给出如下定义。

定义 3.6：二维点 $\boldsymbol{P}(x, y)$和凸多边形 A（顶点逆时针顺序为 $A_1, A_2, \ldots, A_n$）之间

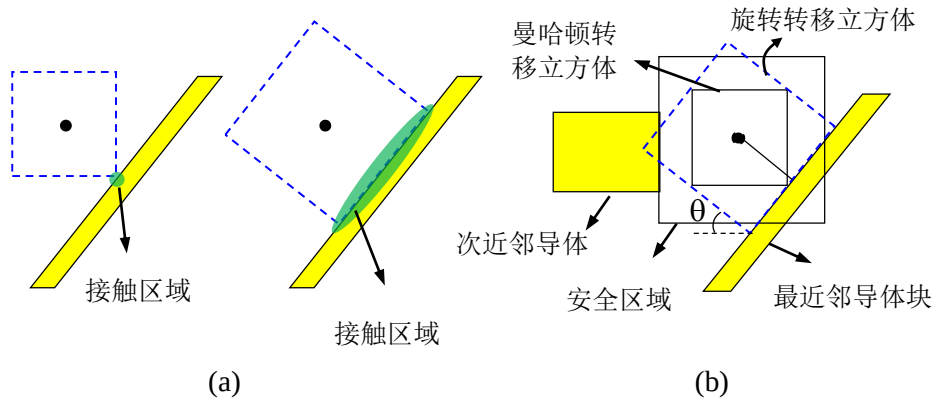


图 3.6 非曼哈顿导体和转移立方体顶视图 (a) 曼哈顿转移立方体 vs. 旋转转移立方体 (b) 曼哈顿转移立方体，旋转转移立方体，安全区域

的二维旋转盒子距离（rotated-box distance）为：

$$dist_r(P, \mathrm{A}) = \max\{\max_{1\le i\le n} \frac{(x-x_i)(y_{i+1}-y_i)-(y-y_i)(x_{i+1}-x_i)}{\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}}, dist_a(P, \mathrm{B_A})\}, \quad (3\text{-}8)$$

其中，$\mathrm{B_A}$ 是凸多边形 A 的曼哈顿外接盒子，(x_i, y_i) 是顶点 A_i 的坐标。

如图 3.7（a）所示，旋转转移立方体要么与导体 A 的侧边接触，要么与侧边接触（如点 $\mathrm{P_3}$）。与 $dist_a(\boldsymbol{P}, \mathrm{A})$的计算类似，$dist_r(\boldsymbol{P}, \mathrm{A})$为点到 A 的可见边正交距离的最大值，图 3.7（b）展示了 $dist_r(\boldsymbol{P}, \mathrm{A})$的计算过程。通过比较公式（3-1）和（3-8），可以发现 $dist_r(\boldsymbol{P}, \mathrm{A}) \ge dist_a(\boldsymbol{P}, \mathrm{A})$。

在悬浮随机行走过程中，我们通过计算对齐盒子距离来查找距离当前点 $\boldsymbol{P}$ 的最近邻导体，若得到的最近邻导体 A 为非曼哈顿型，可以尝试旋转转移立方体来获得一个更大的转移立方体。点 $\boldsymbol{P}$ 到 A 之间的旋转盒子距离计算公式如下:

$$dist_r(\boldsymbol{P}, \mathrm{A}) = \max\{dist_v(\boldsymbol{P}, \mathrm{A}), dist_r(\boldsymbol{P}, \mathrm{P_A})\}, \quad (3\text{-}9)$$

其中，P_A 表示 A 在 *xoy* 平面的投影，若 $dist_r(\boldsymbol{P}, A)$比 $dist_a(\boldsymbol{P}, A)$大，则相关的旋转转移立方体必然是非曼哈顿型的，且与 A 的某个侧面相接触。

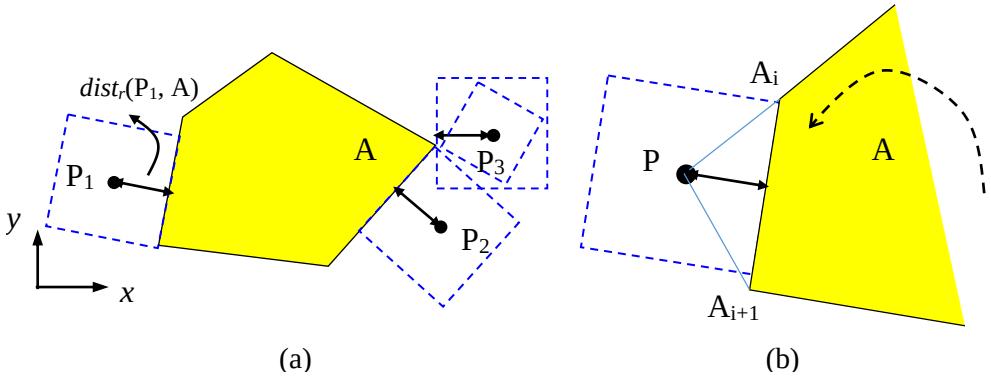


图 3.7. (a) 凸多边形 A 和点到 A 二维旋转盒子距离 (b) 二维旋转盒子距离计算演示

使用旋转转移立方体的一个前提条件是它不与其他任何导体相交，见图 3.6（b），按之前方法得到的旋转立方体会与次近邻导体相交，为了避免这种情况，可以将接触次近邻导体的曼哈顿立方体作为安全区域，只有当旋转转移立方体在安全区域内时才是有效的，否则应当仍然使用更小的曼哈顿转移立方体。于是，剩下的问题就是如何计算安全区域的大小了。

若使用额外的空间网格结构来管理非曼哈顿型导体，通过查询空间单元记录的导体信息，可以很容易的得到次近邻导体块。但若使用改进的空间管理方法，次近邻导体可能被遮挡，于是也就无法计算出安全区域。为了解决这个问题，需要修改遮挡关系判断算法。

定义 3.7： T 是一个空间单元，B_1，B_2 分别为两个导体块，且 B_1 是非曼哈顿型。若对于任意点 $\boldsymbol{P}\in T$，且 $\boldsymbol{P}\notin B_1\cup B_2$，$dist_a(\boldsymbol{P}, B_1)\le 0.5\times dist_a(\boldsymbol{P}, B_2)$，则我们说对于 T，$B_1$ 强遮挡 B_2。

定理 3.5：假设点 ***P*** 的最近邻导体块是非曼哈顿型导体 A，且导体块 B 是次近邻导体块，若 $dist_a(\boldsymbol{P}, A)\le 0.5\times dist_a(\boldsymbol{P}, B)$，则以 ***P*** 为中心的旋转转移立方体必然不会与任何导体相交。

证明：根据二维对齐距离和旋转盒子距离的定义可知 $dist_r(\boldsymbol{P}, A) \le \sqrt{2}dist_a(\boldsymbol{P}, A)$，如果有 $dist_a(\boldsymbol{P}, A)\le 0.5\times dist_a(\boldsymbol{P}, B)$，则有

$$dist_a(\boldsymbol{P}, B) \ge 2dist_a(\boldsymbol{P}, A) \ge \sqrt{2}dist_r(\boldsymbol{P}, A), \qquad (3\text{-}10)$$

如图 3.6(b)所示，以 ***P*** 为中心点的旋转转移立方体被包裹在一个半边长为 $(\sin\theta+\cos\theta)dist_r(\boldsymbol{P}, A)$，可以看出，半边长的大小不会超过$\sqrt{2}dist_r(\boldsymbol{P}, A)$，根据公式（3-10），也不会大于 $dist_a(\boldsymbol{P}, B)$。因此，可以看出，旋转转移立方体不会超出导体 B 所限制的安全区域的范围，定理 3.5 得证。

下面给出一个定理 3.5 的推论。

定理 3.6：假设点 ***P*** 的最近邻导体块是非曼哈顿型导体 A，点 ***P*** 在空间单元 T 中，若对于 T，A 强遮挡另外一个导体块 B，那么以 ***P*** 为中心的旋转转移立方体必然不会与导体块 B 相交。

基于定理 3.6，修改检查导体块 B 是否可以被添加到空间单元 T 的候选列中的算法，得到算法 3。

算法 3 修改的将新导体加入到空间单元的候选列表算法

输入：导体 B，空间单元 T，T 的候选导体列表 $list_T$
输出：添加成功返回 ture，否则返回 false
1：$d := dist_a(B, T)$; l 为 T 的边长
2：**If** $d \geq L(T)$ **then return** false
3：**For** b $\in list_T$ **do**
4：**If** b 是非曼哈顿型且 b 强遮挡 B 或 b 是曼哈顿型且 b 遮挡 B **then return** false
5： **Elseif** B 是非曼哈顿型且 B 强遮挡 b 或 B 是曼哈顿型且 B 遮挡 b **then**
6： 将 b 从 $list_T$ 中移除
7： **End**
8：**End**
9：将 B 加入到 $list_T$
10：**If** $(d + l) < L(T)$ **then** $L(T) := d + l$
11：**return** ture

由于强遮挡关系的判断条件比普通遮挡关系判断条件更加严格，所以，相对于使用曼哈顿转移立方体，使用旋转转移立方体会使得空间单元的候选导体列表中导体数目更多一些，这会对空间管理的效率造成一些影响。

综合来看，结合 3.4.1 节中提出的两种不同的空间管理策略（添加额外的空间结构方法和分组遮挡判断方法）和是否使用旋转转移立方体，可以组合得到四种不同的快速算法，下一节将对这四种快速算法进行实验比较。

3.4 实验结果与分析

我们在 RWCap2[18]的基础上实现了本章提出的各种算法，所有的实验都是在一台 Intel Xeon E5-2650 2.0GHz CPU 的 Linux 服务器上进行的。所有实验结果中的时间，都是串行计算的结果。所有实验的终止条件都是主导体电容精度为 0.5%。

本节所使用的 12 个测试用例描述如下：

测例 1~4：提取自混合信号电路设计中（如图 3.8）。每个测例的最大线宽均为 0.4μm，有四个金属层，包含 90 到 516 个导体块。从底向上，介质层的介电常数为 4.0 和 3.5 相互交替。在图 3.8(b)(c)中，展示了两个金属层布局示意图（主导体

显示为红色），且其中包含部分非曼哈顿的型导体块。

测例 5~7：如图 3.9(a)所示，由 *m* 根有损导线组成的包装互连结构，单介质[12]。其中 *m* 的取值分别为 8、16 和 32。每根导线的宽度和高度分别为 0.12μm 和 0.2μm。每个测例的中间导线被设为主导体。

测例 8：如图 3.9(b)所示，包含两个金属层。第一层有 50 根与 *x* 轴平行的对齐导线，第二层有 50 根倾斜 40°角的斜导线。两层导线间的距离为 0.36μm，每根导线的宽度和高度分别为 0.12μm 和 0.2μm，导线长度为 35μm。

