面向复杂大规模互连结构的悬浮随机行走电容提取算法

（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：张　超
指 导 教 师：喻　文　健　副　教　授

二〇一五年五月

The Floating Random Walk Capacitance Extraction Algorithm for Very Large Scale and Complicated Interconnect Structures

Thesis Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Master of Science

in

Computer Science and Technology

by

Zhang Chao

Thesis Supervisor : Associate Professor Yu Wenjian

May, 2015

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应遵守此规定）

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘 要

随着现代集成电路工艺的发展，集成电路中互连线的寄生参数对电路的影响越来越明显，甚至超过了门电路本身的影响，这使得准确地提取电路的电容并计算延迟变得越来越重要。只有能准确地提取和模拟寄生参数，才能准确地验证电路的可靠性。基于悬浮随机行走的电容提取算法，作为一种高性能、高精度、内存占用小且易于并行的算法，应用场合越来越广泛。与此同时，日益增大的电路规模、愈来愈复杂的电路结构也为该算法的应用提出了新的挑战。为了使该算法能以更高的效率来适应规模更为庞大、结构更为复杂的电路，本文做出了如下贡献。

- 提出了利用门限距离优化的快速空间管理数据生成技术，以及一种将网格与八叉树混合的空间数据索引结构。相比原有的空间数据生成方法，本文方法在处理超过 10 万导体块的算例时取得 6000 倍以上的加速，能在 20 秒时间内建立包含 50 万导体块的空间管理数据。所提出的混合空间管理结构相比已有结构能使随机行走时间缩短 12%，在相同计算速度的前提下减少一半的内存用量。此外，提出改进的最近导体查询算法也使随机行走过程加速 2 倍。
- 针对全线网电容提取任务，提出虚拟高斯面采样技术，避免了为建立包含整个线网的高斯面所需的复杂几何计算，使得随机行走算法可更方便地用于实际的全线网电容提取与时延计算。此外，提出最优的重要性采样和分层采样等减小方差的策略，将随机行走过程加速 1.5 倍。
- 针对三维芯片和数模混合电路中常见的非曼哈顿结构（如圆柱形硅通孔和倾斜导线等），提出了旋转转移立方体以尽快终止随机行走、适合非曼哈顿形体的空间管理方法、以及针对硅通孔结构的高斯面生成与重要性采样等一系列技术。与将圆形横截面近似为正方形的处理方法相比，本文方法使计算出的硅通孔电容误差缩小 10 倍，而不增加计算时间。相比快速边界元电容提取算法，本文方法显示出高达 214 倍的加速比。这些技术大大提高了随机行走方法处理复杂几何形体的能力。

本文提出的算法有效地改善了悬浮随机行走电容提取算法的效率和适用范围，在保留该算法原有优势的同时，使其能更广泛地应用于各种电容提取问题中。

关键词：悬浮随机行走；电容提取；高斯面采样；空间管理；非曼哈顿结构

Abstract

In modern nanometer process technology integrated circuit (IC) design, extracting the parasitic capacitance and calculating the delay accurately is increasingly important. The floating random walk (FRW) capacitance extraction algorithm is now widely used because of its high efficiency, tunable accuracy, lower memory usage and better parallelism. However, with the increase of the density and complexity of IC, the FRW algorithm is now facing some new challenges. Improvements in three aspects are proposed in this thesis to explore the FRW algorithm's potential of handling very large scale and complicated interconnect structures.

- This thesis proposed an efficient space management structure construction technique with distance limit optimization, and a new grid-octree hybrid structure. Compared with the original construction method, the proposed method achieves more than 6000 times speedup with 100,000 conductors. The hybrid structure reduces 12% of the walk time. The proposed nearest block searching algorithm also speeds up the random walk 2 times.
- For the full-net extraction, this thesis proposed a virtual Gaussian surface sampling technique to avoid the complex geometric operations when constructing a full-net's Gaussian surface. And the proposed importance and strata sampling strategy gives the random walk a 1.5× speedup.
- For the non-Manhattan layouts, like cylindrical through-silicon vias (TSV) or bevel conductors in 3D ICs and digital-analog mixed ICs, this thesis proposed the rotated transition cube, the space management structure for non-Manhattan blocks, and the Gaussian surface construction and sampling technique for TSV. Compared with the method that approximates the circle section with square, the proposed method reduces the error of TSV capacitance 10 times without increasing computational time. Compared with boundary element algorithm, this method gives a 214× speedup.

With the proposed improvements, the floating random walk algorithm can achieve better performance and be applied in a wider range of capacitance extraction applications.

Key words: Floating Random Walk; Capacitance Extraction; Gaussian Surface Sampling; Space Management; Non-Manhattan Geometry

目 录

主要符号对照表

BGS	导体块高斯面 (Block's Gaussian Surface)
EDA	电子设计自动化 (Electronic Design Automation)
FRW	悬浮随机行走 (Floating Random Walk)
MOS	金属氧化物半导体 (Metal Oxide Semiconductor)
MC	蒙特卡洛 (Monte Carlo)
PDF	概率分布函数 (Probability Distribution Function)
IS	重要性采样 (Importance Sampling)
SS	分层采样 (Stratified Sampling)
TSV	硅通孔 (Through Silicon Via)
VGSS	虚拟高斯面采样 (Virtual Gaussian Surface Sampling)

第 1 章 引言

1.1 研究背景

近几年来，集成电路设计和制造工艺不断快速发展，电路中元件的密度呈指数级增长。在这个过程中，随着特征尺寸的减小，互连线间的寄生耦合电容产生的延迟已经逐渐超过了门电路的延迟，成为了制约电路性能和可靠性的重要因素。因此为了进行准确的电路仿真，验证电路的正确性和可靠性，必须要准确地提取互连线间的寄生耦合电容。这意味着对电容提取算法的效率和精度有更迫切的需求。

常见的电容提取算法可以分为两类：模式匹配和场求解器。其中，基于模式匹配的算法虽然效率较高，但是其准确性不能满足日益增长的需要。相比之下，场求解器则拥有较好的精度，能够满足高精度电容提取的需求。

用于电容提取的场求解器可以被分为两类，一类是以有限元算法（FEM）[1,2] 和边界元算法（BEM）[3–8] 等为代表的传统确定型算法。它们通常使用离散化问题空间后建立并求解大型线性方程组的方法。这使得这类算法难以被应用到大规模电路的电容提取问题中，因为其所需的内存和时间随着问题规模增长的速度是非常快的，其复杂度通常是 $O(n^3)$ 或更高。目前这类算法主要应用于针对小型结构得到电容的参考值，用以验证和校准其它算法的结果。

另一类是基于悬浮随机行走（FRW）[9–18] 的随机型算法。它通过蒙特卡洛（Monte Carlo, MC）随机采样 [19] 的方式进行电容提取。与传统确定型算法相比，FRW 算法避免使用线性方程组，从而具有内存使用少、更易并行、结果数值稳定性好、能更好的在精度和效率间进行取舍等优势。同时，它在局部计算方面有着其它算法不可比拟的优势。因此悬浮随机行走算法被业界认为是最有前途的算法之一 [18]。

目前，尽管 FRW 算法已经在很多商业软件（如 QuickCap，Rapid3D 等）中得到应用，但是它在面对目前的许多新挑战方面仍有很多的不足。例如在现在的实际应用中，需要计算电容的导线往往是很多相互连接的导体块，这些导体块甚至可能位于不同的层，由通孔连接（图1.1）。传统的 FRW 算法只能应对单个导体块的情形，无法进行全芯片电容提取。另外，尽管该算法被认为是具有进行大规模电路电容提取的潜力，但目前在这方面的研究和应用并不是很充分，其潜力还有待于进一步的挖掘。

目前的 FRW 算法研究的另一个缺陷在于，只考虑了曼哈顿形状（所有边均平

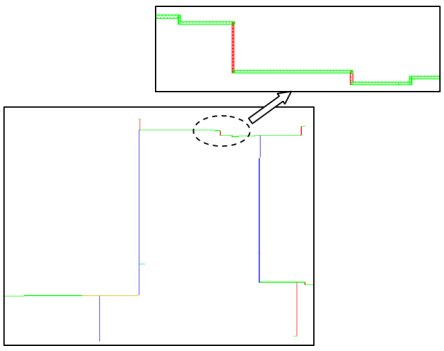


图 1.1　线网由相互连接的导体块组成

行于坐标轴）的导体 [20]。尽管在数字电路中，绝大多数导体都是曼哈顿形状，但是随着工艺和需求的发展，越来越多的非曼哈顿形状的导体被加入到电路中。

在电路密度增长的过程中，电路结构的复杂程度也在增加。目前，三维芯片被广泛的认为是一种可靠的可以超越摩尔定律的技术方向。通过将多层电路在垂直方向上叠加集成起来，三维芯片能够带来更小的面积，更快的速度，更低的功耗以及更好的时序性能。一种简单并且被广泛使用的三维芯片制造工艺是将多层由传统工艺制造的二维芯片通过硅通孔（TSV）[21–24] 堆叠在一起（图1.2）。

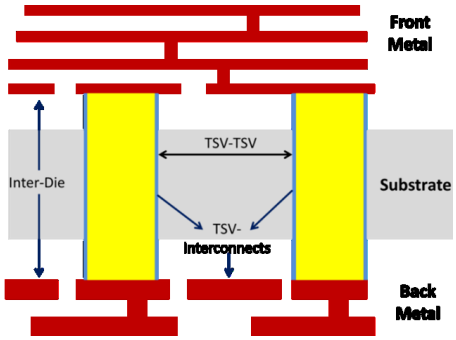


图 1.2　三维芯片中的 TSV 示意

TSV 在三维芯片中扮演着非常重要的角色，它负责在垂直方向上传输信号和电源。因此它的寄生参数需要被准确地模拟。TSV 的耦合电容由于对延迟和噪声的影响非常大，成为了一个非常重要的参数。过高的估计会造成在设计时对电路的过度保护从而不能充分的利用其性能，过低的估计则可能威胁电路的可靠性。已有的工作主要重点研究了 TSV 与 MOS 电容 [21–23,25]，而没有充分地分析 TSV 与一般的水平导线间的耦合电容。目前的电容提取中，都是用正方形截面的形体去近似地替代圆柱形的 TSV，而这会引起较大的误差。

除了圆柱形的 TSV 之外，在数字模拟混合电路中还会有大量的倾斜的导线 [26,27]，这些导线的横截面一般是不平行于坐标轴的长方形、平行四边形或者梯形（图1.3）。它们很难被曼哈顿形体近似。

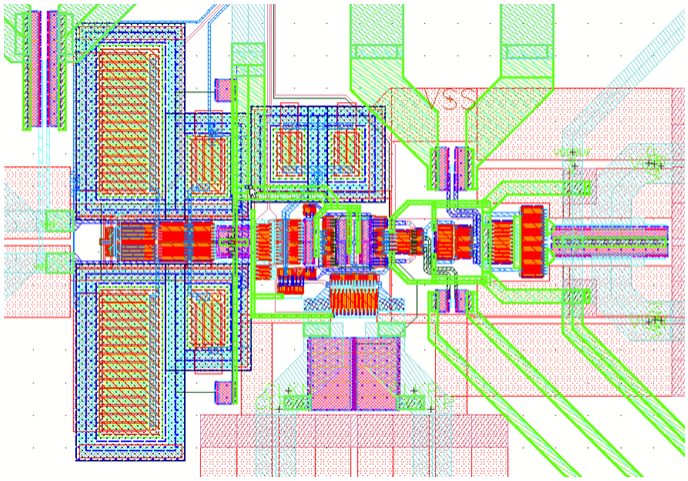


图 1.3　数字模拟混合电路中的倾斜导线示意

1.2　本文贡献

总结起来，目前针对大规模集成电路的 FRW 电容提取算法的主要局限在于：1) 没有充分地研究针对大规模互连结构中的大量导体块的空间管理技术；2) 不能对包含垂直通孔的由多个导体块连接形成的线网直接进行电容提取；3) 只局限于曼哈顿结构，不能准确地应对 TSV、倾斜导线等复杂的几何结构。本文的主要贡献在于针对这三项 FRW 电容提取算法的不足，提出有效的改进技术。

首先本文仔细研究了面向包含数百万导体块的大规模集成电路的空间管理算法，通过对比和改进已有的空间索引，如八叉树、网格等结构，将其优势综合在一起，设计出新的混合结构。它在面对大量导体块时具有较高的性能，并占用较少的存储资源。另外，对空间单元的候选导体列表的生成算法进行了重要的改善，大幅提升了整个空间管理结构的建立和查询效率。

针对包含垂直通孔的由多个导体块连接形成的完整线网，本文提出虚拟高斯面采样技术。该算法首先针对线网中的每个导体块独立地生成高斯面，再利用拒绝采样技术直接在每个导体块的高斯面上进行采样，避免的复杂的几何计算，大大简化了线网高斯面的生成。该技术一方面可以很好的扩展到更为复杂的几何结构，另一方面也能充分的利用重要性采样和分层采样等减小方差的技术。

为了适应非曼哈顿几何结构，本文首先拓展了原有的仅适用于曼哈顿结构的距离计算方法。在此基础上提出了旋转的转移区域行走技术，减少了每次采样所需的跳转次数。更进一步改进了针对曼哈顿结构的空间管理技术，使得它能很好地应用于包含非曼哈顿结构的复杂场景中。

上述改进提升了 FRW 电容提取算法在面对大规模电路时的性能，并且使得该算法能够更好更准确地适应更为复杂的电路结构。这大大地拓宽了 FRW 算法的应用场景，为其进一步地发展和应用做出了重要的贡献。

第 2 章　悬浮随机行走电容提取算法

2.1　利用蒙特卡洛随机采样计算积分

为了利用蒙特卡洛随机采样方法 [19] 计算积分

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

首先需要选择采样概率函数 $p(x)$，它应当满足 $p(x) > 0$ 并且积分 $\int_{\Omega} p(x)dx$ 是有限值（记为 A）[①]。积分 (2-1) 可以改写为：

$$I = A \int_{\Omega} \frac{p(x)}{A} \frac{f(x)}{p(x)} dx。 \tag{2-2}$$

此时由于 $\int_{\Omega} \frac{p(x)}{A} dx = 1$，根据蒙特卡洛方法，可以将 $\frac{p(x)}{A}$ 看做采样的概率密度函数（PDF），并利用求和来近似得到

$$I \approx \bar{I} = \frac{A}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}, \tag{2-3}$$

其中 x_i 是在被积区间 Ω 上按照 $\frac{p(x)}{A}$ 随机得到的第 i 个采样，N 是采样次数。

最简单的采样概率密度函数是均匀采样，即 $p(x) = 1$，此时 $A = |\Omega| = \int_{\Omega} dx$ 为被积区间的长度（一维情形）或面积（二维情形）。此时求和 (2-3) 可以被简化为

$$I \approx \bar{I} = |\Omega| \overline{f(x_i)}。 \tag{2-4}$$

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，而式 (2-3) 则是更一般的形式。

2.2　利用悬浮随机行走计算电势

对于空间中任意一点 r 处的电势 $\phi(r)$，有

$$\phi(r) = \oint_{S} P(r, r_{(1)}) \phi(r_{(1)}) dr_{(1)}, \tag{2-5}$$

其中 S 是一个的包围着点 r 的封闭区域的表面，$P(r, r_{(1)})$ 是与区域 S 对应的表面格林函数（surface Green's function）[18]，它只与区域 S 的形状和介质有关，与电

① 在本文中不要求采样概率密度函数的积分等于 1。

势无关。容易知道

$$\oint_S P(r, r_{(1)}) dr_{(1)} = 1, \tag{2-6}$$

这意味着，对于一个固定的点 r，$P(r, r_{(1)})$ 可以看做是在包围点 r 的区域 S 上随机选择一点 $r_{(1)}$ 的概率密度函数。因此，可以在 S 上随机选取大量点并用其电势的平均值来估计 $\phi(r)$。

如果区域 S 是一个立方体，并且点 r 是其中心点，那么表面格林函数 $P(r, r_{(1)})$ 只与点 $r^{(1)}$ 的相对位置和区域 S 内的电介质分布有关，而与 S 的大小无关，即等比例的放大或缩小 S 并不会改变 $P(r, r_{(1)})$ 的值。当区域 S 是单介质区域时，$P(r, r_{(1)})$ 可以被解析的得到。对于分层的多介质情形，也可以使用其它数值方法得到 [18]。因此可以预先计算好 $P(r, r_{(1)})$，并利用它在区域表面 S 上随机采样。

当 $r_{(1)}$ 点的电势 $\phi(r_{(1)})$ 未知的时候，可以递归地使用公式 (2-5) 得到如下的复合积分公式：

$$\begin{aligned} \phi(r) = &\oint_{S_{(1)}} P_{(1)}(r, r_{(1)}) \oint_{S_{(2)}} P_{(2)}(r_{(1)}, r_{(2)}) \cdots \\ &\oint_{S_{(k)}} P_{(k)}(r_{(k-1)}, r_{(k)}) \phi(r_{(k)}) dr_{(k)} \cdots dr_{(2)} dr_{(1)}, \end{aligned} \tag{2-7}$$

其中 $S_{(i)}, (i = 1, \ldots, k)$ 分别是以点 $r_{(i-1)}$ 为中心的立方体，$P_{(i)}(r_{(i-1)}, r_{(i)})$ 是 $S_{(i)}$ 的表面格林函数。在单介质问题中，有 $P_{(i)}(r_{(i-1)}, r_{(i)}) = P_{(i+1)}(r_{(i)}, r_{(i+1)})$，即只需要一个表面格林函数即可。

公式 (2-7) 可以被理解为一个悬浮随机行走过程：对于第 i 次跳转，先以点 $r_{(i-1)}$ 为中心构造一个最大的不包含任何导体的立方体 $S_{(i)}$，然后在上面按照概率密度函数 $P_{(i)}(r_{(i-1)}, r_{(i)})$ 随机地选择一个点 $r_{(i)}$。当 k 次跳转之后到达的点 $r_{(k)}$ 处的电容已知时，这次采样就可以结束了。在电容提取问题中，导体上的电势均为已知的，因此只要点 $r_{(k)}$ 落在导体表面就可以结束这次采样。在很多次采样之后，就能用大量的采样值来很好的近似 $\phi(r)$。

2.3　利用悬浮随机行走计算电容

上述利用悬浮随机行走计算电势的方法，很容易被应用于电容的计算。假设对于导体 i，其电势和电荷量分别为 V_i, Q_i，则根据电容的定义，有

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

考虑其第 i 行：

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

如果令公式 (2-9) 中 $V_i = 1V, V_{j(j\neq i)} = 0V$，便可得到 $C_i = C_{ii} = Q_i$，即导体 i 的总电容在数值上等于其电荷量。如此一来，只要能求出电荷量 Q_i 就能得到电容 C_i。

根据高斯定理，一个闭合区域内的总电荷量等于其表面的法向电位移的积分。如果用一个闭合区域包围导体 i，则其内部的总电荷量就是 i 上的电荷 Q_i，因此有：

$$C_i = Q_i = \oint_G D(r)\cdot\hat{n}(r)dr = -\oint_G \varepsilon(r)\bigtriangledown\phi(r)\cdot\hat{n}(r)dr, \tag{2-10}$$

其中 G 是一个包围导体 i 的封闭区域的表面，常被称为高斯面，$\varepsilon(r)$ 是点 r 处的电介质常数，$\hat{n}(r)$ 是点 r 处 G 的法向量。将公式 (2-5) 代入公式 (2-10) 可以得到：

$$C_i = -\oint_G \varepsilon(r)\oint_{S_r}\bigtriangledown P(r,r_{(1)})\cdot\hat{n}(r)\phi(r_{(1)})dr_{(1)}dr, \tag{2-11}$$

其中 S_r 是包围 r 点的封闭区域的表面。

为了利用蒙特卡洛随机采样方法来计算公式 (2-11) 中的两个积分，首先需要设定采样概率，分别记它们为 $f(r)$ 和 $h(r,r_{(1)})$，下一节将详细的讨论它们的具体选择。令 $F = \oint_G f(r)dr$，$H(r) = \oint_{S_r} h(r,r_{(1)})dr_{(1)}$，公式 (2-11) 变为

$$C_i = \oint_G \frac{f(r)}{F}\oint_{S_r}\frac{h(r,r_{(1)})}{H(r)}\omega(r,r_{(1)})\phi(r_{(1)})dr_{(1)}dr, \tag{2-12}$$

其中

$$\omega(r,r_{(1)}) = -F\frac{\varepsilon(r)}{f(r)}\frac{H(r)\bigtriangledown P(r,r_{(1)})\cdot\hat{n}(r)}{h(r,r_{(1)})} \tag{2-13}$$

称为权值函数 [18]。

公式 (2-12) 的第一个积分可以看做是在高斯面 G 上以 $\frac{f(r)}{F}$ 为 PDF 的随机采样，第二个积分则是在 S_r 上以 $\frac{h(r,r_{(1)})}{H(r)}$ 为 PDF 的随机采样，其中唯一的未知量 $\phi(r_{(1)})$ 可以根据公式 (2-7) 利用悬浮随机行走得出。图2.1和算法1 描述了完整的悬浮随机行走电容提取算法流程，其中被提取电容的导体称为主导体。

2.4　重要性采样与分层采样

从算法1可以看出，悬浮随机行走算法的主要部分在于两重循环，因此其时间消耗可以由下式表示：

$$T_{tital} = N_{walk}\cdot N_{hop}\cdot T_{hop}, \tag{2-14}$$

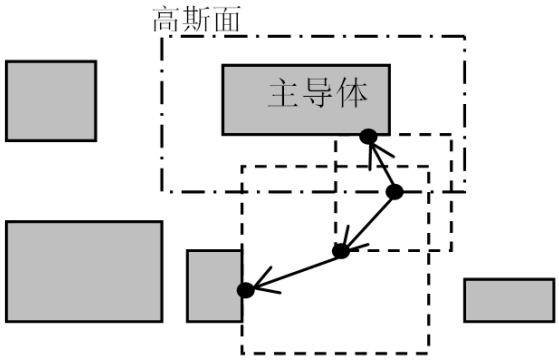


图 2.1　悬浮随机行走电容提取算法示例

算法 1 悬浮随机行走电容提取算法

输入： 互连线导体的三维版图, 主导体 i

输出： 电容 C_{ij}

为主导体 i 生成高斯面 G
$n \leftarrow 0$
$C_{ij} \leftarrow 0$
repeat
    $n \leftarrow n+1$
    在 G 上随机选择一点 r，并以 r 为中心生成立方体转移区域 S
    在 S 上随机选择一点 t，并计算权值 $w \leftarrow \omega(r,t)$
    $r \leftarrow t$
    while r 不在导体 j 表面 **do**
        以 r 为中心生成立方体转移区域 S
        在 S 上随机选择一点 t
        $r \leftarrow t$
    end while
    $C_{ij} \leftarrow C_{ij} + w$
until C_{ij} 的精度满足终止条件
return $\frac{C_{ij}}{n}$

其中 N_{walk} 是达到指定的精度所需的采样数，N_{hop} 是每个采样点所需的平均随机行走跳转次数，T_{hop} 是每次跳转所需的平均时间。减小其中的每一个量都可以提高整个算法的效率。这一节主要考虑减小 N_{walk}。

回顾一下利用蒙特卡洛随机采样计算积分的公式 (2-3)。根据大数定律 [28]，$\bar{I}$

是一个满足正态分布的随即变量，其均值为 I，方差为

$$var(\bar{I}) = \frac{var(\frac{f(x)}{p(x)})}{N}。 \tag{2-15}$$

因此，为了让随机得到的计算结果 $\bar{I}$ 的标准差不大于指定的阈值 σ，需要的最少的采样次数应当满足

$$N \geq \frac{var(\frac{f(x)}{p(x)})}{\sigma^2}。 \tag{2-16}$$

这意味着如果想减少达到指定精度所需的采样次数，就需要减少被积函数 $\frac{f(x)}{p(x)}$ 的方差。在 $\frac{f(x)}{p(x)}$ 中，需要被计算的函数 $f(x)$ 是不可变的，因此需要选择合适的 $p(x)$。选择的 $p(x)$ 与 $f(x)$ 越接近，则被积函数的方差越小，计算的效率就越高。这种根据待计算函数来选择采样概率从而减小方差的方法称为重要性采样（IS）[13]。

回到利用随机采样计算电容的公式 (2-12)，其被积函数为 $\omega(r, r_{(1)})\phi(r_{(1)})$，其中 $\phi(r_{(1)})$ 由随机行走最终击中的导体决定，其值只能为 0 或者 1，并且不能提前预知。可以改变的只有 $\omega(r, r_{(1)})$(公式 (2-13)) 中的 $f(r)$ 和 $h(r, r_{(1)})$。

注意到公式 (2-13) 可以被分为两部分：

$$\begin{aligned} \omega(r, r_{(1)}) &= -F\omega_G(r)\omega_S(r, r_{(1)}) \\ \omega_G(r) &= \frac{\varepsilon(r)H(r)}{f(r)} \qquad (2\text{-}17) \\ \omega_S(r, r_{(1)}) &= \frac{\nabla P(r, r_{(1)}) \cdot \hat{n}(r)}{h(r, r_{(1)})}。 \qquad (2\text{-}18) \end{aligned}$$

首先考虑式 (2-18)，注意到其分子 $\nabla P(r, r_{(1)}) \cdot \hat{n}(r)$ 是一个可以很容易预先计算的函数，因此可以考虑选择它本身作为采样概率。但是注意采样概率需要是非负的，因此选择

$$h(r, r_{(1)}) = |\nabla P(r, r_{(1)}) \cdot \hat{n}(r)|。 \tag{2-19}$$

此时有

$$\omega_S(r, r_{(1)}) = sign(\nabla P(r, r_{(1)}) \cdot \hat{n}(r)) = \pm 1, 0。 \tag{2-20}$$

其中

$$sign(x) = \begin{cases} +1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \end{cases}。 \tag{2-21}$$

再来考虑式 (2-17)，其中

$$H(r) = \oint_{S_r} | \nabla P(r, r_{(1)}) \cdot \hat{n}(r)| dr_{(1)}, \tag{2-22}$$

虽然与 S_r 的选取有关，但是，可以预先为单位立方体 U 求出 $H_u(r) = \oint_U h(r, r_{(1)}) dr_{(1)}$。对于实际的 S_r，有

$$H(r) = \frac{H_u(r)}{L(r)}, \tag{2-23}$$

其中 $L(r)$ 是实际转移立方体 S_r 的边长。此时公式 (2-17) 变为

$$\omega_G(r) = \frac{\varepsilon(r) H_u(r)}{L(r) f(r)}, \tag{2-24}$$

进一步的，公式 (2-13) 变为

$$\omega(r, r_{(1)}) = \mp F \frac{\varepsilon(r) H_u(r)}{L(r) f(r)}。 \tag{2-25}$$

第4章将继续讨论如何选取 $f(r)$ 以进一步减小 ω 的方差。

除了重要性采样之外，还有一种可以有效减小方差的方法，称为分层采样（SS）[13]。再来回顾下积分 (2-1)，如果将被积区间 Ω 分为几个部分 Ω_i 满足 $\bigcup_i \Omega_i = \Omega$ 并且 $\Omega_i \cap \Omega_j = \emptyset$，则积分 (2-1) 可以被改为

$$\begin{aligned} I &= \sum_i I_i = \sum_i \int_{\Omega_1} f(x) dx \\ &= \sum_i A_i \int_{\Omega_i} \frac{p_i(x)}{A_i} \frac{f(x)}{p_i(x)} dx, \end{aligned} \tag{2-26}$$