测例 9~12：分别将测例 1~4 以 2×2 的方式平铺，得到更大的测试用例，包含 360 到 2064 个导体块。

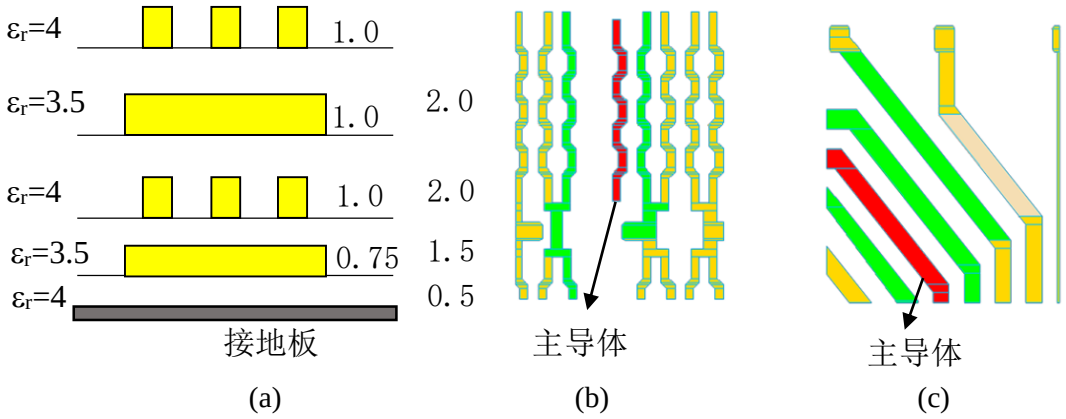


图 3.8 (a) 测例 1~4 工艺技术横截面示意，介质层厚度和金属高度单位都是μm (b)和(c) 测例 1~4 中两个顶视图示例

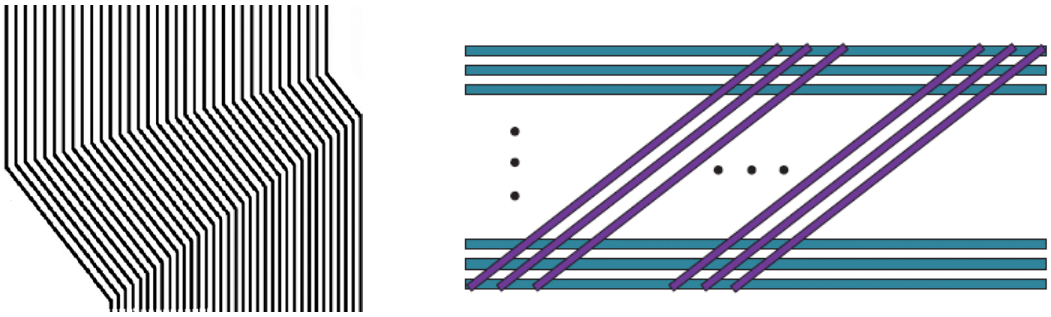


图 3.9 (a)*m* 根有损导线组成的包装互连结构 (b)测例 8 的顶视图，包含两个金属层

根据不同的空间管理策略和是否使用旋转转移立方体，将测试算法分类如下，其中，Alg. 2-3、Alg. 5-6 是本文提出或改进的快速算法。

Alg. 1：基于曼哈顿转移立方体的悬浮随机行走算法，使用遍历的方法处理非

曼哈顿型导体。

Alg. 2：基于曼哈顿转移立方体的悬浮随机行走算法，使用额外的空间网格结构来处理非曼哈顿型导体。

Alg. 3：基于曼哈顿转移立方体的悬浮随机行走算法，使用改进的空间管理技术来处理非曼哈顿型导体。

Alg. 4：基于旋转转移立方体的悬浮随机行走算法，使用遍历的方法处理非曼哈顿型导体。

Alg. 5：基于旋转转移立方体的悬浮随机行走算法，使用额外的空间网格结构来处理非曼哈顿型导体。

Alg. 6：基于旋转转移立方体的悬浮随机行走算法，使用改进的空间管理技术来处理非曼哈顿型导体。

在下面的 3.4.1 到 3.4.3 节中，所有测试用例都被修改为介电常数为 1 的单介质结构，在 3.4.4 节，测试了多介质结构（如测例 1~4 和测例 9~12）的情况。

3.4.1 使用曼哈顿型转移立方体实验结果

表 3.1 展示了 Alg. 1~3 和 QBEM 的实验结果。由于 Alg. 1~3 的电容结果几乎相同（误差小于 2%），所以表 3.1 中仅列出了 Alg. 3 的电容结果。从表 3.1 中可以看到 QBEM 与 FRW 算法得到的电容结果之间的误差非常小，在 2.5% 以内。由于 QBEM 默认为纽曼边界[7]，所以 FRW 算法得到的结果会比 QBEM 结果偏大一些。这一实验结果验证了我们所提出的处理非曼哈顿型导体块方法的正确性。

表 3.1 中同样展示了 Alg. 1~3 之间的性能差异，可以看到，使用了空间管理的 Alg. 2 和 Alg. 3 比不使用空间管理的 Alg. 1 速度更快，且随着非曼哈顿型导体数目的增多加速比也越大，最大可以加速 84 倍。对比 Alg. 2 和 Alg. 3，可以发现 Alg. 3 的速度比 Alg. 2 更快一些，可以加速 1.2 倍~1.9 倍，特别对于测例 8，Alg. 3 比 Alg. 2 快 4 倍，这主要是因为 Alg. 3 的候选导体列表比 Alg. 2 的更小，使得每次跳转速度更快。表 3.2 中展示了这三个算法的更多细节，可以看出，Alg. 1 和 Alg. 3 每次随机行走的平均跳转数（#hop）几乎相同，Alg. 2 比它们稍微大一些，这是因为使用了邻近区域和不完全候选导体列表技术。由于 Alg. 3 在生成候选导体列表时，需要考虑非曼哈顿型导体之间的遮挡关系，所以在建立空间管理结构时需要花费更多的时间（T_{sp}），但内存使用上却会比 Alg. 2 更少。Alg. 1~3 所使用的的方法并不会影响随机行走的总行走数（#walk），所以表 3.2 中没有列出这个值。

为了比较基本的 FRW 算法和 QBEM，回顾表 3.1，可以发现对于前 8 个测例，Alg. 3 最大比 QBEM 快了 36 倍，对于后 4 个测例，由于计算所需内存太大，QBEM

无法得到结果。从内存使用上来看，与 QBEM 相比，FRW 算法具有巨大的优势。这些实验结果反映出了基于 BEM 和 FRW 算法的电容求解器的主要不同之处。

表 3.1 QBEM 与使用曼哈顿转移立方体的 FRW 算法实验结果比较

测例	导体块数	非曼哈顿型导体块数	QBEM			FRW			
			内存	电容(aF)	时间(s)	#walk	#hop	内存	电容(aF)
1	90	28	101MB	173.3	2.88	260K	15.1	<1MB	176.1
2	264	134	982MB	264.9	28.6	295K	16.9	<1MB	266.4
3	378	188	1.4GB	406.1	51.3	394K	30.2	<1MB	413.4
4	516	310	1.6GB	489.5	55.5	208K	13.3	1.1MB	486.9
5	24	24	103MB	585.0	3.24	361K	25.2	<1MB	600.1
6	48	48	180MB	586.7	5.74	355K	24.3	1.1MB	595.6
7	96	96	324MB	584.8	10.3	321K	22.7	1.5MB	599.0
8	100	50	550MB	1218	42.4	370K	34.0	2.6MB	1275
9	360	112	--	--	--	260K	14.5	<1MB	176.4
10	1076	536	--	--	--	291K	16.6	1.7MB	268.3
11	1512	752	--	--	--	393K	30.5	2.5MB	412.6
12	2064	1240	--	--	--	206K	13.3	3.7MB	480.3

续表 3.1 QBEM 与使用曼哈顿转移立方体的 FRW 算法实验结果比较

测例	FRW 算法 CPU 时间 (s)			SP_{Alg1}^{Alg2}	SP_{Alg1}^{Alg3}	SP_{QBEM}^{Alg3}
	Alg. 1	Alg. 2	Alg. 3			
1	5.02	2.07	1.76	2.4	2.9	1.6
2	21.1	3.12	2.56	6.8	8.2	11
3	35.8	5.85	4.73	6.1	7.6	11
4	24.2	1.95	1.56	12	16	36
5	10.63	6.15	4.31	1.7	2.5	0.8
6	16.17	7.23	3.86	2.2	4.2	1.5
7	26.2	6.07	3.48	4.3	7.5	3.0
8	27.7	23.0	5.84	1.2	4.7	7.3
9	16.81	2.19	1.60	7.7	11	--
10	111.7	3.40	2.70	33	41	--
11	167.2	6.61	4.77	25	35	--
12	132.1	2.11	1.58	62	84	--

SP_{Alg1}^{Alg2} 是 Alg. 2 相对于 Alg. 1 的加速比. SP_{Alg1}^{Alg3}是 Alg. 3 相对于 Alg. 1 的加速比. .

SP_{QBEM}^{Alg3}是 Alg. 3 相对于 QBEM 的加速比.

表 3.2 使用曼哈顿转移立方体和使用旋转转移立方体 FRW 算法的更详细实验结果比较

测	Alg. 1		Alg. 2				Alg. 3			
例	#hop	时间(s)	#hop	T_{sp}(s)	Mem_{sp}	时间(s)	#hop	T_{sp}(s)	Mem_{sp}	时间(s)
1	15	5.02	17	0	2.1MB	2.07	15	0	<0.1MB	1.76
2	16.9	21.1	17.9	0	2.5MB	3.12	16.8	0	0.1MB	2.56
3	30.2	35.8	33.3	0.01	3.1MB	5.85	30.3	0.01	0.1MB	4.73
4	13.3	24.2	13.3	0.01	2.5MB	1.95	13.3	0.02	0.1MB	1.56
5	24.8	10.63	27.6	0	1.8MB	6.15	25.5	0.01	0.1MB	4.31
6	23.7	16.17	25.8	0	2MB	7.23	24.4	0.02	0.3MB	3.86
7	22.1	26.2	23.7	0	2.5MB	6.07	22.8	0.03	0.4MB	3.48
8	30.7	27.7	30.8	0.01	2.1MB	23.0	34.2	0.14	0.9MB	5.82
9	14.5	16.81	15.2	0	2MB	2.19	14.5	0.01	<0.1MB	1.60
10	16.5	111.7	17.1	0.02	2.6MB	3.40	16.5	0.03	0.3MB	2.70
11	30	167.2	31.5	0.03	3.5MB	6.61	30.2	0.05	0.4MB	4.77
12	13.3	132.1	13.3	0.03	2.9MB	2.11	13.3	0.07	0.9MB	1.58

续表 3.2 使用曼哈顿转移立方体和使用旋转转移立方体 FRW 算法的更详细实验结果比较

测	Alg. 4			Alg. 5			Alg. 6			
例	#hop	时间(s)	SP_{Alg1}^{Alg4}	#hop	Mem_{sp}	时间(s)	#hop	Mem_{sp}	时间(s)	SP_{Alg1}^{Alg6}
1	13.5	4.68	1.1	15.5	2.1MB	1.92	13.5	<1MB	1.73	2.9
2	13.2	16.48	1.3	14.1	2.5MB	2.79	13.2	<1MB	2.43	8.7
3	14.2	18.60	1.9	17.3	3.1MB	3.66	14.2	<1MB	2.95	12
4	10.7	18.10	1.3	10.7	2.5MB	1.76	10.7	<1MB	1.41	17
5	14.8	7.20	1.5	17.6	1.8MB	4.11	15.1	<1MB	5.13	2.1
6	14.3	11.47	1.4	16.3	2.1MB	4.64	14.7	2.3MB	3.35	4.8
7	13.8	18.57	1.4	15.4	2.5MB	4.06	16.2	<1MB	3.61	7.3
8	13.4	12.99	2.1	13.6	2.1MB	10.43	17.0	<1MB	3.81	7.3
9	12.9	14.77	1.1	13.6	2MB	1.97	12.9	<1MB	1.58	11
10	12.8	86.64	1.3	13.4	2.6MB	2.99	12.8	<1MB	2.41	46
11	13.9	88.53	1.9	15.5	3.5MB	3.87	14.2	1MB	3.04	55
12	10.7	99.10	1.3	10.7	2.9MB	1.87	10.9	2.1MB	1.37	96

SP_{Alg1}^{Alg4} 是 Alg. 4 相对于 Alg. 1 的加速比. SP_{Alg1}^{Alg6}是 Alg. 6 相对于 Alg. 1 的加速比.

3. 4. 2 使用旋转转移立方体实验结果

Alg. 4~6 是使用了旋转转移立方体的 FRW 算法，表 3.2 中同样列出了其实验结果，为了便于更好的比较 Alg. 1~6，我们已经验证了它们的电容结果都是正确的，所以电容结果数据没有包含在表 3.2 中。通过比较 Alg. 4 和 Alg. 1，我们发现使用旋转转移立方体带来了平均 1.47 倍，最高 2.1 倍的加速比，其主要原因是使用旋转转移立方体减少了每次行走的平均跳转数（#hop）。当比较 Alg. 6 和 Alg. 3 的实验数据时，可以发现使用旋转转移立方体仍然是有优势的，但是加速比比之前要减小，只有平均 1.2 倍，最大 1.6 倍，这主要是因为 Alg. 6 的空间管理需要进行强遮挡关系检查，这会引入额外的开销。与使用曼哈顿型转移立方体类似，使用改

进的空间管理技术（Alg. 6）比使用空间网格的空间管理技术（Alg. 5）在总运行时间和内存使用上都要更佳。

如果将 Alg. 6 和 Alg. 1 的实验结果进行比较，可以发现其加速比达到了 2.9 到 96 倍，这主要是因为使用了旋转转移立方体和改进的遮挡关系判断算法。同样的，Alg. 6 对 QBEM 最大有 39 倍的加速比（测例 4）。这说明使用旋转转移立方体是非常有效的一种技术。

3.4.3 进一步的准确度和有效性验证

为了进一步对我们提出的算法进行准确度和有效性的验证，我们额外测试了两个交叉线的例子。第一个例子与测例 8 很相似，不同之处在于第二层的导线与第一层的导线是垂直的。第二个例子是[17]中的提到的 100×100 交叉线测例。由于这两个例子都只包含曼哈顿型导体，所以可以被已有的电容求解器（如 RWCap2[17，18]）求解。我们将这两个测例旋转 40°角，得到其对应的非曼哈顿型结构，再用我们提出的方法对这两个旋转后的测例进行测试，将实验结果与 RWCap2 的结果进行对比。表 3.3 中列出了对应的实验结果。从表 3.3 中可以看到，旋转后的倾斜测例的电容结果与原始测例的电容结果基本相同，且 100×100 交叉线这个测例与[17]中的结果相同，这进一步验证了我们提出的方法的正确性。另外可以看到，我们提出的方法计算速度比 RWCap2 要慢，这是因为处理非曼哈顿型结构时，每次行走所需的平均跳转数更大，这会使得计算时间更长。对于这两个测例，当比较 Alg. 1 和 Alg. 6 时，可以发现我们的方法分别带来了 11 倍和 8 倍的加速比。为了观察当非曼哈顿型导体数目较少时，我们提出的算法的性能，我们对测例 1 进行了修改，使其仅包含 5 个非曼哈顿型导体块，且总导体块数为 90，得到测例 1a，表 3.4 中列出了对应的实验结果，列出测例 1 的数据是为了方便比较。