其中 $p_i(x)$ 是在 Ω_i 上选择的采样概率，$A_i = \int_{\Omega_i} p_i(x) dx$。此时根据蒙特卡洛方法，有

$$I \approx \bar{I}' = \sum_i \bar{I}_i = \sum_i \frac{A_i}{N_i} \sum_j \frac{f(x_{ij})}{p_i(x_{ij})}, \tag{2-27}$$

其中 N_i 是在 Ω_i 上的采样次数。显然的 $\bar{I}'$ 也是一个以 I 为均值的正态分布随机变量，可以证明 [28] 其方差满足

$$var(\bar{I}) \geq var(\bar{I}')。^{①} \tag{2-28}$$

分层采样技术通过将被积函数按照取值的不同分为不同区间进行计算，在每个区间上的方差可以较小，并且式 (2-28) 说明，这样做一般会使得总体的方差减小。

① 一般情况下不等号都是严格的。

再来看计算电容时的权值公式 (2-25)，其中 $H_u(r)$ 只与 $\hat{n}(r)$ 有关，而 $L(r)$ 与高斯面到主导体的距离有关，并且一般的 $\varepsilon(r)$ 与高斯面的位置有关。由此考虑，对于公式 (2-12) 的第一个积分，将长方体高斯面 G 分为 6 个朝向不同方向的面进行分层采样可以有效的减小方差。

第 3 章　面向大规模结构的快速导体索引空间管理技术

本章将主要介绍在悬浮随机行走算法中用于加速构造转移区域的空间管理结构。3.1节主要介绍为什么需要空间管理结构以及空间管理结构的基本设计思想。3.2节主要介绍每个空间单元如何管理附近的候选导体，以及如何优化候选导体列表的建立和查询过程。3.3节主要介绍空间单元的索引方式，将通过对比分析找出最优的索引结构。最后在3.4节将通过多个实际的或人造的例子来验证本章算法的正确性与效率。

3.1　空间管理的基本概念和方法

根据算法1的描述，在每次进行跳转之前都需要先建立包围当前点的转移区域。为了便于计算并且能更好的适应数字电路中导线的结构，转移区域是一个以当前点为中心的立方体。并且公式 (2-5) 要求转移区域内不能包含任何导体。

回顾公式 (2-14)，为了减少 N_{hop}（每次采样所需的跳转次数），每次使用的转移区域尽量与至少一个导体相贴，这样可以增加这一次跳转后采样点落在某个导体上从而结束这次行走的概率。

为了得到这个最大的转移区域，需要计算当前点到所有导体的最短距离。在本章中，只考虑曼哈顿形状的导体 [20]，并且转移区域也是曼哈顿形状的，因而最常见的欧拉距离在这里并不适用，取而代之的是一种基于无穷范数的距离，具体的：

定义 3.1：　三维点 $p = (x_p, y_p, z_p)$ 与 $q = (x_q, y_q, z_q)$ 间的距离：$dist(p, q) \leftarrow \max(|x_p - x_q|, |y_p - y_q|, |z_p - z_q|)$；

定义 3.2：　点 p 与长方体 $C = (l, r)$（l 和 r 分别是 C 的坐标值最小和最大的两个点）之间的距离：$dist(p, C) \leftarrow \max(x_l - x_p, y_l - y_p, z_l - z_p, x_p - x_r, y_p - y_r, z_p - z_r)$；

定义 3.3：　两个长方体 $A = (l_A, r_A)$ 与 $B = (l_B, r_B)$ 间的距离：$dist(A, B) \leftarrow \max(x_{l_A} - x_{r_B}, y_{l_A} - y_{r_B}, z_{l_A} - z_{r_B}, x_{l_B} - x_{r_A}, y_{l_B} - y_{r_A}, z_{l_B} - z_{r_A})$。

可以认为定义3.1和3.2是定义3.3在其中一个或两个长方体缩小到一个点时的特例。从定义3.3和图3.1可以看出，对于相交的两个长方体，它们的距离可以为负值，这点与一般意义下的距离定义不同。另外指出一点，在这样的定义下，到定点的距

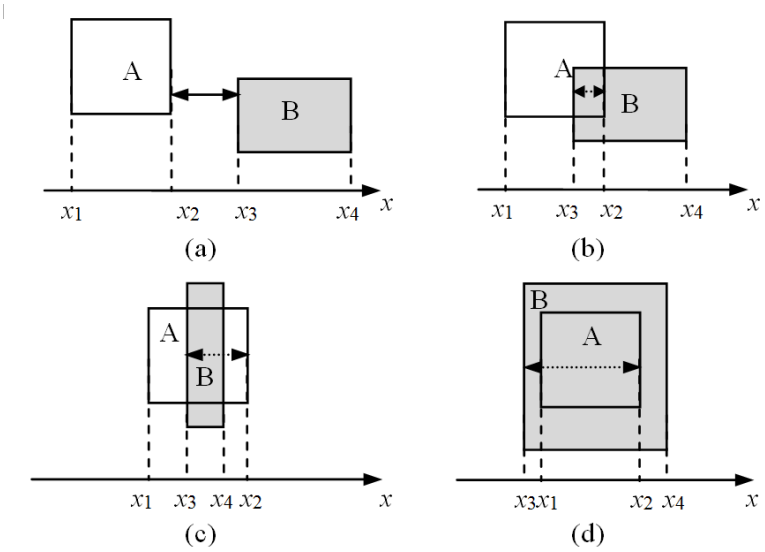


图 3.1　两个长方体（二维情形）间的四种位置关系，四个例子都有 $dist(A,B) = x_3 - x_2$

离等于定长的形体不再是球体而是立方体，这个距离就是立方体的半边长，也可以称为立方体的半径。

有了上述距离的定义，对当前点计算最大转移区域就很直观了，只需计算当前点到所有导体的距离的最小值，以这个最小值作为半边长就可以得到一个立方体形状的转移区域，并且这个转移区域恰好与最近的一个导体相贴。

对于空间中导体很多的情形，如果依次遍历每个导体并计算距离将会是很慢的。因为这个过程在每次跳转时都需要进行一遍，所以这部分时间会计入 T_{hop}[①]，从而会明显地影响整个电容提取算法的效率的。因此需要一种快速的计算转移区域大小的方法。

为了减少每次需要尝试的导体数量，需要将所有的导体建立一个索引，这样每次只需要尝试附近的一些导体即可，从而加速转移区域的构造。这样的导体索引就是空间管理。它需要支持两个操作：1) 初始化，将所有导体块添加入索引；2) 对于任给的查询点[②]，返回合适的转移区域的大小（半边长）。注意，这里没有要求空间管理结构一定给出最大的转移区域半边长，这是因为，最大是出于减少 N_{hop} 提高性能的考虑，但并不影响正确性。因此只是希望得到的转移区域尽量大就好。

实现空间管理结构的基本思路是首先将整个空间划分成许多小的单元，每个空间单元中包含附近的导体。所有的空间单元被索引在一起。查询时首先找到包含查询点的单元，然后在这个单元及其附近找到最近的导体。按照这样的思路，

① 事实上也是 T_{hop} 中比例最大的部分。
② 查询点不会出现在任何导体内部。

将会面对两个问题：1) 每个空间单元应该包含哪些导体，如何找到距离查询点最近的；2) 所有的空间单元应该组织成什么结构，如何查找查询点落在哪个单元内。本章接下来的两节将分别解决它们。

3.2　空间单元和候选导体列表

3.2.1　候选导体列表

当找到包含查询点的空间单元之后，就可以利用这个空间单元所包含的信息找到距离查询点最近的导体。如果这个空间单元只包含与其相交的导体，那么当查询点靠近空间单元的边缘时，还需要访问相邻的空间单元。如果这个空间单元能够包含附近所有导体，那么便可以仅利用这个空间单元中包含的信息找到最近的导体。然而需要注意的是，如果一个单元包含的导体太多，又会影响到查询的效率，因此本文使用了候选导体（candidate）的概念，一个空间单元只需要包含较少的导体，又能保证对其内部的任何一个查询点都能正确找到最近的导体。

在定义候选导体之前，先定义遮挡关系 [17,18]，它是两个导体相对于一个空间单元的位置关系，

定义 3.4：　导体 A 在空间单元 T 上遮挡 B: $A \leq_T B \Leftrightarrow \forall p \in T, dist(p,A) \leq dist(p,B)$;

定义 3.5：　导体 A 是空间单元 T 的候选导体：$\forall$导体$B, \neg(B \leq_T A)$。

从定义3.4可以看出，遮挡关系满足传递性，即 $A \leq_T B \wedge B \leq_T C \Rightarrow A \leq_T C$。因此，判断一个新的导体是否是这个空间单元的候选导体时只需要比较新的导体与这个空间单元中已有的候选导体间的遮挡关系，不必每次都查看所有导体，算法2描述了这一过程。

定义3.4不能直接用来判断两个导体间是否有遮挡关系，因为无法遍历空间单元中的所有点。为此本文提出算法3来判断两个导体相对于一个空间单元的遮挡关系。

可以看出算法3涉及大量的几何计算，因而计算效率必然不高。再结合算法2可以看出，将 n 个导体加入一个空间单元需要 $O(n^2)$ 次调用算法3。考虑到整个问题空间可能会划分成众多空间单元，在初始化时尝试将所有导体插入所有空间单元的候选导体列表将会是一个非常耗时的过程。为此提出两种技术来避免如此庞大的计算量。

算法 2 尝试将一个新导体加入空间单元的候选导体列表

输入： 导体 A，空间单元 T，T 的候选导体列表 $list_T$

输出： 更新 $list_T$

```
for B ∈ list_T do
    if A ≤_T B then
        从 list_T 中删除 B
    end if
    if B ≤_T A then
        return
    end if
end for
将 A 插入 list_T
```

3.2.2　遮挡关系的快速计算

对于一个比较小的空间单元，大部分导体都不是它的候选导体，但是根据算法2，这些导体依然需要与空间单元的候选导体比较遮挡关系。针对这一点，如果能快速地过滤掉那些肯定不是候选导体的导体，算法2的效率将能提高很多。

对于一个空间单元 T 和它的候选导体列表 $list_T$，可以如下定义门限距离：

定义 3.6：　空间单元 $T=(l,r)$ 的大小：$size_T \leftarrow \min(x_r-x_l, y_r-y_l, z_r-z_l)$；

定义 3.7：　空间单元 T 的门限距离：$l_T \leftarrow size_T + \min_{C\in list_T}(dist(T,C))$；
如果 $list_T$ 为空：$l_T \leftarrow +\infty$。

根据上述定义，容易得出如下定理：

定理 3.1：　除非 T 完全被导体覆盖[①]，否则 $l_T \geq 0$；

证明　显然的 $size_T \geq 0$。
对任意导体 A，如图3.1(a) 所示，当 A 与 T 不相交时，有 $dist(A,T) \geq 0 \geq -size_T$；
如果3.1(b-d) 所示，当 A 与 T 相交时，虽然有 $dist(A,T)<0$，但仍保证 $-dist(A,T) \leq size_T$。
综上有 $dist(A,T)+size_T \geq 0$，故 $l_T \geq 0$。　□

定理 3.2：　对于空间单元 T 和导体 A，$l_T \leq dist(A,T) \Rightarrow A$ 不是 T 的候选导体。

① 当一个空间单元完全被导体覆盖时，查询点不会落在这个空间单元中，因此在本文中不考虑这样的空间单元。

算法 3 判断两个导体间的遮挡关系

输入： 导体 $A=(l_A,r_A)$，导体 $B=(l_B,r_B)$，空间单元 $T=(l_T,r_T)$

输出： 是否有 $A \leq_T B$

```
1: d_A ← dist(A,T)
2: d_B ← dist(B,T)
3: if d_A > d_B then
4:     return false
5: end if
6: for w ∈ {x,y,z} do
7:     if w_{l_A} > max(w_{l_B}, w_{l_T} + d_B) then
8:         return false
9:     end if
10:    if w_{r_A} < min(w_{r_B}, w_{r_T} − d_B) then
11:        return false
12:    end if
13: end for
14: return true
```

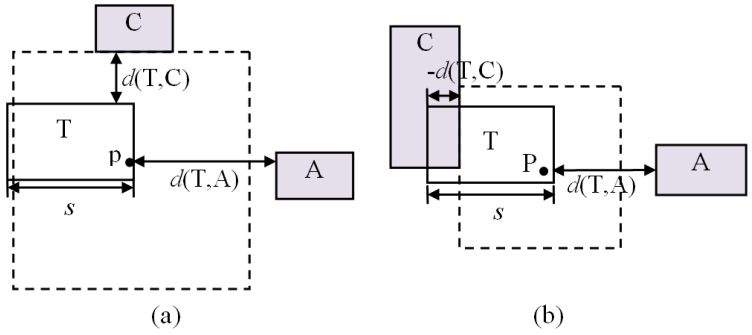


图 3.2　空间单元 T，候选导体 C 和新导体 A 的位置关系

证明 首先，根据定理3.1可以得到，若 $dist(A,T) \geq l_T$ 则 $dist(A,T) \geq 0$，即 A 与 T 不相交。

如图3.2所示，此时有 $\forall p \in T$[①]$, dist(A,T) \leq dist(p,A)$。

同时，对于任意导体 C 还有 $\forall p \in T, dist(p,C) \leq dist(C,T) + size_T$。

不妨取 C 为满足 $l_T = size_T + dist(C,T)$ 的导体，此时联立上述两个不等式可以得到：

$$\forall p \in T, dist(p,C) \leq dist(C,T) + size_T = l_T \leq dist(A,T) \leq dist(p,A)$$

① 只需要考虑不被包含在任何导体内的点。

即 $C \leq_T A$。故 A 不是 T 的候选导体。 □

定理3.2可以将算法2修改为算法4。修改后的算法将在大多数情况下只需进行一次距离计算即可结束。

算法 4 利用门限距离优化的空间单元候选列表插入算法

输入： 导体 A，空间单元 T，T 的候选导体列表 $list_T$，T 的门限距离 l_T

输出： 更新 $list_T, l_T$

```
d ← dist(A, T)
if d ≥ l_T then
    return
end if
for B ∈ list_T do
    if A ≤_T B then
        从 list_T 中删除 B
    end if
    if B ≤_T A then
        return
    end if
end for
size_T = min(x_{r_T} − x_{l_T}, y_{r_T} − y_{l_T}, z_{r_T} − z_{l_T})
l_T ← min(l_T, d + size_T)
将 A 插入 list_T
```

3.2.3　不完整的候选导体列表

注意到在定义3.7中，当 T 的候选导体列表 $list_T$ 为空时，T 的门限距离 l_T 被定义为 $+\infty$。这使得任何一个候选导体，不论它距离这个空间单元有多远，都一定会被加入到这个空间单元的候选导体列表中。这样做的好处是，它能保证在这个空间管理单元内一定能找到最近的导体，从而保证每次得到的转移区域都是最大的。

但是这样做也是有代价的。考虑到，对于大多数空间单元，最终的候选导体都会是它内部或者附近的导体，其门限距离 l_T 不会是一个很大的数。而将其初始化为 $+\infty$ 意味着，需要在动态插入的过程中逐渐将 l_T 缩小的最终值，而同时，在初期插入的很多候选导体可能又在后期被删除了。如果考虑最坏的情况，所有导体按照距离空间单元 T 由远及近的顺序插入，那么门限距离将几乎不能起到过滤

导体的作用。为了让门限距离优化发挥最大的作用，l_T 的的初始值应该尽可能的接近最终值。如果 l_T 的初始值过小，则不能保证每个候选导体都被找到，就不能保证得到最大的转移区域。

然而，正如3.1节所述，每次都找到最大的转移区域并不是算法所必须的。此时，如果愿意放弃这一点，将可以进一步大幅度的加快候选导体列表的建立过程。为此，本文提出临近区域和临近距离的概念。

定义 3.8：　对于空间单元 T 和任意非负长度 d_{nb}，T 的临近区域：$neighbor_T \leftarrow \{p|dist(p,T) \le d_{nb}\}$，其中 d_{nb} 称为临近距离。
A 在 B 的临近区域内，即 B 在 A 的临近区域内：$dist(A,B) < d_{nb}$。

如果只考虑临近区域内的导体，即将 $list_T$ 初始化为 d_{nb}，那么对于大部分与空间单元 T 距离较远的导体就可以直接被排除，而不需要进行复杂的遮挡关系判断（算法3）。

然而如果忽略了远处的导体，得到的候选导体列表可能会是不完整的，空间单元将无法保证给出最大的转移区域。以 p 为中心的转移区域的半边长将不能大于点 p 到临近区域 $neighbor_T$ 的边界的最小距离 d_p（$= d_{nb} - dist(p,T)$）。因为这个空间单元并不知道 $neighbor_T$ 以外的导体分布情况，所以其产生的转移区域也只能在 $neighbor_T$ 以内。

如图3.3所示，对于空间单元 T 和它的临近区域 $neighbor_T$，导体 B_1，B_2 和 B_3 与 $neighbor_T$ 相交，并且在候选导体列表 $list_T$ 中；导体 B_4 在 $neighbor_T$ 之外，因此它不在 $list_T$ 中。对于查询点 p，距离它最近的候选导体是 B_3 且距离为 d_3，然而，因为 $d_3 > d_p$，只能以两者中较小的 d_p 为半边长生成转移区域。如果以 d_3 为转移区域的半边长，那么转移区域将会与 B_4 相交。

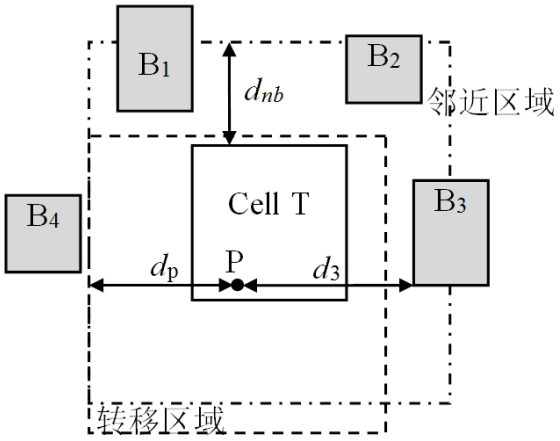


图 3.3　空间单元 T 的临近区域和候选导体

d_{nb} 的值决定了 $neighbor_T$ 的大小。其越小能直接忽略的导体越多，同时候选导体列表不完整的可能性也越大。下面讨论 d_{nb} 该如何取值，以及么什么情况下能保证候选导体列表是完整的。

定理 3.3：　对于一个空间单元 T，$l_T < d_{nb} \Rightarrow list_T$ 是完整的。

证明　对于任意一个不在 $neighbor_T$ 中导体 A，有 $dist(A,T) \geq d_{nb}$。若 $l_T < d_{nb}$，那么就有 $dist(A,T) > l_T$，由定理3.2可知 A 不是候选导体。因此 $list_T$ 是完整的。　□

定理 3.4：　对于一个空间单元 T，若有导体与 T 相交，则 $l_T \leq size_T$。

证明　若有导体与 T 相交，则一定有候选导体 C 与导体相交，即 $dist(C,T) < 0$。根据定义3.7即可得到 $l_T \leq size_T + dist(C,T) < size_T$。　□

综合定理3.3和3.5，可以得到

定理 3.5：　对于一个与某个导体相交的空间单元 T，若 $d_{nb} \geq size_T$，则 $list_T$ 是完整的。

根据经验，大部分的空间单元尤其是处于导体密集区域的空间单元[①]，基本都与导体相交。根据定理3.5，如果取 $d_{nb} = size_T$ 则能保证大部分空间单元的候选导体列表都是完整的。

3.2.4　对候选导体列表排序

为了从包含查询点的空间单元的候选导体列表中找到距离查询点最近的导体，需要遍历整个候选导体列表。在一般情况下，列表中导体排列的顺序并不影响遍历过程的效率，因为必须看到每个导体才能确定找到了最近的那个。但是若对候选导体列表做适当的排序，则有机会能够提前结束遍历，即不需要尝试所有的导体也能找到最近的那个。减少尝试候选导体的次数能显著的加快找到最近导体的速度。从而减少整个算法的时间消耗。

首先，在遍历候选导体列表的过程中，如果当前的最小距离 d_{min} 变为了 0，那么遍历就可以提前结束了，因为最小距离 d_{min} 已经不能更小了。另一方面可以得到，

定理 3.6：　对于空间单元 T，导体 A 和当前已经得到的最小距离 d_{min}，若有 $dist(A,T) \geq d_{min}$，那么 A 不可能是限制转移区域大小的导体。

① 这样的空间单元也是查询频率最高的。

证明 因为 $d_{min} \geq 0$，所以 $dist(A,T) \geq d_{min} \geq 0$ 意味着 A 与 T 不相交。
此时，对于 T 内的查询点 p，容易得到 $dist(p,T) \geq dist(A,T) \geq d_{min}$。
因此导体 A 不能更新 d_{min}，即不会限制转移区域的大小。 □

定理 3.7： 如果 $list_T$ 中的所有导体 A_i 满足 $dist(A_i,T) \leq dist(A_{i+1},T)$，当对某个导体 A_k 有 $dist(A_k,T) \geq d_{min}$ 时，那么对于任何 $i \geq k$，导体 A_i 都不会限制转移区域大小。

证明 $\forall i \geq k$ 有 $dist(A_i,T) \geq dist(A_k,T) \geq d_{min}$，根据定理3.6，$A_i$ 不会限制转移区域大小。 □

定理3.7意味着可以在候选导体列表建立完成之后，对其中的导体按照到空间单元的距离排序。这样当遍历到某个导体 A_k 时，若满足 $dist(A_k,T) \geq d_{min}$，就可以直接断言 d_{min} 就是安全的转移区域的半边长，而不必再尝试剩下的导体。算法5描述了对于空间单元 T 和其内一点 p 计算转移区域半边长的过程。

算法 5 计算转移区域的半边长

输入： 空间单元 T，T 内的点 p
输出： 转移区域半边长 d_{min}

```
1:  d_min ← d_nb − dist(p,T)
2:  for C ∈ list_T do
3:      if dist(C,T) ≥ d_min then
4:          break
5:      end if
6:      d ← dist(p,C)
7:      if d = 0 then
8:          break
9:      else if d < d_min then
10:         d_min ← d
11:     end if
12: end for
13: return d_min
```

3.3　空间单元的组织和索引结构

空间索引结构有很多种，比如 K-D 树 [29]，均匀网格 [30]，八叉树 [17,31] 等。其中 K-D 树是最常用的高维索引结构。与二叉树相似，每个 K-D 树的节点都有两

个子节点，分别代表着当前节点所描述的区域的两个子区域。分割当前区域的平面的方向在树的不同深度上各不相同，这样它便能适应高维而不仅仅是一维的情形。K-D 树的一个特点是，对每一个节点，其两个子节点被查询的频率应当是相同的。然而，在电容提取问题中，无法在实际进行随机行走之前估计查询点的分布情况。因此 K-D 树在这里并没有什么特别的优势。相反的，由于它每层只有两个子节点，其深度会是一般的八叉树的 3 倍，这反而是不利的。

3.3.1　八叉树索引结构

首先考虑八叉树结构的建立过程。基于树结构的特点，为了将一个导体插入合适的叶子节点，必须从八叉树的根节点遍历到它的每个叶子节点，然后利用算法4将导体插入候选导体列表。在这个过程中，虽然非叶子节点不包含候选导体，但它们的门限距离对于避免不必要的计算依然是很有帮助的。因为如果一个导体不是某个空间单元的候选导体，那么它也不会是这个空间单元的任何子空间的候选导体，所以当一个导体与八叉树中某个节点的距离大于这个节点的门限距离时，就不必再尝试这个节点的子节点。当导体被加入到一个叶子节点的候选导体列表时，这个节点的门限距离可能会相应的更新。同时也可以更新这个节点的所有父节点的门限距离。然而这样做的消耗比它带来的好处大，因此在本文的实现中，只是更新了叶子节点的门限距离。算法6描述了将导体 A 插入节点 T 的过程。整个八叉树的建立就是将所有导体依次插入到根节点中。

在算法6中，当一个叶子节点的候选导体数量大于 $threshold_l$ 时，将其划分为 8 个子节点。同时还限制了每个叶子节点的大小不能小于 $threshold_s$，这是因为有时将一个节点分为 8 个子节点并不能减少候选导体的数量，因此使用这个阈值来避免无限的节点划分。

八叉树结构的建立过程也可以利用并行计算进行加速。其基本思路是利用导体被加入八叉树的顺序对结果没有影响的特点，让每个计算单元独立地将一个导体加入到八叉树中。在这个过程中如果多个计算单元需要同时访问一个节点则需要利用锁来避免冲突。可以利用随机访问顺序等技术来减少冲突，也可以利用预分裂技术来彻底避免冲突。详细地讨论请参考 [32]。

3.3.2　多层网格索引结构

在空间管理结构中查询最近的导体[①] 时，首先需要找到包含查询点的空间单元。使用八叉树结构时，这个过程需要从树的根节点开始，直到到达某个叶子节

① 事实上，需要的是转移区域的半边长，并不是最近的导体。

算法 6 将导体插入八叉树的节点

输入： 导体 A，八叉树节点 T

输出： 更新八叉树

```
1: if dist(A, T) ≥ l_T then
2:     return
3: end if
4: if T 是叶子节点 then
5:     调用算法4将 A 插入 list_T 中
6:     if length(list_T) > threshold_l ∧ size_T < threshold_s then
7:         将 T 分为 8 个子节点：T_1, T_2, ..., T_8
8:         for C ∈ list_T do
9:             for i ← 1...8 do
10:                将 C 插入 list_{T_i}
11:            end for
12:        end for
13:    end if
14: else
15:    for i ← 1...8 do
16:        将 A 插入 T_i
17:    end for
18: end if
```

点为止。为了减小这个开销，可以尝试使用三维的均匀网格结构来索引空间单元。由于每个单元的大小都是相同的，因此可以很容易地将查询点的坐标与单元格的大小相除来得到包含查询点的单元的坐标。此时，虽然每个单元的大小是相同的，但是由于导体的分布并不是完全均匀的，每个单元包含的导体以及候选导体数并不相同。使用较小的单元格大小可以减少最大的候选导体数量，但是会产生非常大量的网格节点。比如一个 916μm×923μm×210μm 的问题空间将包含 6.6×10^6 个边长为 3μm 的立方体单元格。

如果每个导体都可能被插入到每个单元格，那么建立整个网格结构的过程将非常缓慢。好在不完整的候选导体列表技术可以改善这一点[①]。这里设置临近距离 d_{nb} 为单元格边长 $size_T$，根据定理3.5，对于导体分布较密集的区域的单元格，它们的候选导体列表将很有可能是完整的。

① 对于八叉树结构而言，不完整的候选导体列表是一个可有可无的选择，但对于网格结构是必须的。

与具有相同数量的叶子节点的八叉树结构相比，网格索引中单元格的最大候选导体列表数会更大，这是由于导体分布不均匀造成的，并且会严重影响到查询操作的效率。解决这一问题的一种有效方法是使用类似树结构的多层网格，将包含太多候选导体的单元格替换成一个第二层的网格，如图3.4所示。第二层网格中的每个单元格可以进一步的被替换为第三层的网格，然而实际中很少有这样的需要，并且网格层数太多也会影响查询的效率。因此在本文的实现中，限制网格最多有两层。