表 3.3 交叉线测例和其对应的倾斜测例实验结果（电容单位为 aF）

测例	RWCap2			Alg. 1		Alg. 3		Alg. 6		
	#walk#hop	时间(s)	电容	时间(s)	电容	时间(s)	电容	时间(s)	电容	#walk#hop
50×50	248K	11.3	1.17	856	76.7	853	10.3	853	6.7	855	488K	22.9
100×100	152K	8.4	0.54	120	40.0	120	6.7	120	5.2	120	418K	11.6

从表 3.4 中可以看到，我们提出的方法在包含少量非曼哈顿型导体时同样具有与之前类似的加速比。由于非曼哈顿导体数目的减少，使得基于 FRW 方法的求解器计算时间减少，其对于 QBEM 的加入比从 1.6 倍增加到了 2.2 倍。我们更进一步的修改测例 1a 得到一个不包含非曼哈顿型的测例 1b，测例 1b 可以用 RWCap2 来进行计算，其计算时间为 1.23 秒，这仅比用我们方法计算测例 1a 的 1.26 秒少一

点点，由此可见，我们提出的方法的计算时间与所计算的结构所包含的非曼哈顿型导体块的数目是成正比的。

表 3.4 包含少量非曼哈顿型导体块结构的实验结果

测例	QBEM		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6		SP_{Alg1}^{Alg3}	SP_{QBEM}^{Alg3}
	电容(aF)	时间(s)	时间(s)					电容(aF)	时间(s)		
1	173.3	2.88	5.02	2.07	1.76	4.68	1.92	176.1	1.73	2.9	2.6
1a	176	2.82	3.45	1.65	1.26	3.38	1.57	180	1.31	2.7	2.2

SP_{Alg1}^{Alg3} 是 Alg. 3 相对于 Alg. 1 的加速比. SP_{QBEM}^{Alg3}是 Alg. 3 相对于 QBEM 的加速比.

表 3.5 多介质结构测例实验结果

测例	导体块数	非曼哈顿型导体块数	QBEM			FRW			
			内存	电容(aF)	时间(s)	#walk	#hop	内存	电容(aF)
1	90	28	100MB	667.8	2.98	300K	16.5	11.3MB	682.5
2	264	134	985MB	1023	28.5	886K	15.6	11.6MB	1040.0
3	378	188	1.42GB	1573	51.2	423K	15.8	11.6MB	1602.0
4	516	310	165GB	1801	55.8	240K	12.1	12.1MB	1781.0
9	360	112	--	--	--	304K	16.4	11.5MB	680.8
10	1056	536	--	--	--	865K	15.4	12.8MB	1038.0
11	1512	752	--	--	--	424K	16.1	13.4MB	1600.0
12	2064	1240	--	--	--	238K	12.1	15.0MB	1785.0

续表 3.5 多介质结构测例实验结果

测例	FRW 算法 CPU 时间(s)			SP_{Alg1}^{Alg6}	SP_{QBEM}^{Alg6}
	Alg. 1	Alg. 3	Alg. 6		
1	7.39	2.65	2.61	2.8	1.1
2	76.53	9.51	8.87	8.6	3.2
3	42.23	5.64	3.75	11.3	14
4	29.22	2.09	1.92	15.2	29
9	25.46	2.52	2.41	10.6	
10	382.53	9.57	9.29	41.2	
11	200.84	5.96	3.88	51.8	--
12	172.36	2.01	1.89	91.2	--

SP_{Alg1}^{Alg6} 是 Alg. 6 相对于 Alg. 1 的加速比. SP_{QBEM}^{Alg6}是 Alg. 6 相对于 QBEM 的加速比.

3.4.4 多介质测例结果

对于多介质测例，首先需要用 TechGFT 程序[16]来生成 FRW 算法所需要的转移概率和权值表。表 3.5 中列出了 QBEM 和我们提出的方法的实验结果，对于大例子，QBEM 无法计算出结果。从表中可以看出两种方法得到的导体自电容误差小于 2.5%，且 Alg.6 对 QBEM 的加速比为 29 倍，对 Alg. 1 的加速比为 91 倍，与

单介质情况结果类似。这些实验结果说明本章提出的方法同样适用于多介质结构。

3.5 本章小结

本章讨论了 FRW 算法对于非曼哈顿型导体的处理，根据 3.4 节的实验结果，可以得到如下结论：

1. 我们提出的方法可以准确的处理包含非曼哈顿型导体块的结构，实验结果很好的验证了电容结果的准确性。

2. 与通过遍历非曼哈顿型导体块来构造转移立方体的方法相比，我们提出的利用遮挡关系的空间管理技术可以带来 2.9~84 倍的加速比，且非曼哈顿导体块数目越多，加速效果越好。

3. 与普通的曼哈顿型转移立方体相比，利用旋转转移立方体的 FRW 方法可以带来最大 2.1 倍的加速比。当与我们提出的基于遮挡关系的空间管理技术结合使用时，可以带来最大 96 倍的加速比。

4. 与利用额外的空间网格结构来管理非曼哈顿型导体块的方法相比，我们提出的基于遮挡关系的空间管理技术带来了最大 4 倍的加速比，且使用的内存更少。

5. 相比于基于 BEM 的 QBEM 电容求解器，我们提出的 FRW 算法比它快了最大 39 倍，并且使用的内存比之少了数个量级。

6. 尽管我们提出的方法在处理非曼哈顿型导体块时会牺牲一些运行时间，但运行时间的开销基本上与计算结构中非曼哈顿型导体块的数目成正比。这意味着当计算结构中只包含少量非曼哈顿型导体时，对运行时间基本上不会有太大影响。

从结果上看，使用旋转转移立方体并没有带来非常大的加速比，事实上，当计算结构中只包含少量非曼哈顿型导体块时，它甚至会比使用曼哈顿型转移立方体的算法更慢。在[24]中，虽然旋转转移立方体并不能显著的增大在圆柱形硅通孔结构的圆柱表面的接触面积，但它仍然带来了非常好的加速效果，原因其实与旋转转移立方体的使用频率有关，在本章的测试用例中，只有当随机行走的当前点位于非曼哈顿型导体的倾斜面附近时，才会使用旋转转移立方体，但这种倾斜面的面积只占了多有导体总表面积很小的一部分。而在[24]中的圆柱形硅通孔测例中，圆柱侧表面积所占比例很大，所以随机行走的当前点经常位于其附近，这使得旋转转移立方体被频繁使用，所以带来了比本章更大的加速比[24]。

第4章 悬浮导体的处理

本章主要介绍了悬浮随机行走（Floating Random Walk，FRW）算法处理悬浮导体的解决方案。4.1 节从悬浮导体的物理特性进行推导，得到了算法的理论依据。4.2 节介绍了曼哈顿型悬浮导体的处理方法。4.3 节介绍了关于非曼哈顿型悬浮导体的处理方法。4.4 节通过实验验证了本章算法的正确性，并对影响电容结果准确性和算法运行时间的积分面距离参数作了进一步的讨论，提出了一种可以自动确定该参数取值的策略。

4.1 理论推导

由于悬浮导体本身不带电且不接地，所以其满足电荷量 $Q = 0$。根据高斯定理，作包围悬浮导体的积分面 S，则有 $\oint_S \varepsilon(r) \cdot \vec{E} ds = 0$，用电势的梯度替换电场强度 $\vec{E}$，则有

$$\oint_S \varepsilon(r) \cdot (-\nabla \phi(r)) ds = 0, \tag{4-1}$$

当积分面距离悬浮导体很近时，积分面上一点的电势梯度可以用中心差分的方法来近似，如图 4.1(a)。于是有

$$\oint_S \varepsilon(r) \cdot \left(\frac{\phi(r^{(1)}) - \phi(r^{(2)})}{2d(r)} \right) ds = 0, \tag{4-2}$$

其中 $d(r)$为 r 点到悬浮导体的距离。将（4-2）整理可得

$$\left(\oint_S \frac{\varepsilon(r)}{d(r)} ds \right) \phi(r^{(2)}) = \oint_S \varepsilon(r) \frac{\phi(r^{(1)})}{d(r)} ds, \tag{4-3}$$

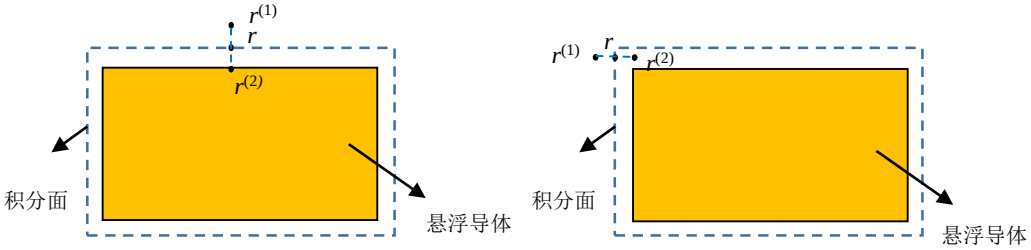
对于给定的悬浮导体，（4-3）左边积分为常数且非零，假设其值为常数 K。则有

$$\phi(r^{(2)}) = \oint_S P(r) \cdot \phi(r^{(1)}) ds, \tag{4-4}$$

其中，$P(r) = \frac{\varepsilon(r)}{K \cdot d(r)}$。因为悬浮导体本身是一个等势体，所以悬浮导体的电势 $\phi(f)$= $\phi(r^{(2)})$。（4-4）表明悬浮导体的电势 $\phi(f)$可以通过在 $r^{(1)}$点所构成的采样面上按照概率密度函数 $P(r)$进行采样得到。

在实际悬浮随机行走过程中，当点位于悬浮导体上时，根据对应的概率密度

函数，在 $r^{(1)}$点构成的采样面上随机选取一点，以该点为当前点继续随机行走过程，算法 4 描述了可以处理悬浮导体的 FRW 电容求解算法过程。



(a) (b)

图 4.1 中心差分近似计算悬浮导体积分面上 r 点电势梯度
(a) 点 $r^{(2)}$在悬浮导体上 (b) 点 $r^{(2)}$不在悬浮导体上

算法 4 可以处理悬浮导体的 FRW 电容求解算法

输入： 互连线导体的三维版图（包括悬浮导体），主导体 i

输出： 电容 C_{ij}，(j=1,2,…)

1: 加载预计算的转移概率表和权值表
2: 构造包裹主导体 i 的高斯面 G
3：C_{ij} := 0，$\forall i$；n_{path} := 0
4：构造包裹悬浮导体 k,(k=1,2,…)的积分面 S_k
5：计算采样面 S_k 对应的概率密度函数 $P_k(r)$
6：**Repeat**
7： n_{path} := n_{path} + 1
8： 在 G 上随机选取一点 $r^{(0)}$，并以 $r^{(0)}$为中心生成转移立方体区域 T，根据转移概率表在区域 T 的表面随机选取一点 $r^{(1)}$，根据权值表计算出权值 w
9： **While** $r^{(1)}$不在导体（非悬浮导体）表面 **do**
10： **If** $r^{(1)}$在悬浮导体 k 表面 **do**
11： 根据概率密度函数 $P_k(r)$在积分面 S_k 上选择一点 r，再找到该点对应的采样点 $r^{(2)}$
12： **Else**
13： 以 $r^{(1)}$为中心生成新的转移立方体区域 T，并在 T 的表面根据转移概率表随机选取一点 $r^{(2)}$
14： **End**
15： $r^{(1)}$:= $r^{(2)}$
16：**End**
17：C_{ij} := C_{ij} + w
18：**Until** C_i^j 精度满足终止条件
19：C_{ij} := C_{ij} / n_{path}，$\forall i$

4.2 曼哈顿型悬浮导体处理

[31]提出了一种处理曼哈顿型悬浮导体的方法，为了论文完整性，本节对其作简要介绍。另外需要指出的是，由于算法实现问题，[31]中的实验结果部分并不正确，本章在 4.4 节中给出了正确的实验结果。

[33]对利用悬浮随机行走算法处理曼哈顿型悬浮导体进行了研究，其理论与 4.1 节类似，使用的积分面如图 4.1 所示，但仔细分析可以发现，当积分面边角处的点使用中心差分计算时，其所对应的 $r^{(2)}$点不在悬浮导体上[如图 4.1(b)]时，所以使用这种积分面的方法其实是不准确的，可能对结果产生影响。

为了解决这一问题，[31]提出了修改积分面的方法，主要思想就是“切掉”积分面的边角处，如图 4.2 所示，为了简化问题，边角处倾斜度为 45°，这样就使得 $r^{(2)}$点总是可以落在悬浮导体上。