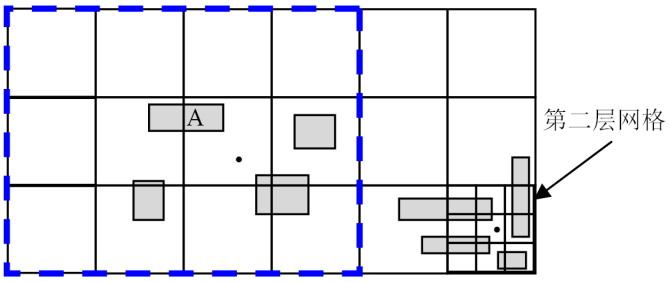


图 3.4　网格索引结构示例

在建立网格索引的过程中，想要找到一个单元格的临近区域中的所有导体并不是件容易的事情。因此根据定义3.8，考虑将每个导体插入到其临近区域内的单元格中。如图3.4所示，导体 A 会被尝试插入蓝色虚线内的单元格中。算法7描述了将一个导体插入网格索引的过程。将所有导体依次插入即可建立整个网格结构。

算法7中包含两个参数，$threshold_s$ 确定网格单元的大小，阈值 $threshold_l$ 限制第一层网格的最大候选导体数量，候选导体多于这个阈值的单元格会被换为第二层网格。

网格结构的另一个特点是，当找到包含查询点的单元格时，自然也找到了与其相邻的单元格。这意味着可以不使用候选导体列表，而是每个单元格只记录与其相交的导体（即将 l_T 初始化为 0）。这样做能大幅度地加快网格结构的建立，并且能减小内存占用。但在查询时，需要不仅仅检查包含查询点的单元格，还需要检查与其相邻的 26 个（三维情形）单元格。考虑到相邻的单元格往往会包含同一个导体，这意味着在查询过程中会重复尝试同一个导体。同时，这样的网格结构产生的转移区域的大小与将 d_{nb} 设定为单元格大小 $size_T$ 的使用不完整候选导体列表的网格是一样的。综合来看，这种不使用候选导体列表的结构会拖慢行走过程的效率。

算法 7 将导体插入网格索引

输入： 导体 A，网格 G，单元格大小 $threshold_s$

输出： 更新网格

for $w \in \{x, y, z\}$ **do**
    $w_1 \leftarrow \lfloor \max((w_{l_A} - w_{l_G})/threshold_s, 1) \rfloor$
    $w_2 \leftarrow \lceil \min((w_{r_A} - w_{l_G})/threshold_s + 1, (w_{r_G} - w_{l_G})/threshold_s) \rceil$
end for
for $i = x_1, x_2, j = y_1, y_2, k = z_1, z_2$ **do**
    单元格 $T \leftarrow G[i, j, k]$
    if T 是第二层网格 **then**
        递归调用本算法将 A 插入 T 中
    else if T 位于第一层 $\wedge length(l_T) \geq threshold_l$ **then**
        将 T 替换为第二层网格 T'
        for $B \in list_T \cup \{A\}$ **do**
            递归调用本算法将 B 插入 T' 中
        end for
    else
        调用算法4将 A 插入 $list_T$ 中
    end if
end for

3.3.3　网格与八叉树混合索引结构

注意到在定义3.7中，门限距离中一部分 $size_T$ 是单元格 T 的长宽高中的最大值。因此，当单元格的形状与立方体差别很大时，门限距离的筛选效果将大大折扣。同时，如果按前文所讨论，设定 $d_{nb} = size_T$，那么临近区域也将远远大于空间单元本身。因此希望每个包含候选导体列表的空间单元都尽可能接近立方体。

一般的，空间管理结构只需要索引问题区域。通常情况下，问题区域是一个扁平的长方体，其在水平方向的尺度会被竖直方向大很多，差距可达 2~3 个数量级（数字电路）甚至 5~6 个数量级（触屏电路）。对于网格结构，只要设定各个方向上的划分步长相等，那么它只会影响不同方向的单元格的数量，因此并不是问题。

但是对于八叉树结构，由于其每个单元格在分裂时是均匀地分为八个与原单元格相似的单元格，其长宽比将从根节点即问题区域一直继承到每个叶子节点。这不是希望的结果。一种解决方案是选择一个能包含整个问题区域的立方体作为八叉树的根节点，但是这样整个空间管理结构将需要索引远大于实际问题区域的

无用的空间。另一种解决方案就是利用将网格结构，先将整个问题空间划分成为小立方体，再为每个立方体建立八叉树，这样就能保证每个节点都是立方体。

注意这里使用的网格的主要目的是保证每个单元格是立方体，对单元格的大小没有要求，因此网格的单元格应尽量大一些。这样的好处在于一方面网格较大的意味着网格数量较少，可以减少存储开销；另一方面，越大的网格的候选导体列表越可能是完整的。在实现中，本文选择以问题区域的长宽高中最小的值或其一半作为第一层网格的大小。

在查询时，首先找到包含查询点的网格节点，然后再在它对应的八叉树中找到包含查询点的叶子节点。与其它结构相比，混合结构可以使用更小的内存，并提供更快的建立和查询速度。

3.4　实验结果和分析

本文在 RWCap[18] 的基础上实现了本章提出的各种算法，并利用多个 45-nm 或 180-nm 工艺 [33] 的大规模 VLSI 结构验证它们的正确性与效率。所有的实验都是在一台使用 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。

本节将主要使用四个测试用例：

- 测例 1：1000×1000 交叉线结构。包含 2000 个导体块，每个导体块的宽度和间距均为 14nm，高度为 28nm，层间距为 86nm。
- 测例 2：被称为“FreeCPU”的真实设计。基于 180-nm 工艺，最小线宽（w_{min}）为 200nm。包含 3036 个线网，37062 个导体块，5 个金属层。外形尺寸大约为 700μm×700μm×9.4μm。图3.5 展示了它的部分结构。
- 测例 3：基于 45nm 工艺的人造例子。包含 101595 个随机尺寸（最小线宽 70nm）、随机位置的导体块，3 个金属层。外形尺寸约为 1000μm×1000μm×0.6μm。
- 测例 4：一个更大的 180-nm 工艺的人造例子。包含 484441 个导体块，导体尺寸与分布与测例 2 类似。

3.4.1　验证空间单元加速算法

这一小节使用八叉树结构来验证3.2节中提出的加速技术。在这些实验中，候选导体最大数量 $threshold_l$ 限制为 21，叶子节点最小尺寸 $threshold_s$ 限制为 40 倍最小线宽。对于不完整候选导体列表技术，限定邻近距离 d_{nb} 为 25 倍线宽，这样得到的候选导体列表几乎是完整的。

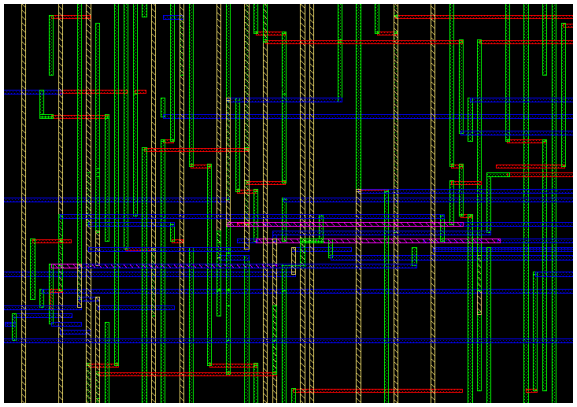


图 3.5　FreeCPU 的局部二维展示，不同层的导体由颜色区分

FRW 算法的完整计算时间包含两部分：空间管理结构建立时间和随机行走过程的时间。文献 [17] 中提出的八叉树结构记为 $Octree_O$，使用门限距离优化的版本记为 $Octree_D$，同时使用门限距离和不完整候选导体列表的版本记为 $Octree_I$。表3.1列出了三个不同版本的八叉树的建立时间。

表 3.1　不同技术下八叉树结构的建立时间的比较

测例	导体块数目	建立时间 (s)			加速比
		$Octree_O$	$Octree_D$	$Octree_I$	
1	2000	81.3	0.36	0.29	280
2	37062	1757.9	3.10	0.74	2375
3	101595	16595.6	8.21	2.43	6829
4	484441	–	83.12	17.12	–

从表3.1可以看出，门限距离限制可以带来巨大的加速，并且例子中包含的导体数量越多，其加速比越高。不完整的候选导体列表可以进一步减少建立时间（加速 4 倍以上）。综合使用这两种加速方法，包含 101595 块导体的测例 3 的八叉树的建立过程可以加速 6829 倍，建立时间变为 2.4 秒。对于最大的例子，$Octree_O$ 无法在 1 天内建立八叉树，而 $Octree_I$ 只需要 17 秒。

为了验证提出的技术在空间管理的查询阶段的效率，对每个例子随机提取了 100 个线网，提取精度设置为 1-σ 误差不大于 0.5%。正如所期望的，门限距离对查询时间没有影响，不完整候选导体列表只有很小的影响。表3.2列出了排序候选导体列表的试验结果，其中原 FRW 算法用 FRW_O 表示，使用了排序候选导体列表的算法用 FRW_S 表示。数据表明，对于较大的例子，排序候选导体列表能加速 2 倍以上。

为了了解不完整的候选导体列表技术中邻近距离 d_{nb} 对算法效率的影响，图

表 3.2　排序候选导体列表对随机行走过程的加速

测例	采样次数	跳转次数	提取时间 (s)		加速比
			FRW_O	FRW_S	
1	28.2	9.1	3.04	1.61	1.89
2	30.3	12.5	2.96	1.41	2.10
3	18.9	10.5	1.72	0.81	2.12
4	27.7	12.7	3.63	1.49	2.44

中3.6画出了建立时间、平均每百万次采样时间、平均每百万次跳转时间和平均跳转次数与 d_{nb} 的关系曲线。其中横轴均为 d_{nb}/w_{min} 以 5 为底的对数坐标（w_{min} 为最小线宽）。如图3.6(b-d) 所示，当 $d_{nb} > 25w_{min}$ 时，其值对随机行走过程几乎没有影响。随着 d_{nb} 减小，每次采样的跳转次数迅速增大，尽管每次跳转的时间因为候选导体列表的缩短而减少，但是总体的每次采样时间会增加。因此选择 $25w_{min}$ 作为 d_{nb} 的最优值，它能显著的减少建立时间，同时对随机行走过程影响很小。

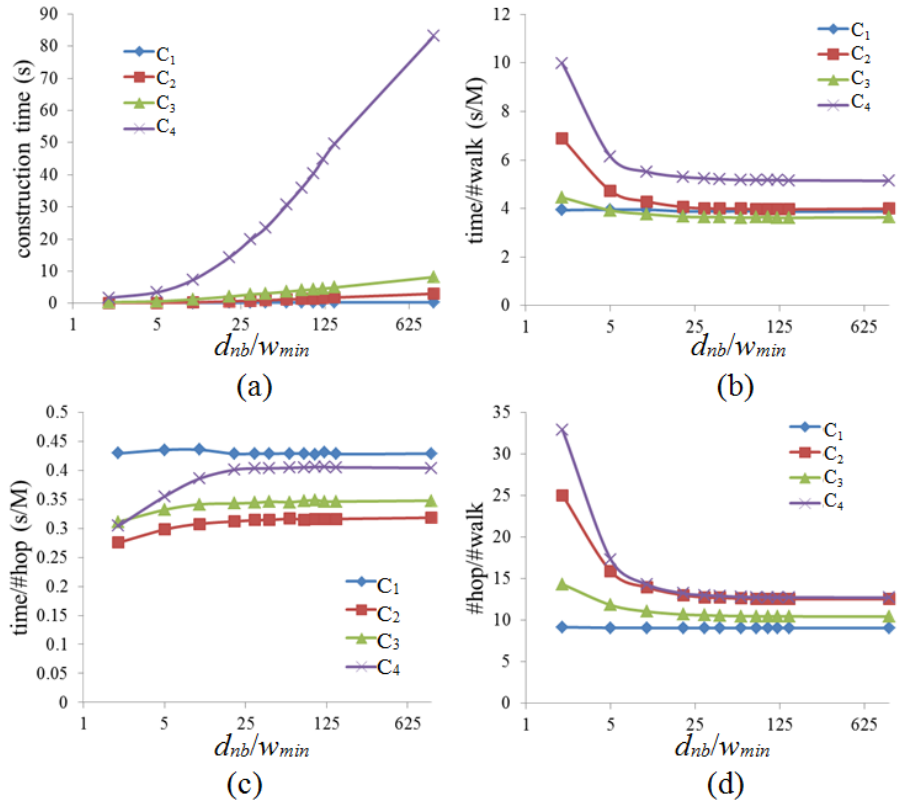


图 3.6　建立时间 (a)、平均每百万次采样时间 (b)、平均每百万次跳转时间 (c) 和平均跳转次数 (d) 与 d_nnb 的关系曲线

3.4.2　对比不同索引结构

对于3.3节中提出的四种索引结构，每种都有一些参数，列在3.3中。其中，对于八叉树和混合结构将 d_{nb} 设为 $25w_{min}$；对于混合结构，将第一层网格大小设为问题空间长宽高的最小值[①]，这样使得网格实际上是二维网格。对于网格结构，设定 d_{nb} 与单元格的大小相同。这样每种索引结构都只剩下两个参数，虽然它们的语义在不同的索引结构中略有不同，但是简单起见统一用 $threshold_s$（或 t_s）和 $threshold_l$（或 t_l）分别表示它们。注意 $threshold_s$ 作为描述单元格尺寸的变量，也应该像 d_{nb} 那样用 w_{min} 的倍数来表示，以使得它能更好的适应工艺的变化。

表 3.3　四种空间单元索引结构的参数

结构	参数说明	变量
八叉树结构	叶子节点最小尺寸	t_s
	最大候选导体列表数	t_l
	邻近区域距离	d_{nb}
网格结构 *	第一层网格大小	t_s
	第一层最大候选导体列表数	t_l
混合结构	叶子节点最小尺寸	t_s
	最大候选导体列表数	t_l
	邻近区域距离	d_{nb}
	第一层网格大小	-