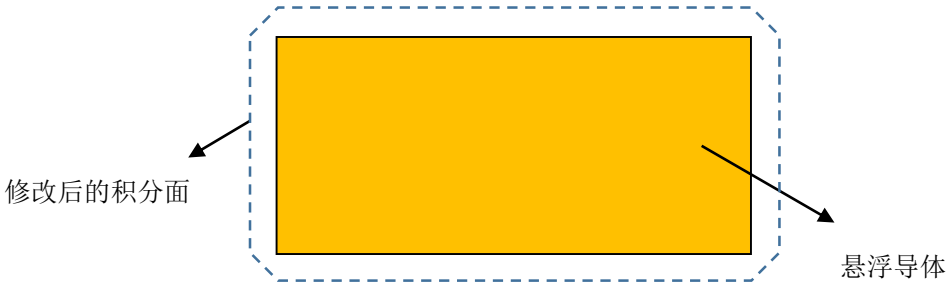


图 4.2 修改后的积分面示意图

在三维情况下，修改后的积分面实际上一个有 26 面构成的多面体，可以根据各个面上 $d(r)$取值的情况将 26 个面分为 3 类[31]：

1. 由悬浮导体 6 个表面“外移”得到，称之为一类面
2. 由相邻两个一类面的两条棱所处平面构成的 12 个面，称之为二类面
3. 由相邻三个一类面的相邻顶点所确定的面，共 8 个，称之为三类面

通过计算推导可以得到[31]：一类面 $d(r)$为常数，记为 d，面上所有点的概率积分为$\frac{S}{d}$，S 为一类面面积；二类面$d(r)=\sqrt{2}d-x\left(x\le\frac{\sqrt{2}}{2}d\right)$，面上所有点的概率积分为 $2l\ln 2$，l 为二类面的棱长；三类面$d(r)=2H-h(r)$，H 为四面体的高，$h(r)$为点 r 到侧面的距离，面上所有点的概率积分为 $3d$-$3d\ln 2$。所有面的概率密度函数都是 $P(r)$都正比于$\frac{1}{d(r)}$。

4.3 非曼哈顿型悬浮导体处理

[31]和[33]中并没有提到如何处理非曼哈顿型悬浮导体，实际上，对于非曼哈顿型悬浮导体，可以同样使用类似于 4.2 节中提到的在边角处修改的积分面，如图 4.3 所示。但对于这种积分面，像 4.2 节中那样精确计算其各个面的概率积分非常困难，所以直接使用均匀采样的方法进行计算，4.4 节中的实验结果表明，这种情况下得到的电容准确度仍然较高（误差在 2%以内）。

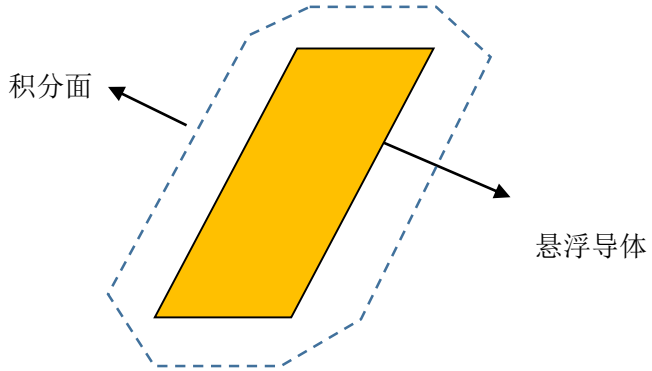


图 4.3 非曼哈顿型悬浮导体积分面

4.4 实验结果与分析

基于 RWCap2[18]我们实现了本文提出的处理悬浮导体的算法，所有实验的终止条件都是主导体电容精度为 0.5%。

本节所使用的 7 个测试用例描述如下：

测例 1~3：[33]中的三个测试用例，分别包含 24、34 和 53 块曼哈顿型悬浮导体块，不含有非曼哈顿型悬浮导体块。

测例 4~6：三个测试用例均为 10×10 交叉线结构，包含两个金属层，第一层有 10 根与 x 轴平行的对齐导线，第二层有 10 根倾斜 40°角的斜导线。两层导线间的距离为 0.36μm，每根导线的宽度和高度分别为 0.12μm 和 0.2μm，导线长度为 35μm。不同之处为分别从 10 根倾斜导线中选出 1、3、5 根作为悬浮导体。

测例 7：线宽为 1μm，只包含 4 块导体，且其中只有一块悬浮导体，其顶视图如图 4.4 所示,其中导体 F 为悬浮导体块，导体 A 为主导体，且导体 A、B 和位于同一金属层，导体 C 位于另一金属层内。

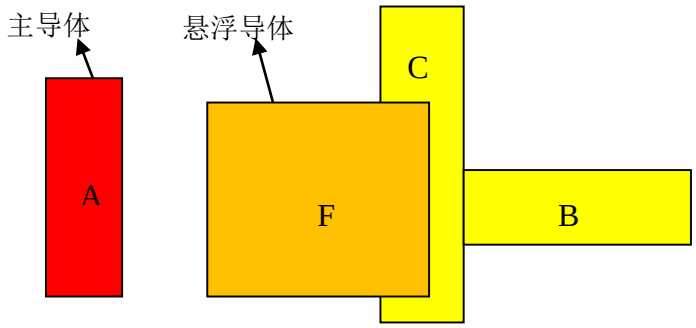


图 4.4 测例 7 顶视图

4.4.1 准确性验证

为了验证本文提出的算法的准确性，使用商业软件 Raphael[13]的计算结果作为电容参考值，RWCap3 表示我们提出的方法，表 4.1 和表 4.2 分别给出了在单介质和多介质两种情况下的实验结果数据，为了方便对比，表 4.2 中同样给出了[33]中的实验结果。从表中数据可以看出，我们提出的方法与 Raphael 给出的电容参考值之间的相对误差都在 2%以内，这验证了我们提出的方法的准确性。对比[33]的实验结果，发现我们的方法误差更小，这说明 4.2 中提出的修改积分面的方法确实有效。从表 4.1 和表 4.2 中都可以发现测例 1~3 的误差要小于测例 4~6，这是因为测例 4~6 中包含非曼哈顿型悬浮导体，4.3 节中提到过，对于非曼哈顿型悬浮导体的处理方法其实是并不十分精确的。

表 4.1 悬浮导体处理单介质情况下实验结果

测例	导体块数	悬浮导体块数	Raphael 电容(aF)	RWCap3 电容(aF)	误差(%)
1	27	24	1861	1867	0.32
2	38	34	666.8	667.7	0.13
3	57	53	570.8	569.4	-0.25
4	20	1	127	128	0.79
5	20	3	125.8	127.2	1.11
6	20	5	116.6	117.7	0.94

表 4.2 悬浮导体处理多介质情况下实验结果

测例	导体块数	悬浮导体块数	Raphael 电容(aF)	[33] 电容(aF)	误差(%)	RWCap3 电容(aF)	误差(%)
1	27	24	2470	2446	-0.97	2468	-0.08
2	38	34	753.2	766.1	1.71	758.4	0.69
3	57	53	741.6	732.1	-1.28	739.2	-0.32
4	20	1	431.3	-	-	436.3	1.16
5	20	3	425.8	-	-	432.8	1.64
6	20	5	-	-	-	383.2	-

4.4.2 参数影响分析

回顾 4.1 节的理论推导，有一个积分面距离参数 $d(r)$，在实际实现中，参数 $d(r)$ 为一个常数，假设为 d，根据中心差分的原理知道参数 d 会影响中心差分结果的准确度，d 越小，结果越精确。但当 d 很小时，采样面距离悬浮导体的距离就会很近，采样点就也在悬浮导体附近，这会使得下一次跳转仍然落在悬浮导体的概率大大增加，不利于随机行走过程的终止，导致计算时间增加。为了进一步探讨参数 d 对准确度和计算时间的影响，我们特别构造了测例 7，测例 7 中悬浮导体对电容结果的影响很大，可以更容易的测试出不同数值的参数 d 对电容结果的影响。

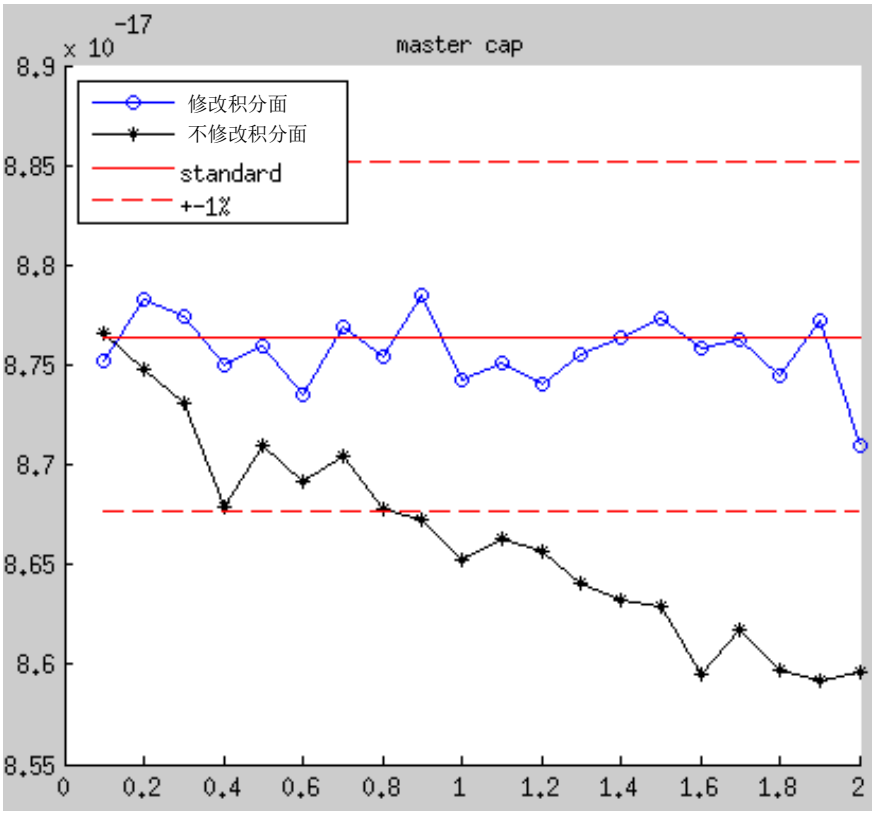


图 4.5 参数 d 对电容结果准确度的影响，横轴为参数 d 的值，单位为μm；纵轴为电容值，单位为 F

图 4.5 展示了参数 d 对电容结果准确度的影响，由于测例 7 的线宽为 1μm，所以采样点的选取方法为从 d=0.1μm 开始，以 0.1μm 为步长，直到 d=2μm 结束，注意，d=2μm 时悬浮导体 F 的采样面可以保证不与任何导体接触。以 d=0.01μm 计算 3000 次得到的平均电容值作为标准值，在图 4.5 中用红色实线标出，上下两条红色虚线标示出偏离标准值正负 1%的范围。从图 4.5 中可以看出，采用修改积分面后的方法计算得到的电容结果具有良好的准确度，即使在 d 很大时，其结果误差仍然处于±1%范围内，而不修改积分面的方法在 d 比较小时准确度较高，但当 d

增大时，准确度明显开始下降。

图 4.6 展示了参数 *d* 对计算时间的影响，可以看到，随着 *d* 的增大，计算时间越来越小，但当 *d* 的值超过某一界限时，计算时间的减少其实并不明显了。另外，*d* 的取值大小其实是有上限的，回顾图 4.1，当随机行走过程当前点落在悬浮导体上时，根据前面推导的定理，实际上需要将当前点转移到采样面上某一点处，但转移后的点应当不能与其他导体接触或在其他导体内部，所以这要求采样面所包裹的区域不包含任何导体，而采样面的构造与参数 *d* 的取值有关，*d* 越大，采样面包裹的区域就越大，所以 *d* 的取值不能无限大，参数 *d* 取值的最大值应当为悬浮导体距离其他导体最近距离的一半。

综合图 4.5 和图 4.6 来看，修改积分面方法在 *d*=1.9μm 时与不修改积分面方法在 *d*=0.2μm 时精确度差不多，但前者的计算速度比后者快了大约 35%。这说明对于曼哈顿型悬浮导体，修改积分面的方法具有精度高、速度快的优点。