* 包含使用和不使用候选导体列表的两种网格结构。

对于不同的索引结构，通过实验研究了空间管理建立时间、随机行走时间和存储使用量随着 t_s 和 t_l 的变化。图3.7画出了八叉树结构和使用候选导体列表的网格结构对于测例 2 的空间管理建立时间。从图中可以看出，对于八叉树结构，建立时间随着 t_s 或 t_l 的增加而减少，这是因为当 t_s 或 t_l 较大时，整个八叉树包含的空间单元数较少。

网格结构的情形则相对复杂一些，较小的 t_l 意味着第一层的单元格有较大的可能被分裂为第二层网格，从而增加建立时间。而当 t_s 足够大时，每个第一层单元格几乎都会包含足够多的候选导体，进而分裂成第二层网格，因此建立时间不再随着 t_l 变化，但是较大的 t_s 也会使得建立时间较长。当 t_s 较小时，建立时间随着 t_l 的减小而快速增加，因为更多的第一层单元格被分裂为第二层网格。然而由于较小的 t_s 意味着较小的邻近区域，所以如图3.7(b) 所示，最小的 t_s 并没有带来

① 对于这里的四个例子，均为高度。

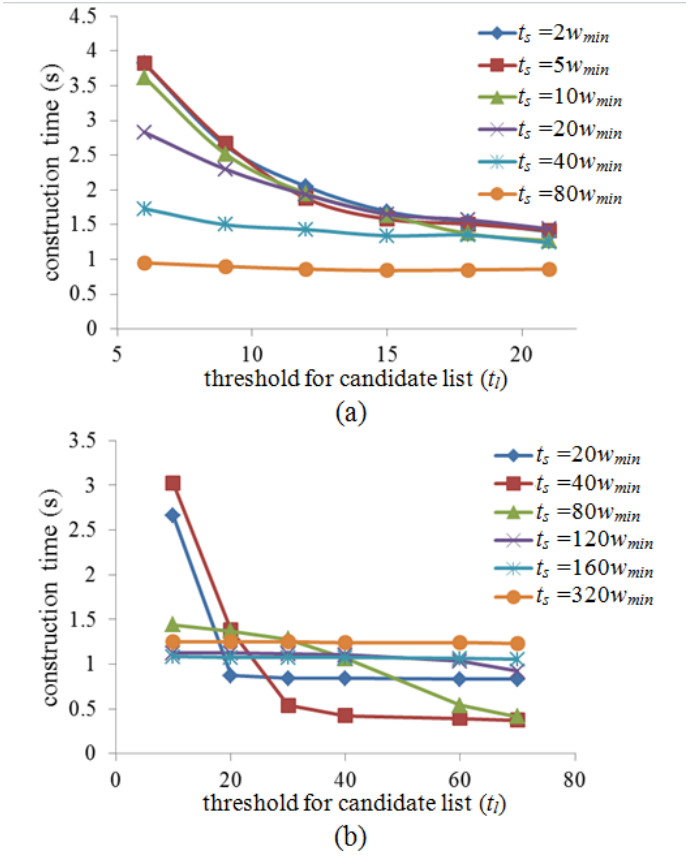


图 3.7　八叉树 (a) 和使用候选导体列表的网格 (b) 的建立时间趋势

最大的建立时间。

对于不包含候选导体的网格和混合结构，它们的建立时间的趋势与图3.7(a) 相似，但是时间要小的多，大部分小于 1 秒。

图3.8展示了八叉树结构和不使用候选导体列表的网格结构在随机行走过程中平均每百万个采样需要的时间随着 t_s 和 t_l 的变化趋势。从中可以看出，对于八叉树结构，随机行走时间随着 t_l 的增加几乎不变或有微小的增加。这是因为尽管较大的 t_l 增加了遍历候选导体列表的时间，但是减小了八叉树的高度。对候选导体列表的排序也有助于减小遍历时间。对于 t_s 而言，它只在非常大以使得叶子节点的候选导体列表非常长时才有影响。对于不使用候选导体列表的网格结构，随机行走过程的时间随着 t_l 或 t_s 的增加而增加。

对比图3.8(a) 和 (b)，显然的，不使用候选导体列表的网格会导致随机行走的时间长很多。对于使用候选导体列表的网格和混合结构，随机行走时间的变化趋势与八叉树相似，前者的时间也与八叉树相似，后者则更快一些。

这四种索引结构的存储使用量随 t_s 和 t_l 的变化趋势是相同的，当 t_s 或 t_l 增加时，存储使用量减少，这与图3.7(a) 中的建立时间相似。对于相同的空间单元大小，不使用候选导体列表的网格结构消耗的存储最小。

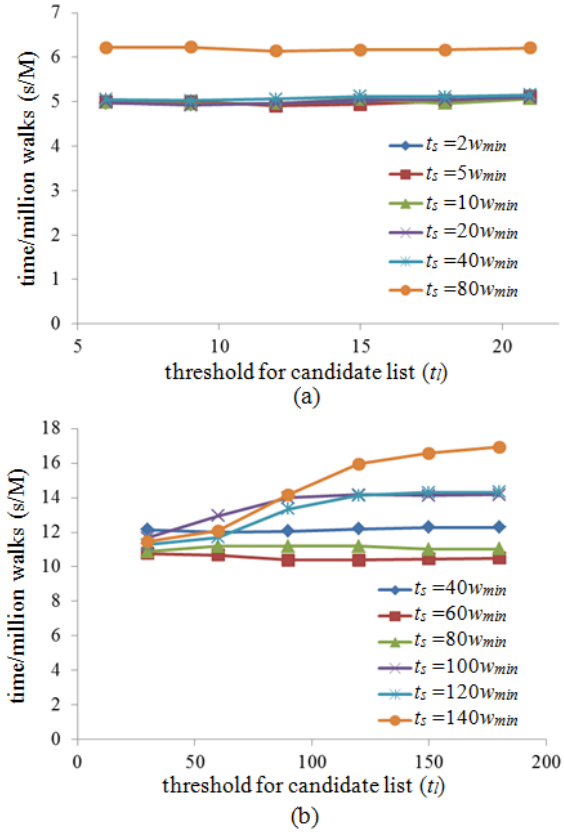


图 3.8　八叉树 (a) 和不使用候选导体列表的网格 (b) 的随机行走时间趋势

表 3.4　四种索引结构的存储使用量与随机行走时间

索引结构	$t_s(w_{min})$	t_l	存储使用量 (MB)	随机行走时间 (s/M)
八叉树结构	80	12	17	6.14
	10	18	37	4.97
有候选导体列表的网格	320	70	22	5.48
	160	20	50	4.98
无候选导体列表的网格	60	120	20	10.38
混合结构	1	21	19	4.84
	16	21	18	4.96

对于每种索引结构，都有一个存储消耗和随机行走速度的折中。表3.4列出了四种索引结构在不同的 t_s 和 t_l 设定下的存储使用量和随机行走时间。从中可以看出，相比于其它三种结构，混合结构更有优势。当存储使用量相似时，相比于八叉树或使用候选导体列表的网格，混合结构可以将随机行走时间减小 12%，并且相比与不使用候选导体列表的网格减少超过 50%。当随机行走速度相同时，混合结构使用存储比其它结构少一半以上。对于其它几个例子也能得到相似的结论。

3.4.3　与 RWCap 和 Rapid3D 的综合对比

本文在 RWCap 的基础上实现了改进的网格八叉树混合结构。改进后的程序称为 RWCap2。首先将 RWCap2 与 RWCap 进行对比。表3.5列出了初始化（建立空间管理结构）的时间和提取 1 个线网和 100 个线网的时间。这里没有使用测例 4，因为 RWCap 无法对其建立空间管理结构。从表中可以看出，当只需要提取 1 个线网时，RWCap2 相比于 RWCap 加速能达到 7829 倍，当提取 100 个线网时，加速比仍然可以达到 231。将表3.5与表3.1和3.2相比，也能看出混合结构相比于简单的八叉树的优势。

表 3.5　RWCap2 和 RWCap 的比较

	RWCap				RWCap2					
测	建立	存储	提取时间 (s)		建立	存储	提取时间 (s)		加速比 *	
例	时间 (s)	+(MB)	1 个线网	100 个线网	时间 (s)	+(MB)	1 个线网	100 个线网	1 个线网	100 个线网
1	81.3	7	3.00	304.1	0.13	6	1.17	118.9	65	3.2
2	1758	29	1.80	296.7	0.34	17	0.80	132.0	1544	15.5
3	16596	113	1.78	172.6	1.37	87	0.75	71.2	7829	231

+ 空间管理结构的存储使用量。

* 基于总提取时间。

表 3.6　RWCap2 和 Rapid3D 的比较

测例规模	Rapid3D		RWCap2			
	电容 (fF)	时间 (s/M)	电容 (fF)	误差	时间 (s/M)	加速比
100×100	120.3	10.61	120.4	0.1%	3.85	2.8
500×500	598.9	12.91	597.5	-0.2%	3.73	3.5
800×800	952.0	13.44	955.8	0.4%	3.78	3.6
1000×1000	1196.6	14.80	1192	-0.4%	3.88	3.8

最后，本章提出的 RWCap2 与 Rapid3D[34] 进行了对比。这里使用了几个不同规模的交叉线结构作为测例，它们与测例 1 结构相似，但导体数不同。对于每个例子，以 0.5% 的误差阈值提取了中间金属层的中间导体的电容。由于 RWCap2 还使用了减少采样数的技术，因此为了公平起见，只比较平均每次采样时间。表3.6列出了提取的电容结果和时间。从结果来看，RWCap2 使用的空间管理技术能带来超过 3 倍的加速，并且这个加速随着 VLSI 结构的增大还能继续增加。

本节的实验均使用了单介质的例子。需要指出的是，这里对于空间管理结构

的改进和优化都是与介质无关的，它们可以直接被应用于多介质的情形，详细的实验结果请参考文献 [35]。

3.5　本章小结

本章主要介绍了对空间管理结构的优化，主要分为三个方面。首先利用门限距离加快了候选导体列表的建立过程，其次通过对候选导体列表进行排序，使得查找最近导体的过程无需遍历整个候选导体列表。最后对比和改进了多种空间单元的索引结构，并提出了新的网格-八叉树混合结构来充分利用两种结构的优点。实验表明，对于一个包含近 50 万导体块的结构，混合结构的建立只需 17 秒。同时，改进的空间管理结构对于随机行走过程也能加速 2 倍。

第 4 章　面向全线网电容提取的高斯面生成和采样技术

本章将主要介绍针对全线网的高斯面建立及采样过程。4.1节主要介绍全线网电容提取的目的和实现方式。4.2节主要介绍如何生成单个导体块的高斯面以及如何优化高斯面的位置以减小误差。4.3节主要介绍如何在不实际生成全线网的高斯面的情形下直接在上面进行采样。最后在4.4节将通过多个例子来验证本章算法的正确性以及对比调整算法参数从而提高效率。

4.1　全线网电容提取的基本概念和方法

为了准确地计算整个信号线由于电阻和电容效应带来的延迟，一般将包含垂直通孔在内的整个线网分割成较小的导体块。通过提取由导体块组成的电阻-电容网络的参数，来得到整个线网的延迟 [15]。其中电阻的计算相对简单并且精度较高，因此主要的挑战在于电容的提取。然而现有的 FRW 电容提取算法只能针对单个导体块进行提取，无法很好地适应具有复杂连接关系的线网，不能满足得到整个线网的电容并将其合理地分配于各个导体块上的需求。

为了得到由多个导体块组成的整个线网的电容，需要构造一个包含整个线网的高斯面，并以它作为随机行走的起点。为了将得到的电容分配到每一小段导体块上，需要将整个线网的高斯面分配给各个导体块。以一个导体块所拥有的高斯面为起点的随机行走得到的电容被看做是这个导体块的电容。

使用这种方法的一个问题是，每个导体块的电容的相对误差将大于整个线网的电容的相对误差。然而由于统计误差的相互抵消，基于这种方法得到的线网延迟的误差与线网总电容的误差是大致相当的[①]。从一些例子中可以看出，当全线网总电容的统计误差是 ±2% 时，基于 RC 网络得到的 Elmore 延迟的误差只有 ±2.3%[15]。因此只需要控制线网总电容的提取精度就可以保证需要的延迟的精度。

生成整个线网的高斯面并不是非常容易的。一种直观的方法是为每个导体块生成一个高斯面，再将它们合并起来，以包络面作为整个线网的高斯面。然而由于一条线网往往是由大量导体块和通孔通过复杂关系连接成的，计算包络面将涉及大量复杂的几何计算。

注意到，在 FRW 过程中，实际上并不需要准确地知道高斯面的所有几何参数，

① 电阻的误差远小于电容的误差，因此可以忽略。

需要的只是在高斯面上随机采样。因此，本文提出一种虚拟高斯面采样（VGSS）技术，可以在不计算包络面的情况下在整个线网的高斯面上进行采样。

4.2　单个导体块的高斯面优化

4.2.1　导体块高斯面生成

先来考虑生成单个导体块的高斯面（BGS），在这个过程中与主导体在同一个线网中的导体应当被忽略。本章只考虑曼哈顿形状的导体块，因此导体块 B 的高斯面 G_B 自然是一个包围 B 的曼哈顿长方体。为了更好的描述导体块的高斯面，可以使用如下的定义：

定义 4.1：　从长方体 A 到 B[①] 的 $+x$ 方向距离：$dist_{+x}(A,B) \leftarrow x_{l_B} - x_{r_A}$；
从 A 到 B 的 $-x$ 方向距离：$dist_{-x}(A,B) \leftarrow x_{l_A} - x_{r_B}$；
从 A 到 B 的 $+y,-y,+z,-z$ 方向距离的定义类似。

利用定义4.1，可以使用在六个方向上从 B 到 G_B 的相应平面的距离来描述 G_B。

得到一个正确的高斯面是很容易的，只需要在六个方向上计算出从主导体到其它导体的最近距离，使用任何一个大于 0 但小于这个最短距离的值来定义 G_B 都是可以的。算法8描述了这个过程。

算法 8 生成单个导体块的高斯面

输入： 主导体 B，其它导体列表 Cs

输出： 高斯面 G_B

1: **for** $s \in \{+x,-x,+y,-y,+z,-z\}$ **do**
2: 　$d_s \leftarrow +\infty$
3: 　**for** $C \in Cs$ **do**
4: 　　**if** $dist(B,C) = dist_s(B,C) \wedge dist(B,C) < d_s$ **then**
5: 　　$d_s \leftarrow dist(B,C)$
6: 　　**end if**
7: 　**end for**
8: 　确定合适的数值 d'_s 并满足 $0 < d'_s < d_s$
9: **end for**
10: $G_B \leftarrow \{\{x_{l_B} - d'_{-x}, y_{l_B} - d'_{-y}, z_{l_B} - d'_{-z}\}, \{x_{r_B} + d'_{+x}, y_{r_B} + d'_{+y}, z_{r_B} + d'_{+z}\}\}$
11: **return** G_B

① 长方形平面和点可以看做是长方体在某些方向长度为 0 的特例。

4.2.2　导体块高斯面位置优化

高斯面的位置会影响到 FRW 算法的效率。在文献 [13] 中，每个方向的高斯面都被设置在主导体和这个方向上最近的导体的中间，即

$$d'_{s-1} \leftarrow \frac{d_s}{2}。\tag{4-1}$$

这样的好处是高斯面上的每一点到主导体和另外一个最近导体的距离相同，因而第一个转移区域可以同时与主导体和另外一个临近的导体相贴，使得第一次跳转就能结束这次随机行走的概率增加，从而减少平均每次行走所需的跳转次数 N_{hop}。然而这样的方法不适合于主导体附近导体分布较稀疏的情形，此时高斯面在各个方向上到主导体的距离的差别会非常大。根据公式 (2-25)，第一步的转移区域的大小会影响采样结果，从而会减慢整个随机行走的收敛速度。

文献 [18] 使用了六个方向距离的最小值的一半即

$$d'_{s-2} \leftarrow \frac{\min(d_{+x}, d_{-x}, \ldots, d_{-z})}{2}。\tag{4-2}$$

这样的好处是不同方向的高斯面与主导体的距离完全相同。然而这样又会导致有时高斯面与主导体距离太近，转移区域太小，从而一次行走所需的跳转次数增多，同样不利于整个算法的效率。

综合这两种方案，提出高斯面放缩因子 $factor_G$，这样高斯面到主导体的距离可以如下计算：

$$d'_s \leftarrow \frac{\min(\min(d_{+x}, d_{-x}, \ldots, d_{-z}) * factor_G, d_s)}{2}。\tag{4-3}$$

可以看出，式 (4-1) 是式 (4-3) 在 $factor_G = +\infty$ 时的情形，式 (4-2) 则是 $factor_G = 1$ 时的情形。$factor_G$ 的值越大则不同方向上高斯面与主导体的距离的差别越大。考虑到高斯面不应该太小，因此限定 $factor_G \geq 1$。

4.3　全线网的虚拟高斯面采样算法

4.3.1　虚拟采样技术

对于组成一条线网的每个导体块，都可以按照算法8生成它的高斯面（BGS）。此时，所有的导体块的高斯面有可能是相交的。这使得导体块的高斯面上有一些位置是不能作为采样点的。虚拟高斯面采样技术的核心思想是在所有导体块的高斯面上采样，然后舍弃不合法的点。这样保留的采样点就可以看做是从线网的高斯面上得到的。这样就能避免计算所有导体块的高斯面的包络面，从而简化算法。

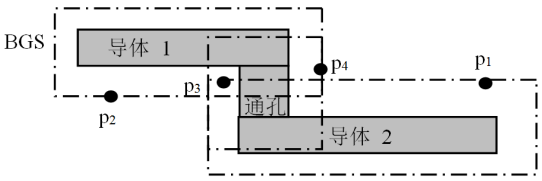


图 4.1　两个由通孔连接的导体块的高斯面

以图4.1为例，其中如点 p_1 和 p_2 这样只在一个导体块的高斯面表面上的点是合法的点，像 p_3 这样在另一个导体块的高斯面的内部的点是非法的。像 p_4 这样同时在多个导体块的高斯面上的点也是合法的，但是当期望得到均匀采样时，p_4 只能以概率 $\frac{1}{2}$ 被接受。这样就实现了在线网的高斯面上进行均匀采样。

在高斯面上的采样并不一定要求是均匀的。根据式 (2-12)，可以任意定义在高斯面上采样的概率函数 $f(r)$，从而利用重要性采样技术来加快算法的收敛。为了实现这一点，需要对均匀采样的结果再做一次筛选。对于给定的采样概率函数 $f(r)$，以及常数 U 满足 $U \geq f(r)$，每个均匀采样得到的点 r 以概率 $\frac{f(r)}{U}$ 被接受，否则需要重新生成一个采样点。

4.3.2　采样概率和权值

在式 (2-25) 中，需要计算 $f(r)$ 在整个线网的高斯面 G 上的积分 F。在使用虚拟高斯面采样技术的情况下，并没有实际计算出 G，也不能直接计算积分。因此考虑再次使用蒙特卡洛随机采样 [19] 来计算 F 的值。可以选择最简单的均匀采样计算 F，根据式 (2-4) 有

$$F \approx |G|\bar{f}(r) = |G|\frac{\sum f(r_j)}{N}, \tag{4-4}$$

然而由于 G 未知，其面积 $|G|$ 也是未知的。再次使用蒙特卡洛随机采样，定义指示函数

$$g(r) = \begin{cases} 1 & \text{if } r \in G \\ 0 & \text{otherwise} \end{cases} 。 \tag{4-5}$$

这样有

$$|G| = \oint_G g(r)dr = \oint_{\oplus G_{B_i}} g(r)dr, \tag{4-6}$$

其中 G_{B_i} 表示导体块 B_i 的高斯面，$\oplus$ 表示不去除重复元素的集合并运算，如 $\{a,b\} \oplus \{b,c\} = \{a,b,b,c\}$。于是

$$|G| \approx |\oplus G_{B_i}|\bar{g}(r) = \sum |G_{B_i}|\frac{N_G}{N_T}, \tag{4-7}$$

其中 N_T 表示在所有导体块的高斯面（$\oplus G_{B_i}$）上的总采样数，N_G 是落在线网高斯面（$G = \bigcup G_{B_i}$）上的采样数。将式 (4-7) 代入式 (4-4) 即可得到

$$F \approx \sum |G_{B_i}| \frac{\sum f(r_j)}{N_T}。 \tag{4-8}$$

为计算式 (4-8)，需要首先在 $\oplus G_{B_i}$ 上进行均匀采样，然后对其中落在 G 上的点 r_j 计算 $\sum f(r_j)$。这个过程与虚拟高斯面采样的过程是一样的，因此可以在随机行走过程结束以后利用采样情况来计算 F，这样并不会带来很多额外的计算。同时由于计算 F 的准确度远远高于计算电容的准确度，因此这里的近似带来的误差也可以忽略。

如算法9的描述，虚拟高斯面采样实际是一个拒绝采样 [28] 的过程。当在所有的导体块的高斯面 $\oplus G_{B_i}$ 上采样时，得到的采样点可以看做在 $\bigcup G_{B_i}$ 上以 $p_0(r) = \frac{n_c(r)}{N_c}$ 为概率密度函数的采样，其中 $n_c(r)$ 表示点 r 在 $\oplus G_{B_i}$ 中的出现次数，$N_c = \oint_{\bigcup G_{B_i}} n_c(r)dr$。在虚拟高斯面采样中，每个采样点被接受的概率为

$$p_{ac}(r) = g(r)\frac{1}{n_c(r)}\frac{f(r)}{U}。 \tag{4-9}$$

根据拒绝采样的原理 [28]，实际得到的采样概率函数为

$$p_f(r) \propto p_0(r)p_{ac}(r) = \frac{g(r)f(r)}{UN_c} \propto \begin{cases} f(r) & \text{if } r \in G \\ 0 & \text{otherwise} \end{cases}, \tag{4-10}$$

即在 G 上正比于 $f(r)$，否则是 0，这正是期望的采样。

在虚拟高斯面上采样时，有些采样点可能会被抛弃，这使得整个采样过程可能会重复多次才能得到一个最终可以接受的点。不过考虑到对于每次行走，在高斯面上采样所需的时间远小于整个随机行走的时间（大约 1%），只要尝试次数不太多就不会影响算法的效率。实践表明，即使是使用非常不均匀的采样概率密度函数，平均的重复次数也不超过 5 次。

关于采样概率函数 $f(r)$ 的选择，根据公式 (2-25)，权值函数包含三个变量：采样点的介电常数 $\varepsilon(r)$，第一个转移区域的边长 $L(r)$ 和第一个转移区域上的积分 $H_u(r)$，其中 $H_u(r)$ 只与 $\hat{n}(r)$ 有关，因此在使用分层采样技术后，在同一个积分区间内可以是常数，并且在不同积分区间之间的值差别也很小。因此希望 $f(r)$ 的选择能尽量消除剩余的权值 $\frac{\varepsilon(r)}{L(r)}$ 带来的变化。其中 $L(r)$ 无法预先知道，因此使用 $D(r) = dist(r, B)$ 即采样点到主导体块的距离来近似。这样，就有了 $f(r)$ 的四种选择：

$$f_0(r) \quad = \quad 1 \tag{4-11}$$

算法 9 虚拟高斯面采样过程

输入： 所有主导体块 Bs，采样概率 $f(r)$，总采样数 N_T，$f(r)$ 的和 S_f

输出： 采样点 r，更新 N_T 和 S_f

1: 根据面积随机选择一个导体块 B_i
2: 在 G_{B_i} 上均匀的随机选择一个点 r
3: $N_G \leftarrow N_G + 1$
4: **if** r 在某个 G_{B_j} 内部 **then**
5: 　　**go to**1
6: **else if** r 在某个 G_{B_j} 的表面，并且该面与选择 r 的面的法向相反 **then**
7: 　　**go to**1
8: **end if**
9: $n_c(r) \leftarrow \oplus G_{B_i}$ 中 r 的出现次数
10: **if** $random() > \frac{1}{n_c(r)}$ **then**　　▷ $random()$ 每次返回一个 [0, 1] 区间内的随机数 [36]
11: 　　**go to**1
12: **end if**
13: $S_f \leftarrow S_f + f(r)$
14: **if** $random() > \frac{f(r)}{U}$ **then**
15: 　　**go to**1
16: **end if**
17: **return** r

$$f_1(r) = \varepsilon(r) \tag{4-12}$$

$$f_2(r) = \frac{1}{D(r)} \tag{4-13}$$

$$f_3(r) = \frac{\varepsilon(r)}{D(r)} \tag{4-14}$$

下一节将通过实验来选择对性能最有利的 $f(r)$。

4.4　实验结果和分析

本文在 RWCap2 中实现了本章中提出的虚拟高斯面采样技术。首先将通过一个人造的测例 5 来验证虚拟高斯面采样技术的正确性。

- 测例 5：人造复杂结构。如图4.2所示，主导体包含 11 个导体块和 6 个垂直通孔，周围有 12 个导体块，三个金属层。

对于测例 5，可以手工生成它的真正的高斯面（由矩形片来描述）。同时在虚拟高斯面和真实的高斯面上进行采样和随机行走。结果表明，两种不同的方法得

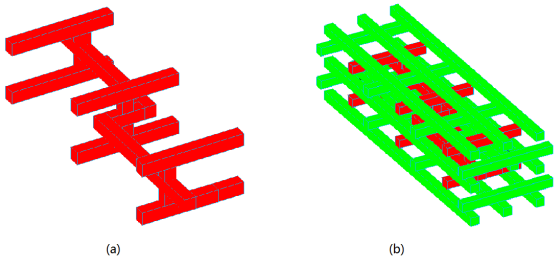


图 4.2　测例 5 的主导体 (a) 和周围导体 (b)

到的电容值误差小于设定的误差限（0.5%），并且两个随机行走过程的采样数也几乎相同。

4.2节提出的 $factor_G$ 决定着高斯面的位置，并且同时影响着达到指定精度的采样数和平均每次采样的跳转次数。可以通过尝试各种不同的值来找到最优的设定。另外4.3 节中提出的不同的在高斯面上进行采样的概率密度函数的影响也需要实验的比较。这里，使用了测例 2 和 3 作为测试例子，分别在其中随机选择 33 条线网进行 1-σ 误差限为 0.5% 的提取。图4.3展示了随机行走过程的时间。

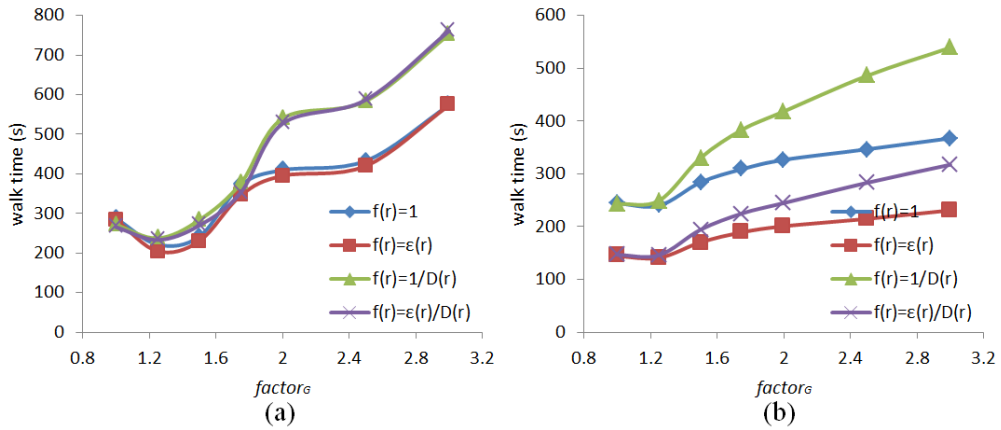


图 4.3　使用不同 $factor_G$ 和采样 PDF 时随机行走过程的时间，测例 2(a)，测例 3(b)