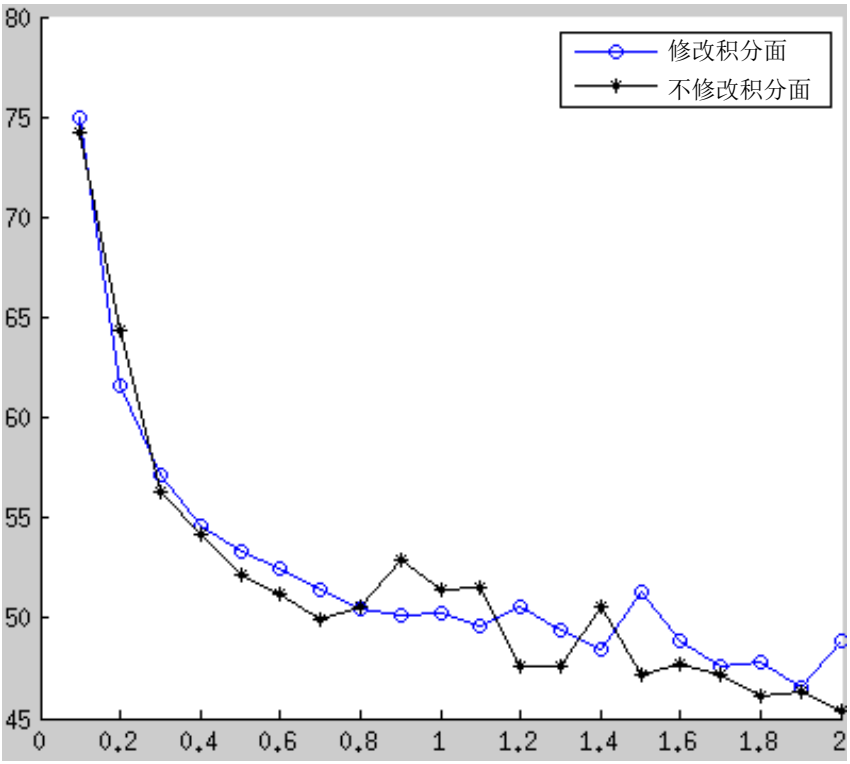


图 4.6 参数 *d* 对计算时间的影响，横轴为参数 *d* 的值，单位为μm；纵轴为时间，单位为秒

对于不同的悬浮导体，其对应的积分距离参数 *d* 的取值应当是不同的。结合上面的实验，可以给出一种经验性的自适应取值方法：假设悬浮导体的最小线宽为 *W*，悬浮导体到其他导体的最近距离为 *D*，则对于曼哈顿型悬浮导体，根据前面的实验结果可以看出，*d* 的取值可以尽可能大，即取值为 *D*/2；而对于非曼哈顿

型悬浮导体，d 太大会导致结果不准确，折衷来看，其取值大小可以为

$$d = min\left(\frac{W}{2},\ \frac{D}{2}\right), \tag{4-5}$$

这样的取值不一定是最优的，但在实际应用中却可以很好的平衡精确度和计算时间之间的关系。

4.5 本章小结

针对曼哈顿型悬浮导体，实现了一种快速计算方法，并进一步将算法扩展为可以处理非曼哈顿型悬浮导体。最后讨论了积分距离参数对精确度和计算时间的影响，并提出了一种自动确定该参数的策略来平衡准确度和程序运行时间。

第5章 统一的多介质预刻画技术

为了使得悬浮随机行走（Floating Random Walk，FRW）算法可以计算多介质结构，首先需要预计算包含多层介质的转移立方体的转移概率和权值表，这一过程我们称之为介质预刻画。针对平板显示（Flat Panel Display，FPD）结构的特点，本章提出了一种适用于 FPD 设计的统一的多介质预刻画技术。5.1 节首先介绍了 VLSI 电路设计领域中两种已有的多介质预刻画方法，并讨论了这两种方法不适用于 FPD 结构的原因，5.2 节介绍了我们提出的统一的多介质预刻画技术，5.3 节用实验结果证明了我们提出的方法的优越性。

5.1 已有方法的介绍

基于包含两层介质转移立方体的表面格林函数的数值特性，[16]中提出了一种可以处理多介质结构的方法，其主要思想为：对一个给定的介质配置，对所有的包含两层介质的转移立方体进行预刻画，并将预刻画的数据存储起来，在随机行走过程中再使用。但利用这种方法的 FRW 算法每次跳转最多只能跨越一个介质交界面，而在 VLSI 工艺中，通常会有 10 层甚至更多的介质层，这会使得每次随机行走需要的跳转数很大，所以这种方法效率不高。[26]提出了一个改进方法，主要思想是对包含 3 或 4 层介质层的转移立方体进行预刻画，这样在随机行走过程中，每次跳转最多就可以跨越 3 个介质交界面，这样就减小了每次随机行走的跳转数，加快了算法终止速度，但这种方法会大大增加需要预刻画的转移立方体种类（不同的介质配置），也会大大增加程序的内存使用。以上方法还有一个明显的缺点就是当多层介质工艺改变时，需要重新进行多介质预刻画过程，在 VLSI 电路设计领域中，多层介质工艺相对固定，这一点还可以接受，但在 FPD 设计中，多层介质工艺多种多样，每次都重新进行多介质预刻画的方法明显不可取。

另外一种介质预刻画方法叫做介质均匀化方法[25]，其主要思想为将包含任意层介质层的转移立方体用包含 4 层等效介质层（每层厚度相同）的转移立方体来近似，图 5.1 为一个包含 5 层介质层的结构，并展示了各种不同的转移立方体和多介质预刻画技术。当使用介质均匀化方法时，可以使用蓝色的框的转移立方体，而若使用[16]中的方法，则只能使用更小的红线框转移立方体。

首先介绍如何对包含 4 层等效介质的转移立方体进行预刻画。假设 4 层等效介质的相对介电常数分别为：$\bar{\varepsilon}_1$，$\bar{\varepsilon}_2$，$\bar{\varepsilon}_3$，$\bar{\varepsilon}_4$，由于该转移立方体的转移概率和权值只

与 4 个介电常数的比值有关，与其具体数值大小无关，所以转移立方体的介质配置（$\bar{\varepsilon}_1$, $\bar{\varepsilon}_2$, $\bar{\varepsilon}_3$, $\bar{\varepsilon}_4$）可以等效看作为($\bar{\varepsilon}_1/\bar{\varepsilon}_{max}$, $\bar{\varepsilon}_2/\bar{\varepsilon}_{max}$, $\bar{\varepsilon}_3/\bar{\varepsilon}_{max}$, $\bar{\varepsilon}_4/\bar{\varepsilon}_{max}$)，其中$\bar{\varepsilon}_{max}$ = max{$\bar{\varepsilon}_1$, $\bar{\varepsilon}_2$, $\bar{\varepsilon}_3$, $\bar{\varepsilon}_4$}。因此，对于预刻画过程，我们可以只考虑以下的介质配置：4 个介质中，其中一个介电常数为 1，其余 3 个介电常数值在(0,1]区间内。由于不可能遍历所有的介质配置，于是需要对介质配置进行采样，令采样步长为 t，采样区间为[s, 1]，s>0。对于 VLSI 电路设计领域的电容提取问题，相邻介质层的介质比通常不会大于 2，所以 s 取 0.5 是比较合理的，这种情况下采样的介质配置数量为 $4\times[(1-s)/t+1]^3$，常数 4 是因为($\bar{\varepsilon}_1$, $\bar{\varepsilon}_2$, $\bar{\varepsilon}_3$, $\bar{\varepsilon}_4$)中任意一个都可能为 1，又根据对称性，介质配置(1, a, b, c)和(c, b, a, 1)其实是等效的，所以介质配置的采样数减少为 $2\times[(1-s)/t+1]^3-[(1-s)/t+1]^2$，其中减去$[(1-s)/t+1]^2$ 是因为介质配置(1, 1, b, c)被计算了两次。

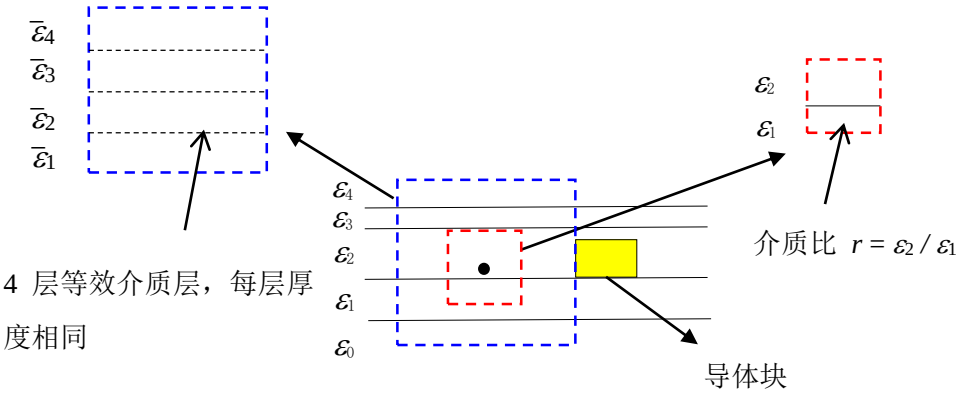


图 5.1 不同的转移立方体和介质预刻画技术

接下来我们进一步讨论使用介质均匀化方法时预刻画数据的大小。根据[16]，对于一个介质配置，需要 $4\times6N^2$ 个浮点数来存储概率密度和权值，其中 N 是转移立方体边的分段数。因此，介质均匀化方法需要存储

$$Size_{DHM} = 48N^2[\frac{1-s}{t}+1]^3 - 24N^2[\frac{1-s}{t}+1]^2, \qquad (5\text{-}1)$$

个浮点数。若 N=31，s=0.5，t=0.05，并且使用单精度浮点数，则预刻画数据大小大约为 238MB，在程序运行时，需要加载这部分数据到内存中。

介质均匀化方法除了使得程序运行速度更快外，另一个好处是不依赖于特定的多层介质工艺，当多层介质工艺改变时，不需要重新进行预刻画过程。然而，这种方法有两个缺点：一是误差问题，由于这种方法使用 4 层等效介质来近似实际的介质层，所以不可避免的会引入误差，如[26]中所提到的，当介质层数很多时，这种方法可能会引入较大的误差；二是内存使用问题，这个缺点主要是在处理 FPD 结构时会遇到，在 FPD 结构中，由于空气介质（介电常数近似为 1）的存在，相邻介质层的介质比可能大于 2，这意味着参数 s 的值需要设置的更小，例如设置为

0.1，根据公式（5-1），可以计算出预刻画数据大小大约为 1.22GB，即算法至少需要 1.22GB 的内存，这会限制算法在某些场合的应用。

5.2 新的方法

为了满足针对 FPD 结构的电容求解需求，我们提出了一种新的方法，这种方法具有预刻画数据量少、精度高、内存消耗低和不依赖特定多介质工艺的优点。其主要思想是：首先列举出一系列含两层介质的转移立方体区域的介质配置，对它们逐个计算其转移概率和权值数据并存储起来；然后，在对一个特定的 FPD 结构进行电容仿真时，先从预计算数据中读取与当前结构的多介质工艺有关的一些数据；在执行随机行走电容计算时，限制每一步跳转所用的转移立方体最多只能包含两层介质，利用线性插值的方法和已读取的多介质预刻画数据得到这个含两层介质的转移立方体的转移概率和权值数据，从而实现随机行走的每步跳转。如图 5.1 所示，我们需要考虑不同的两层介质配置（ε_1, ε_2）和介质分界面位置，根据等效性和对称性，我们只需要考虑两层介质配置为（1, r）的情形，其中 $0<r\leq 1$。当没有歧义时，下文使用 r 代表两层介质配置（1, r）。由于不可能枚举所有两层介质配置，所以需要对 r 进行采样。令 s 代表 r 的最小值，若使用等步长采样，则采样数 $n=(1-s)/t+1$，其中 t 为采样步长，由于介质分界面位置有 N-1 个，则预刻画需要存储的浮点数个数为：

$$Size_{our} = 24N^2(N-1)[\frac{1-s}{t}+1], \qquad (5\text{-}2)$$

当 N=31，s=0.1，t=0.015 时，存储数据大小大约为 177MB，这种情况下 FPD 结构的最大介质比可以达到 10，而为了达到更高的精度，采样步长 t 的值比介质均匀化方法中设置的也更小。

这是一种统一的介质预刻画方法，这意味着预刻画的转移概率和权值数据适用于任何的介质配置。若相邻介质层的介质比 r 处于两个采样值 r_i 和 r_{i+1},之间，则需要用到线性插值方法，令 V_r 两层介质配置 r 的转移概率，则有

$$V_r = V_{r_i}\frac{r-r_i}{r_{i+1}-r_i} + V_{r_{i+1}}\frac{r_{i+1}-r}{r_{i+1}-r_i}。 \qquad (5\text{-}3)$$

尽管预刻画的数据大小超过 100MB，但对于一个给定的多层介质导体结构，并不需要将所有的预刻画数据都加载到内存中。例如，若给定结构的多层介质层介电常数分别为（4, 3.2, 4, 1），其对应的介质比为 0.8 和 0.25，则我们只需要加载介质配置（1.0, 0.790）、（1.0, 0.805）、（1.0, 0.235）和（1.0, 0.250）的预刻画数据，数据大小仅为 11MB。这是相对于介质均匀化方法的一个优点。

这种方法的缺点是计算速度慢，因为转移立方体被限制为只能跨两层介质层。为了减少运行时间，有一种方法是与介质均匀化方法（s=0.5）结合，其主要思想为：在随机行走过程中，首先利用介质均匀化方法可以得到尽可能大的转移立方体，但当转移立方体中出现介质比大于 2 的情况，则使用我们提出的跨两层介质转移立方体。这是一种内存和运行时间折衷的方法，但由于介质均匀化过程，该方法仍然有可能引入较大误差。

5.3 实验结果与分析

本章基于 RWCap2[18]实现了我们提出的算法，并用 TechGFT[16]程序得到了各种预刻画方法所需的预刻画数据。本章所用的 3 个测试用例描述如下：

测例 1：包含 1423 块导体，2 层金属层，高度分别为 70nm 和 220nm；4 层介质层，介电常数分别为 4.0、3.2、4.0 和 1.0，图 5.2 展示了其工艺介绍横截面。

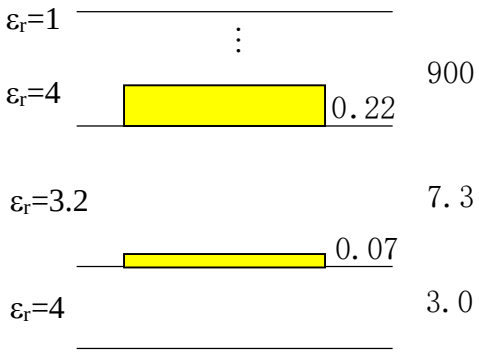


图 5.2 测例 1 工艺技术横截面示意，介质层厚度和金属高度单位都是μm

测例 2：包含 11 块导体，2 层金属层，高度都为 100nm；3 层介质层，介电常数分别为 4.0、3.5 和 7.0。

测例 3：包含 808 块导体，4 层金属层，高度分别为 340nm、220nm、400nm 和 70nm；8 层介质层，介电常数分别为 3.9、6.5、3.5、6.5、4.2、3.2、4.0 和 1.0。

根据不同的多介质预刻画方法，将测试算法进行如下分类：

FRW-2：使用[16]中提到的多介质预刻画方法。

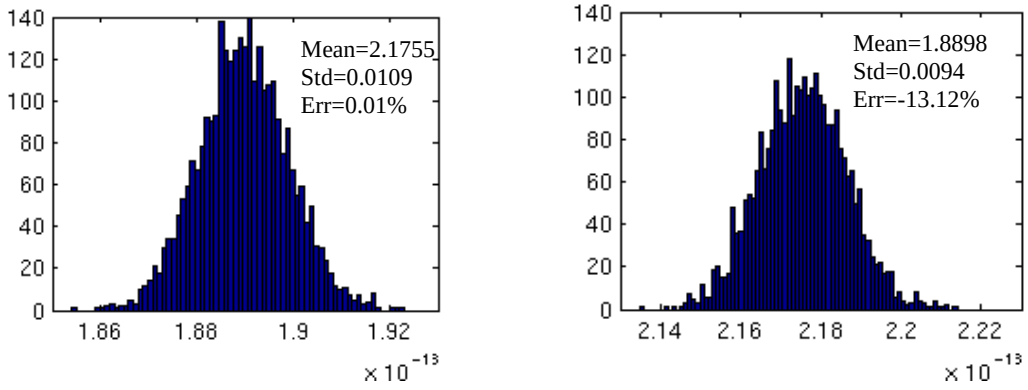
FRW-2unify：使用 5.2 节中我们提出的统一的多介质预刻画方法，相关参数为 N=31，s=0.1，t=0.015。

FRW-mixed：将我们提出的方法和介质均匀化方法结合起来，我们提出的方法的相关参数与 FRW-2unify 中的相同，介质均匀化方法的相关参数为 N=31，s=0.5，t=0.05。

我们以 FRW-2 得到的电容结果作为标准值，比较 FRW-2unify 和 FRW-mixed

处理多介质结构的结果，为了避免随机误差，对每个测试用例，每种算法都运行 3000 次，并取平均值作为其电容结果，对于测例 1 和测例 2，两种方法的误差都在 0.03%以内，不列出其具体数据，对于测例 3，图 5.3 描绘了其运行 3000 次的电容分布图，可以看出，两次方法的电容分布都近似为正态分布，FRW-2unify 和 FRW-mixed 与 FRW-2 的相对误差分别为 0.01%和-13.13%，这说明包含介质均匀化过程的 FRW-mixed 方法会引入较大误差。

表 5.1 展示了 3 种 FRW 算法的运行时间和内存使用数据，可以看出 FRW-2unify 和 FRW-mixed 的运行时间几乎一样，对于测例 1 和测例 2，FRW-mixed 比另外两种方法慢，但对于测例 3，FRW-mixed 速度却快很多，这是因为测例 1、2 介质层数少，而测例 3 介质层数多，所以介质均匀化起到的加速作用更大。另外，FRW-mixed 方法消耗的内存是其他两种方法的 10 多倍，而且如图 5.3 所示，可能会引入较大误差。FRW-2unify 和 FRW-mixed 的多介质预刻画数据大小分别为 177MB 和 415MB，所以很明显，FRW-2unify 比 FRW-mixed 更好，并且比 FRW-2 方法更适合于多介质工艺多样化的 FPD 设计领域。



(a) (b)

图 5.3 FRW-2unify 和 FRW-mixed 分别运行 3000 次的电容分布图
(a) FRW-2unify, (b) FRW-mixed

表 5.1 FRW-2，FRW-2unify 和 FRW-mixed 方法的运行时间和内存使用

测例	FRW-2		FRW-2unify		FRW-mixed	
	时间(s)	内存(MB)	时间(s)	内存(MB)	时间(s)	内存(MB)
1	2.3	9.6	2.4	12.4	2.8	250.9
2	538.9	5.7	530.2	11.2	629.0	249.6
3	221.5	21.0	227.1	34.7	40.3	273.2

5.4 本章小结

为了满足 FPD 设计领域的电容求解需求，本章提出了一种新的统一的多介质预刻画方法，克服了已有的两种多介质预刻画方法的缺点，新的方法具有精度高、内存使用少和不依赖特定多介质工艺的优点，实验结果同样证明了这种方法的优越性。

第6章 大规模并行计算

为了设计高质量的平板显示（Flat Panel Display，FPD）设备，需要电容求解器能够提供更为精确的耦合电容值，但悬浮随机行走（Floating Random Walk，FRW）算法设置的精度越高，算法运行时间就越长。为了克服这一问题，本章提出了一种基于 MPI 的 FRW 集群并行算法，利用集群强大的硬件环境，我们可以获得极高的并行加速比，从而使得算法运行时间大大降低。6.1 节介绍了基于 MPI 的 FRW 集群并行算法的基本思路，6.2 节对其进行了优化改进，6.3 节给出了实验结果，我们提出的这种算法在使用 120 个计算节点时可以达到 113 倍的加速比。

6.1 算法基本思路

由于每次随机行走都是互相独立的，所以 FRW 算法非常适合并行计算，[16]已经利用 pthread 接口实现了单机并行 FRW 算法，并且取得了良好的并行加速比。但是，单机 FRW 算法的并行加速比受单机条件限制，加速效果有限，而若使用大规模集群来实现并行 FRW 算法，无疑可以得到更大的并行加速比。图 6.1 展示了集群并行 FRW 算法的流程图，首先用 MPI_init 接口初始化 MPI 运行环境，然后

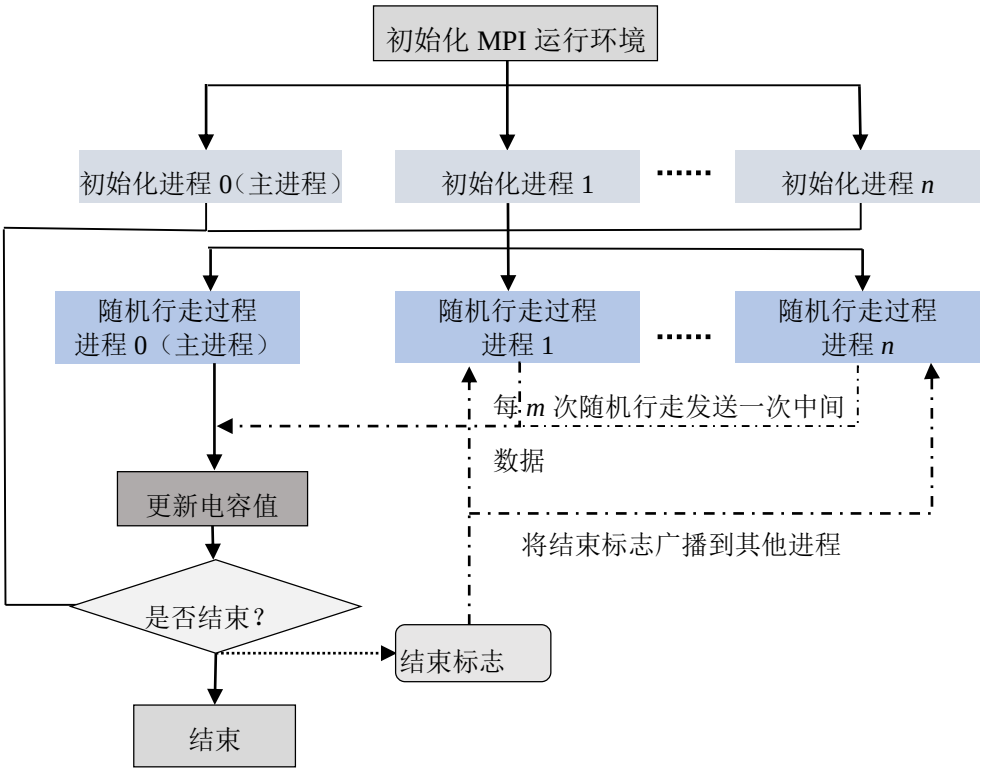


图 6.1 集群并行 FRW 算法流程图

对每个 MPI 进程进行初始化，包括读取输入数据，加载 GFT 和 WVT 表数据，初始化过程结束后，每个进程开始独立的执行随机行走过程，令进程 0 作为主进程，每个进程每执行 m 次（m 一般为 1000）随机行走后会发送一次中间数据，主进程负责收集这些中间数据，然后更新电容值并判断程序是否达到终止条件，然后将程序是否结束的标志广播发送给其他各个进程。

6.2 算法优化

前一节中描述的算法有一个发送中间数据的过程，当程序终止所需的随机行走数较多时，进程间通信和主进程处理中间数据的过程会影响程序的运行时间，从实验结果（见 6.3.2 节的图 6.4）可以看出，该算法在 MPI 进程数较多时并行加速比并不能达到预期值，为了解决这个问题，需要减少进程间通信次数。

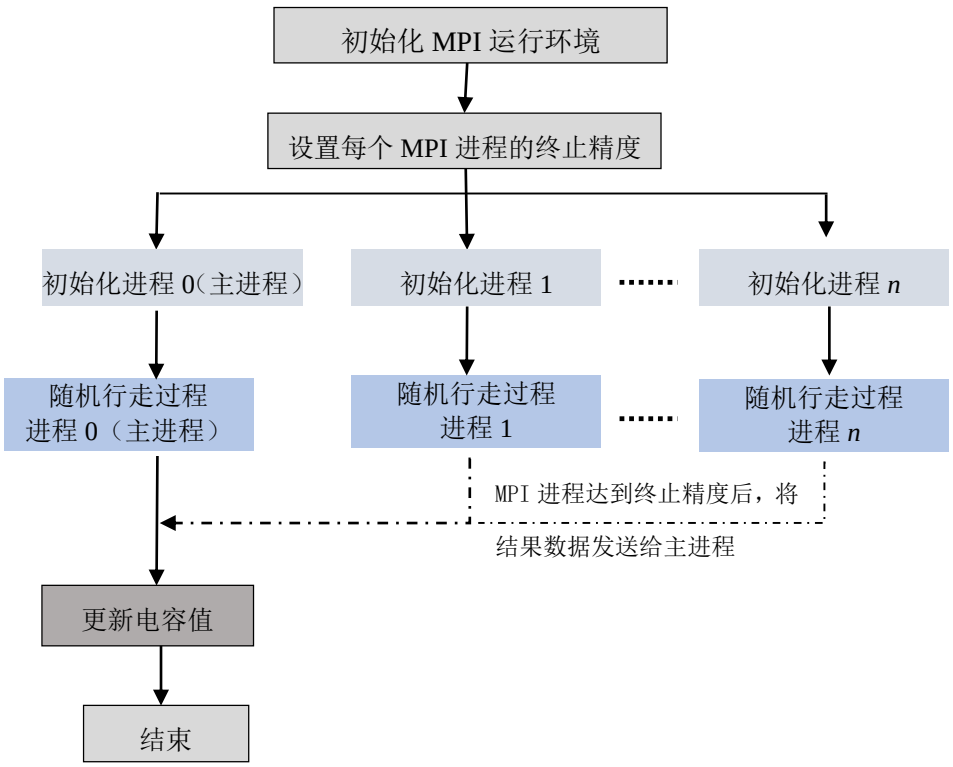


图 6.2 优化后集群并行 FRW 算法流程图

回顾式（2-15），可以发现 FRW 算法的精度（即估计误差）与随机行走次数的平方根成反比，即有

$$err \propto \frac{1}{\sqrt{N_{walk}}}, \tag{6-1}$$

其中，N_{walk} 表示总的随机行走数，err 为估计误差。若有 $N_{process}$ 个 MPI 进程，将

N_{walk} 次随机行走均分给 $N_{process}$ 个 MPI 进程，则每个 MPI 进程需要执行 $N_{walk}/N_{process}$ 次随机行走。所以当程序要求的精度为 err，且有 $N_{process}$ 个 MPI 进程并行执行时，每个进程需要达到的精度应该为

$$err' = \sqrt{N_{process}} \cdot err, \tag{6-2}$$

因此，可以事先为每个 MPI 进程设置终止条件，当进程达到终止条件后，将计算结果发送给主进程，主进程收集这些数据并更新最后的电容值，于是，进程间只需要进行一次通信，大大提高了程序的运行效率。图 6.2 展示了优化算法的流程图。

以上的优化方法有一个假设前提，即所有计算节点的计算性能完全相同，所以是将总随机行走数均分给各个 MPI 进程。当计算节点的计算性能不同时，若仍采样上述策略进行任务分配，会导致计算性能好的节点空闲等待，浪费资源。为了解决这一问题，可以为不同性能的节点分配不同的随机行走数，假设有 n 个计算节点，每个计算节点对应一个 MPI 进程，n 个节点计算性能分别为 $t_1, t_2, \ldots t_n$，计算性能的计量标准为执行指定次数的随机行走所需花费的时间，则第 j 个进程应执行

$$N_{walk}^{j} = \frac{\frac{1}{t_j}}{\sum_{i=1}^{n}\frac{1}{t_i}} \cdot N_{walk}, \tag{6-3}$$

次随机行走，进一步可以得到进程 j 需要达到的精度为

$$err_j' = \sqrt{\frac{\sum_{i=1}^{n}\frac{1}{t_i}}{\frac{1}{t_j}}} \cdot err, \tag{6-4}$$

按照式（6-4）为不同性能的计算节点设置不同的终止条件，可以充分利用计算资源，使进程间等待时间最短。

6.3 实验结果与分析

基于 RWCap2[18]，我们实现了本章提出的集群并行 FRW 算法，本章所使用的 3 个测试用例与第 5 章的 3 个测试用例完全相同，实验终止条件均设为主电容精度 0.1%。

6.3.1 集群实验环境介绍

实验中所使用的集群是清华大学信息科学与技术国家实验室提供的“探索 100”集群，“探索 100”百万亿次集群系统由登录节点、740 个计算节点、24 个 I/O 存

储节点及其他管理节点组成，节点间通过 InfiniBand 网络互连。集群系统理论峰值浮点计算性能达到 104TFlops，存储总容量 1000TB。集群中计算节点均采用两个 Intel Xeon X5670 六核处理器（2.93GHz，12MB Cache）, 160G SATA 硬盘。 740 个节点中， 370 个节点配置 32GB 内存，另外 370 个节点配置 48GB 内存，每个节点都是一个多核 SMP 服务器，计算节点用于运行串行和并行计算任务，支持 MPI、OpenMP 及 MPI/OpenMP 混合并行编程模式。“探索 100”作业管理系统以 CPU 核作为并行作业的资源分配单位，实现并行作业的调度运行。“探索 100”每个计算节点为 12 核的 SMP 服务器,可以最大支持 740* 12 = 8880 核并行作业的计算。

IO 存储系统由 2 个管理节点 MDS0、MDS1 和 22 个 IO 存储节点 OSS1~OSS22 构成，提供集群系统的全局系统数据存储，可提供在线提供 160TB 存储容量。存储系统采用 LUSTRE 并行文件系统进行管理，实测写带宽 4GB/s。所有用户目录下/WORK 目录为全局共享，所有节点/WORK 目录都有读写权限。

“探索 100”由 InfiniBand QDR 通信网络构成，理论带宽 40Gb。所有节点间均可以通过 InfiniBand 网络实现高速通信。支持 MPI 并行任务间通信，并实现全局文件系统的数据传输。

“探索 100”百万亿次集群系统所有节点均采用 RedHat Enterprise Linux 5.5 x86_64 版本，遵循 POSIX，LSB 等标准，提供了 64 位程序开发与运行环境。

6.3.2 实验结果

“探索 100”集群中的计算节点的性能都相同，所以首先测试将总随机行走数均分给各个节点的算法。

表 6.1 中给出了优化前的并行 FRW 算法（对应图 6.1 的算法流程）结果，可以看到对于测例 3，当 MPI 进程数为 120 时，并行加速比只有 23.4 倍，图 6.3 展示了优化前并行 FRW 算法 MPI 进程数与并加速比的关系,可以发现优化前的并行 FRW 算法的加速效果非常差，当 MPI 进程数达到某个值时，并行加速比基本不再继续增加，这也说明了 MPI 通信和主进程中间数据处理过程给程序的运行效率带来了很大的影响。

表 6.1 优化前并行 FRW 算法结果

MPI 进程数	测例 1		测例 2		测例 3	
	时间(s)	加速比	时间(s)	加速比	时间(s)	加速比
1	7452	1.0	3661	1.0	12904	1.0
12	644	11.6	363	10.1	1425	9.1
24	349	21.4	217	16.9	905	14.3
36	259	28.8	163	22.5	747	17.3
48	200	37.3	144	25.4	678	19.0
60	170	43.8	124	29.5	676	19.1
72	150	49.7	111	33.0	647	19.9
84	138	54.0	108	33.9	590	21.9
96	129	57.8	102	35.9	588	22.0
108	152	49.0	99	37.0	516	25.0
120	111	67.1	96	38.1	551	23.4

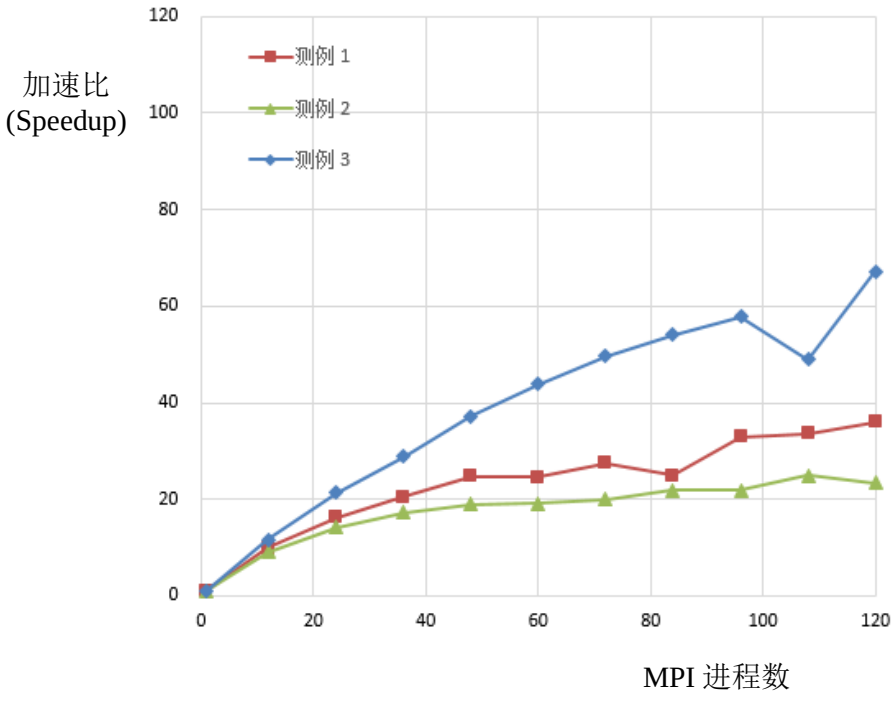


图 6.3 优化前并行 FRW 算法 MPI 进程数与并加速比的关系

表 6.2 中给出了优化后的并行 FRW 算法（对应图 6.2 的算法流程）结果，可以发现并行加速比明显比之前要增加很多，对于测例 3，当 MPI 进程数为 120 时，并行加速比达到了 113.6 倍，已经非常符合预期值。另外图 6.4 展示了优化后并行 FRW 算法 MPI 进程数与并加速比的关系，可以看到此时的 MPI 进程数与并行加速比基本呈线性增长关系，而且根据曲线趋势来看，当 MPI 进程数大于 120 时，并行加速比还可以继续增长。

表 6.2 优化后并行 FRW 算法结果(所有计算节点性能相同)

MPI 进程数	测例 1		测例 2		测例 3	
	时间(s)	加速比	时间(s)	加速比	时间(s)	加速比
1	5937	1.0	3668	1.0	12838	1.0
12	621	9.6	321	11.4	1074	12.0
24	307	19.3	162	22.6	554	23.2
36	205	29.0	107	34.3	362	35.5
48	157	37.8	81	45.3	278	46.2
60	125	47.5	65	56.4	221	58.1
72	106	56.0	55	66.7	187	68.7
84	91	65.2	47	78.0	158	81.3
96	80	74.2	41	89.5	141	91.1
108	72	82.5	37	99.1	124	103.5
120	65	91.3	33	111.2	113	113.6

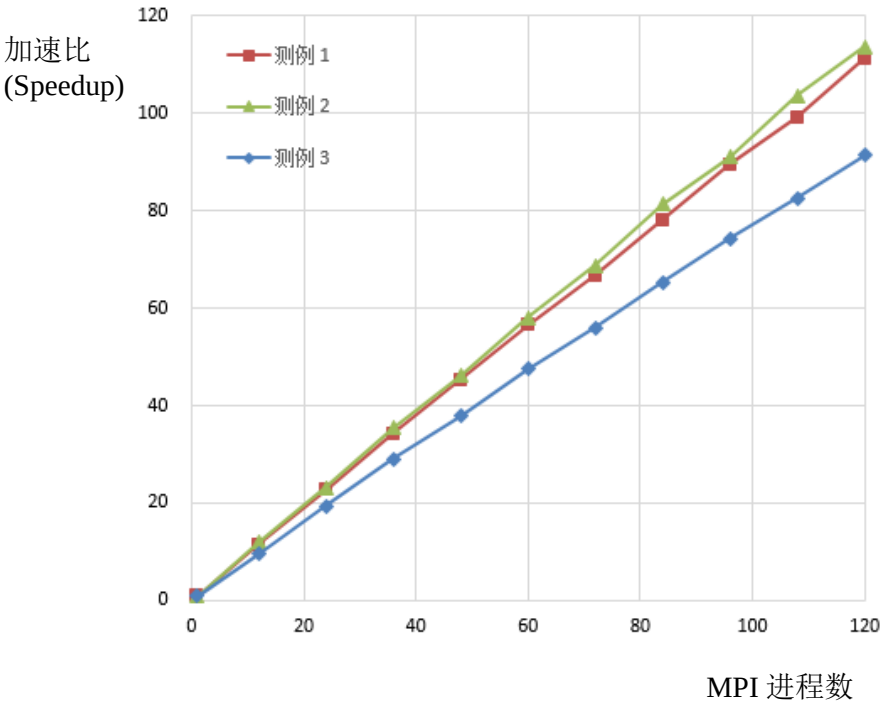


图 6.4 优化后并行 FRW 算法 MPI 进程数与并加速比的关系

然后再测试计算节点性能不同时情况，我们将均分总随机行走数给各个计算节点的方法称为优化算法 1，根据不同节点性能来分配随机行走数的方法称为优化算法 2，使用的实验环境为本课题组的两台 Linux 服务器，服务器 104 和服务器 106。服务器 104 配置为 Intel Xeon E5-2630 2.3GHz CPU，24 核；服务器 106 配置为 Intel Xeon E5-2630 2.4GHz CPU，32 核。选取一个包含 247 块导体、8 层介质层的结构

作为测试用例，表 6.3 展示了实验结果，其中 MPI 进程是均分给两台服务器的，可以看出使用优化算法 2 最大比优化算法 1 快了 22.5%。

表 6.3 计算节点性能不同时实验结果

MPI 进程数	优化算法 1	优化算法 2	加速效果
	时间(s)	时间	
2	113.8	101.89	10.5%
4	55.57	49.39	11.1%
8	30.78	28.1	8.7%
16	19.99	18.12	9.4%
32	13.32	10.32	22.5%

6.4 本章小结

本章主要提出了一种可以在集群上运行的并行 FRW 算法，并在清华大学“探索 100”集群上进行了实验，最初的算法得到的并行加速比并不理想，经过分析，发现性能瓶颈在 MPI 通信和主进程处理中间数据上，于是提出了改进算法，实验证明改进后的算法得到了与预期接近的并行加速比。在该集群上用 120 个 MPI 进程时达到了 113 倍的加速比，这样的加速效果在目前单机上很难达到，而且随着 MPI 进程数的增加，并行加速比还可以继续增长。

第7章 总结与展望

悬浮随机行走算法在 VLSI 电路设计领域取得了广泛的应用，但在平板显示设计领域中遇到了一些新的挑战，包括非曼哈顿型导体和悬浮导体的处理、多介质工艺的多样性特点和高耦合电容精度要求，针对这些挑战，本文一一给出了有效的解决办法。

针对平板显示结构中包括非曼哈顿型导体的情况，首先提出一种用额外空间网格结构管理非曼哈顿导体的改进随机行走空间管理技术，它不需考虑非曼哈顿导体间的遮挡关系，虽然实现简单，但实验表明该方法计算速度快，比不使用空间管理的算法最大快了 62 倍；之后提出了一种分组遮挡判断法，改进了已有的非曼哈顿导体遮挡关系的判断算法，使得可处理非曼哈顿型导体的空间管理效率更高，实验表明该方法比不使用空间管理的算法最大快了 84 倍，而且其使用的内存比使用额外空间网格结构的方法更少。将这两种方法与曼哈顿转移立方体和旋转转移立方体两种技术结合，开发出了四种高效率的随机行走电容求解算法，并通过实验比较了这四种高效算法的优劣，另外，还与快速边界元法作了对比，反映出所提出算法相对于快速边界元法在运算时间和内存用量上的巨大优势。

针对实际工艺中存在的悬浮导体，本文实现了一种针对曼哈顿悬浮导体的快速计算方法，实验表明该方法相对已有方法具有速度快、精度高的优点。之后，进一步将算法扩展到可以处理非曼哈顿型悬浮导体，并仍然能够获得较高的精确度。最后，根据实验研究，提出了一种自动确定积分面距离参数的策略来平衡准确度和程序运行时间。

本文还提出了一种统一的多介质预刻画技术，主要是针对平板显示设计中的多介质工艺多样性的特点。实验表明，与已有的两种方法相比，我们提出的这种方法具有预刻画数据量少、精度高、使用内存小和不依赖特定多介质工艺的优点。

针对高耦合电容精度要求，本文将单机并行实现的悬浮随机行走算法扩展到了大规模集群并行，并利用清华大学提供的“探索 100”集群进行了实验。首先提出的是一种进程间通过定时的通信来判断算法是否终止，但实验结果表明该算法并行加速比低，且并行进程数增加时，加速比并不随之增长，经过分析， 提出了一种改进的算法，在进程执行随机行走之间就先确定各个进程的终止条件，中间不需要再进行通信，减少了 MPI 进程之间的通信，大大提高了程序效率，实验表明，该算法的并行加速比与并行进程数近似成正比。另外，我们还针对计算节点性能不同的情况做了优化，为每个进程设置了不同的终止条件，实验表明，优化

后的算法比之前在速度上有大约 22.5%的提升。

在本文的工作基础上，仍然还有很多需要改进和进一步研究的地方。例如针对非曼哈顿型悬浮导体，目前采用的是一种近似的方法，理论上存在误差。第 5 章提出的统一的多介质预刻画方法在计算速度上仍有改进余地。另外，现有的算法还不能有效的处理很多实际工艺中包含的保形介质结构。因此如何进一步改进随机行走算法也是未来可以继续研究的内容。

基于本文的研究工作，我们开发了专门针对平板显示设计的软件包 FPDCap，并在华大九天公司得到了实际应用，反馈效果良好。

参考文献

[1] Pak H., Kim S., Kim S., Jo Y., Kim S. and McCartney R. I. 2008. Electrical models of TFT-LCD panels for circuit simulations. Journal of the Society for Information Display, 16 (2008), 509-515.

[2] Son Y. S. and Cho G. H. 2008. Design considerations of channel buffer amplifiers for low-power area-efficient column drivers in active-matrix LCDs. IEEE Trans. Consumer Electronics, 54 (Feb. 2008), 648-656.

[3] Egan J. 2014. Resistive vs. Capacitive Touchscreens. Available: http://blog.junipersys.com/resistive-vs-capacitive-touchscreens/

[4] Takagi M., Yamaguchi K., Chida H. et. al. 2012. Layout and reticle verification for FPD. In Proc. SPIE 84410M (2012), 1-4.

[5] Uchida Y., Tani S., Hashimoto M., et. al 2005. Interconnect capacitance extraction for system LCD circuits. In Proc. the 15th ACM Great Lakes Symposium on VLSI (GLSVLSI), (2005). 160-164.

[6] Nabors K. and White J. 1991. FastCap: A multipole accelerated 3-D capacitance extraction program. IEEE Trans. Computer-Aided Design, 10 (Nov. 1991), 1447–1459.

[7] Yu W. and Wang Z. 2004. Enhanced QMM-BEM solver for three-dimensional multiple-dielectric capacitance extraction within the finite domain. IEEE Trans. Microwave Theory Tech., 52 (Feb. 2004), 560–566.

[8] Yan S., Sarin V., and Shi W., “Fast 3-D capacitance extraction by inexact factorization and reduction,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 25(10): 2282-2286, Oct. 2006

[9] Yu W., Zhao C., Yang S., and Lu T., "The application of boundary element method to the resistance calculation problem in designing flat panel displays," Journal of the Society for Information Display, 24(3): 177-186, 2016

[10] Yu W., Zhang M. and Wang Z., "Efficient 3-D extraction of interconnect capacitance considering floating metal-fills with boundary element method," IEEE Trans. Computer-Aided Design, Vol. 25, No. 1, pp. 12-18, Jan. 2006.

[11] Chai W., Jiao, D. and C.-K. Koh, “A direct integral-equation solver of linear complexity for large-scale 3D capacitance and impedance extraction,” in *Proc. Design Automation Conference*, Jul. 2009, pp 752-757.

[12] Cai W. and Jiao D., “Linear-complexity direct and iterative integral equation solvers accelerated by a new rank-minimized H2-representation for large-scale 3-D interconnect extraction,” IEEE Trans. Microwave Theory Tech., 61(8): 2792-2805, Aug. 2013.

[13] Synopsys Inc.. Raphael: 2D, 3D resistance, capacitance and inductance extraction tool.Available: http://www.synopsys.com/Tools/TCAD/InterconnectSimulation/Pages/Raphael.aspx

[14] Chen G., Zhu H., Cui T., Chen Z., Zeng X., and Cai W., “ParAFEMCap: A parallel adaptive finite-element method for 3-D VLSI interconnect capacitance extraction,” *IEEE Trans. Microw. Theory Techn.*, vol. 60, no. 2, pp. 218-231, 2012.

[15] Le Coz Y. and Iverson R. B. 1992. A stochastic algorithm for high speed capacitance extraction in integrated circuits. Solid-State Electronics, 35 (Jul. 1992), 1005–1012.

[16] Yu W., Zhuang H., Zhang C., Hu G. and Liu Z. 2013. RWCap: A floating random walk solver for 3-D capacitance extraction of very-large-scale integration interconnects. IEEE Trans. Computer-Aided Design, 32 (Mar. 2013), 353–366. Available: http://learn.tsinghua.edu.cn:8080/2003990088/rwcap.htm

[17] Zhang C. and Yu W. 2013. Efficient space management techniques for large scale interconnect capacitance extraction with floating random walks. IEEE Trans. Computer-Aided Design, 32 (Oct. 2013), 1633–1637.

[18] Yu W., “RWCap2: Advanced floating random walk solver for the capacitance extraction of VLSI interconnects,” in Proc. ASICON, Oct. 2013, pp. 162-165.

[19] Zhang C. and Yu W., “Efficient techniques for the capacitance extraction of chip-scale VLSI interconnects using floating random walk algorithm,” in Proc. ASP-DAC, Jan. 2014, pp. 756-741.

[20] El-Moselhy T. A., Elfadel I. M., and Daniel L., “A hierarchical floating random walk algorithm for fabric-aware 3-D capacitance extraction,” in *Proc. ICCAD*, Nov. 2009, pp. 752–758.

[21] Brambilla A. and Maffezzoni A., “A statistical algorithm for 3-D capacitance extraction,” IEEE Microw. Guided Wave Lett., 10: 304–306, 2000.

[22] Bansal N., “Randomized algorithms for capacitance estimation,” Indian Institute of Technology, Bombay, Tech. Rep., Apr. 1999.

[23] Bernal F., Acebron J., and Anjam I., “A stochastic algorithm based on fast marching for automatic capacitance extraction in non-Manhattan geometries,” SIAM J. Imaging Sciences, 7(4): 2657-2674, 2014.

[24] Zhang C., Yu W., Wang Q. and Shi Y. 2015. Fast random walk based capacitance extraction for the 3-D IC structures with cylindrical inter-tier-vias. IEEE Trans. Computer-Aided Design, 34(Dec. 2015), 1977-1990.

[25] Rollins G. 2010, Rapid3D 20X performance improvement. Available: http://www.synopsys.com/Community/UniversityProgram/Pages/Presentations.aspx

Other limitations” of Chap. 4)

[26] Zhang B., Yu W. and Zhang C. 2015. Improved pre-characterization method for the random walk based capacitance extraction of multi-dielectric VLSI interconnects. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, DOI: 10.1002/jnm.2042.

[27] Xu Z., Zhang C., and Yu W., "Floating random walk based capacitance extraction for general non-Manhattan conductor structures," IEEE Trans. Computer-Aided Design, 36(1): 120-133, 2017

[28] Xu Z., Yu W., Zhang C., Zhang B., Lu M., and Mascagni M., "A parallel random walk solver for the capacitance calculation problem in touchscreen design," in Proc. Great Lake Symposym of VLSI , Boston, USA, May 2016, pp. 99-104.

[29] Yu W., Zhang B., Zhang C., Wang H., and Daniel L., "Utilizing macro models in floating random walk based capacitance extraction," in Proc. IEEE DATE , Dresden, Germany, Mar. 2016, pp. 1225-1230.

[30] 张超，面向复杂大规模互连结构的悬浮随机行走电容提取算法[硕士学位论文]，北京：清华大学计算机科学与技术系，2015

[31] 黎波，考虑悬浮哑元与触摸屏结构的随机行走电容提取算法研究[学士学位论文]，北京：清华大学计算机科学与技术系，2015

[32] Magma Inc., *QuickCap and QuickCap NX Technology Guide*, 2009. (see Section “Other limitations” of Chap. 4)

[33] Batterywala, Shabbir, et al. "A statistical method for fast and accurate capacitance extraction in the presence of floating dummy fills." VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on. IEEE, 2006.

[34] Hammersley J M, Handscomb D C. Monte carlo methods, volume 1. Methuen London, 1964.

[35] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[36] Yu W. and Wang X., Advanced Field-Solver Techniques for RC Extraction of Integrated Circuits, Springer Inc., 2014.

[37] A. N. Bhoj, R. V. Joshi, and N. K. Jha, “3-D-TCAD-based parasitic capacitance extraction for emerging multigate devices and circuits,” IEEE Trans. Very Large Scale Integr.(VLSI) Syst., 21(11): 2094–2105, Nov. 2013.

致　谢

首先，我要衷心地感谢我的导师喻文健老师，在我攻读硕士的这三年时间里，喻老师不管是生活还是学习上都给了非常大的帮助。在科研工作上，喻老师一直给予我耐心的指导，不管自己有多忙，总是优先给我答疑解惑，使我在这短短的三年时间里学到了非常多的东西。喻老师严谨的科研态度和勤奋的工作态度都给我留下了深刻的印象，这些都将对我以后的工作和学习产生积极的影响。

我还要感谢我在实验室这三年里遇到的师兄师弟们，他们都给予了我很大的支持和帮助，特别是张伯龙师兄和张超师兄，他们帮助我解决了很多工作的难题，让我顺利的完成了很多科研项目，另外黎波师弟的本科毕设工作也给了我很大的帮助。还有秦超师兄、胡君师姐、董超同学、杨飞同学、卢美娟同学和焦凤先同学也一直给了我很大的支持和帮助，感谢他们。另外，还要感谢贾小涛师兄在我第一次出国参加学术会议期间给我的照顾和帮助。

最后，我要感谢我的家人，是他们的支持和鼓励让我如此顺利的完成了学业。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____日　期：_____

个人简历、在学期间发表的学术论文与研究成果

个人简历

1993 年 10 月 21 日出生于江西省樟树市。

2010 年 9 月考入华中科技大学电子科学与技术系电子科学与技术专业，2014 年 7 月本科毕业并获得工学学士学位。

2014 年 9 月通过研究生考试进入清华大学计算机科学与技术系攻读硕士学位至今。

发表的学术论文

[1] **Zhezhao Xu**, Chao Zhang, and Wenjian Yu, "Floating random walk based capacitance extraction for general non-Manhattan conductor structures," *IEEE Trans. Computer-Aided Design*, 36(1): 120-133, 2017 (SCI 收录，CCF A 类期刊)

[2] **Zhezhao Xu**, Wenjian Yu, Chao Zhang, Bolong Zhang, Meijuan Lu and Michael Mascagni, "A parallel random walk solver for the capacitance calculation problem in touchscreen design," in *Proc. Great Lake Symposym of VLSI* , Boston, USA, May 2016, pp. 99-104. (CCF C 类国际会议)

[3] Khalid Al-jabery, **Zhezhao Xu**, Wenjian Yu, Donald C. Wunsch, II, Jinjun Xiong and Yiyu Shi, "Demand-side management of domestic electric water heaters using approximate dynamic programming," *IEEE Trans. Computer-Aided Design*, 36(5): 775-788, 2017 (SCI 收录，CCF A 类期刊)

研究成果

[1] 喻文健，徐哲钊，张伯龙，“一种面向触摸屏电容仿真的多介质预刻画方法”，专利号：201610237910.0 (pending)