从图4.3中可以看出，设置 $factor_G = 1.25$ 和 $f(r) = \varepsilon(r)$ 能带来最好的性能。更大的 $factor_G$ 会增加收敛所需的采样数，因为 $L(r)$ 的变化范围增加了。同时较小的 $factor_G$ 会导致平均每次采样所需的跳转次数增加。如图4.3(a) 所示，与文献 [18] 使用的 $factor_G = 1$ 相比，优化的高斯面位置能带来 1.4 倍加速。如果与文献 [13] 使用的 $factor_G = +\infty$ 相比，加速则更大。

至于高斯面上的采样概率密度函数，使用 $f(r) = \varepsilon(r)$ 相比于其它的优势非常明显。对于测例 3，使用 $f(r) = \varepsilon(r)$ 时采样数为 245182，如果使用 $f(r) = 1$ 则采样数为 422909，相比之下会大 72%。

最后统计了由于拒绝采样造成的重新采样的次数。结果表明，即使是使用最不均匀的概率密度函数，采样的重复次数也不超过 5 次。这意味着重复采样带来

的额外开销相对于随机行走过程是可以忽略的。

4.5　本章小结

本章讨论了如何为由多个导体块和通孔组成的完整线网生成高斯面并进行采样。通过高斯面放缩因子 $factor_G$ 灵活控制高斯面的位置从而取得最高的采样次数和跳转次数的平衡。利用拒绝采样原理，直接在导体块高斯面上进行采样，避免复杂的包络面积算。同时充分利用重要性采样来减少收敛所需的采样次数。实验表明，对于包含数十个导体块和通孔的线网，利用虚拟高斯面采样技术，随机行走算法能迅速并且准确地提取其电容，同时优化的高斯面位置和采样概率密度函数能带来 1.5 倍加速。

第 5 章　面向非曼哈顿结构的电容提取技术

本章将主要介绍如何扩展现有的悬浮随机行走算法，使它能够提取包含非曼哈顿导体的结构的电容。5.1节主要介绍两种常见的非曼哈顿结构（圆柱 TSV 和倾斜导线）的物理意义以及它们的几何模型描述。5.2节主要介绍如何使用旋转的立方体转移区域以增加随机行走点落在非曼哈顿导体表面的概率。5.3节主要介绍如何将空间管理结构扩展到包含非曼哈顿导体的结构，主要研究了遮挡关系的判断。5.4节主要介绍针对 TSV 结构的高斯面位置和重要性采样的优化，以及针对倾斜导线的高斯面生成算法。最后在5.5节将通过与多种算法对比来验证本章提出的算法的正确性以及各种加速技术的效果。

5.1　非曼哈顿结构的介绍和建模

在多数传统的数字电路中，所有的导线都是由平行于坐标轴的长方体导体块组成的，这样的电路结构被称为曼哈顿结构 [20]。本文之前的章节也都是针对曼哈顿结构的电路进行的分析和讨论。然而随着工艺的发展，集成电路的结构和功能也越来越复杂。

三维芯片技术可以大幅度提高电路原件集成度，圆柱形的硅通孔（TSV）[21–24] 在三维芯片中用于连接多层电路的信号，起着至关重要的作用。它的寄生参数对信号的延迟和噪声的影响都非常明显。目前在随机行走电容提取算法中，圆柱形的 TSV 往往被正方形横截面的棱柱近似替代 [9,13,16]。然而这会引入较大的误差，如表5.1和5.2所示，对于 TSV 的总电容，近似替代可以带来约 5%~7% 的误差，而对于 TSV 与邻近的一般导线间的耦合电容，这种近似会带来超过 20% 的误差。因此有必要在随机行走过程中精确的描述圆柱形 TSV。

在许多嵌入式设备、液晶面板和数模混合电路中，有许多并不平行于坐标轴的倾斜导线 [26,27]。它们的横截面一般是平行四边形或者梯形（图1.3），很难用曼哈顿形体的近似，因此也需要精确的几何描述。目前关于将随机行走算法应用到这类包含斜线的电路中的讨论并不充分，并且没有实际可行的、可以针对较大规模结构的算法。

本章将从转移立方体、空间管理和高斯面三个方面，对一般的随机行走算法进行扩展，使它能够高效的应用于这类包含圆柱形 TSV 和倾斜导线等非曼哈顿形体的电路。在描述具体的改进前，需要先描述非曼哈顿形体的几何结构。

圆柱形 TSV A 和倾斜导体 B，均可以通过横截面（分别为圆形 R 和凸多边形 P）和顶面、底面的 z 坐标 z_t, z_b 来描述，记为 $A = (R_A, z_{t_A}, z_{b_A}), B = (P_B, z_{t_B}, z_{b_B})$。其中，圆形 R 由中心点 c 和半径 r 描述，凸多边形平面使用其 n 个逆时针排列的顶点来描述，即 $P = (p_1, p_2, \ldots, p_n)$。这里点 c, p_i 均为二维点。

5.2　使用旋转的立方体作为转移区域

5.2.1　旋转转移区域的构造

在针对一般曼哈顿形体的电路时，平行于坐标轴的立方体是作为转移区域的最好选择，因为它可以与曼哈顿形的导体贴合的更好，同时其转移概率也较容易计算。对于包含非曼哈顿结构的电路，考虑到其中曼哈顿结构导线的比例仍然很高，所以这里继续沿用立方体转移区域。

转移区域的大小是由当前点到最近的导体的距离来确定的。为了将 FRW 算法扩展到非曼哈顿形体上，首先需要将点到导体距离（即曼哈顿立方体转移区域的半边长）的定义3.2扩展到圆柱形 TSV 和倾斜导体：

定义 5.1：　点 $p = (x_p, y_p, z_p)$ 到平面圆 $R = (c, r)$ 的水平距离，记 $x = |x_c - x_p|$，$y = |y_c - y_p|$，当 $|x - y| < r$ 时：$dist_h(p, R) \leftarrow \frac{x+y-\sqrt{2r^2-(x-y)^2}}{2}$；
当 $|x - y| \geq r$ 时：$dist_h(p, R) \leftarrow \max(x, y) - r$。

定义 5.2：　点 $p = (x_p, y_p, z_p)$ 到平面凸多边形 $P = (p_1, p_2, \ldots, p_n)$ 的水平距离：$dist_h(p, P) \leftarrow \max(\max_{i=1,n}(\frac{((p-p_i)\times(p_{i+1}-p_i))\cdot e_3}{|x_{p_{i+1}}-x_{p_i}|+|y_{p_{i+1}}-y_{p_i}|}), dist(p, B_P))$，
其中，$p_{n+k} = p_k$，$e_3 = (0, 0, 1)$ 是 z 方向的单位向量，B_P 表示 P 的最小曼哈顿包围盒，其计算方法见算法11。

定义 5.3：　点 $p = (x_p, y_p, z_p)$ 到圆柱形 TSV 或倾斜导体 $C = (P_C, z_{t_C}, z_{b_C})$ 的纵向距离：$dist_v(p, C) \leftarrow \max(z_p - z_{t_C}, z_{b_C} - z_p)$。

定义 5.4：　点 p 到圆柱形 TSV 或倾斜导体 C 的距离：$dist(p, C) \leftarrow \max(dist_h(p, C), dist_v(p, C))$。

根据定义5.4，可以得到以当前点为中心恰好与一个圆柱形 TSV 或者一个倾斜导体相贴的曼哈顿立方体转移区域。这意味着，即使问题空间中存在非曼哈顿导体，也能正确地得到一个不与任何导体相交的转移区域，随机行走可以正常的进行下去。

然而，如图5.1(a)(c) 所示，曼哈顿立方体转移区域在很多情况下只能与非曼哈顿导体有一条边相接触，这意味着随机行走停在非曼哈顿导体表面的概率非常小。

这个概率在理论上是 0，但在实践中，一般认为只要当前点与导体的距离小于某个阈值 δ 就算接触到了导体 [37,38]。随机行走停止在非曼哈顿导体表面的概率过小，即随机行走有可能不断的在某个非曼哈顿导体附近反复，会大幅的增加 N_{hop} 进而严重损害整个算法的性能。

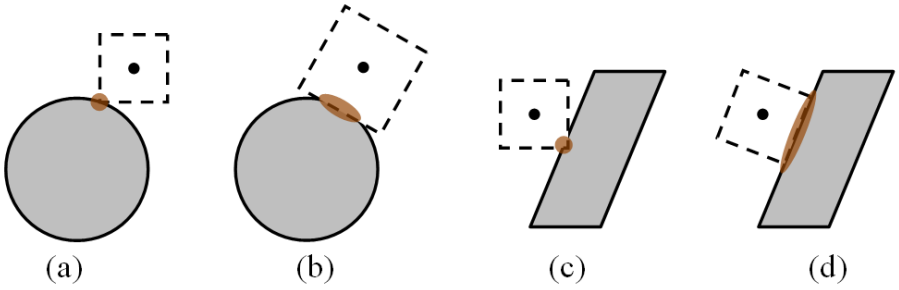


图 5.1　贴近圆柱形 TSV 和倾斜导体的曼哈顿转移立方体 (a)(c) 和旋转的转移立方体 (b)(d) 的俯视图

如图5.1(b)(d) 所示，如果将转移区域旋转到合适的角度，就能大幅度的增加转移区域与非曼哈顿导体间的接触面积。因此，当距离当前点最近[①] 的导体是非曼哈顿导体时，可以考虑使用旋转的转移立方体以增加随机行走在这一步停止的概率。为此，首先需要计算旋转的转移立方体的大小（即半边长），在这里我们称之为点到导体的旋转距离：

定义 5.5：点 p 到平面圆 $R = (c, r)$ 的旋转距离：$dist_r(p, R) \leftarrow \sqrt{(x_p - x_c)^2 + (y_p - y_c)^2} - r$；
旋转角度 θ：$\cos(\theta) \leftarrow \frac{x_p - x_c}{\|p-c\|}$，$\sin(\theta) \leftarrow \frac{y_p - y_c}{\|p-c\|}$。

定义 5.6：点 p 到平面凸多边形 $P = (p_1, p_2, \ldots, p_n)$ 的旋转距离：$dist_r(p, P) \leftarrow \max(\max_{i=1,n}(\frac{((p-p_i)\times(p_{i+1}-p_i))\cdot e_3}{\|p_{i+1}-p_i\|}), dist(p, B_P))$；
如果在 $i = j$ 取得距离的最大值，旋转角度 θ：$\cos(\theta) \leftarrow \frac{y_{p_{j+1}} - y_{p_j}}{\|p_{j+1}-p_j\|}$，$\sin(\theta) \leftarrow \frac{x_{p_j} - x_{p_{j+1}}}{\|p_{j+1}-p_j\|}$；
如果在 B_P 取得距离的最大值，旋转角度 θ：$\theta \leftarrow 0$。

5.2.2　旋转转移区域的使用条件

定义5.5和5.6给出了旋转的转移区域的大小和旋转角度，接下来讨论在什么情况下可以使用旋转的转移区域。基本原则是，只有在旋转的转移区域是正确的，并且能更好的与导体相贴的情况下才考虑使用使用它。

首先，如前所述，只有当最近的导体是非曼哈顿形体时才需要考虑旋转转移区域。

① 曼哈顿距离（定义3.2和5.4）意义下的最近，下同。

根据定义5.1, 5.2, 5.5和5.6，可以得到

定理 5.1：　对于任意非曼哈顿导体 X，$dist_h(p,X) \le dist_r(p,X) \le \sqrt{2}dist_h(p,X)$。

这意味着在水平方向上，旋转转移区域总是比曼哈顿转移区域大。因此只需要考虑旋转转移区域是否比纵向距离大，即只有当 $dist_r(p,X) > dist_v(p,X)$ 时才考虑旋转。

由于旋转转移区域仅仅是根据当前点与曼哈顿意义下的最近导体所确定的，因此还需要确定它是否不与其它导体相交。如图5.2所示，其中与右侧最近导体相贴的旋转转移区域与左侧导体相交，因此不能使用该旋转转移区域。为了判断旋转转移区域是否安全，即是否与其它导体相交，需要找到第二近的导体，以当前点 p 到第二近导体 Y 的曼哈顿距离 $dist(p,Y)$ 为半边长得到安全区域。仅当整个旋转的转移区域都在安全区域的内部时，才能旋转转移区域。即只有当

$$dist_r(p,X)(|\cos(\theta)| + |\sin(\theta)|) \le dist(p,Y) \tag{5-1}$$

成立时，才能旋转转移区域，其中 θ 是转移区域旋转的角度。

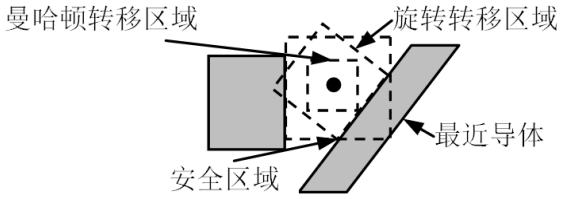


图 5.2　曼哈顿转移区域、旋转转移区域和安全区域

上述三个条件可以确保旋转的转移区域一定是安全的，并且比曼哈顿转移区域更好的与导体相贴。需要指出的是，其中部分条件并不总是必须的，但是准确的判断旋转的转移区域是否可以用过于复杂，因此仅在这三个条件都满足的情况下才使用旋转的转移区域。

5.3　适应非曼哈顿结构的空间管理

在每一步跳转时，仍然需要找到距离当前点最近的导体，并且为了使用旋转的转移区域，在某些情况下还需要找到第二近的导体。为此，需要对导体的空间管理结构进行扩展，使它能够即兼容非曼哈顿导体，又能提供查询第二近导体的功能。

首先考虑对每个空间单元节点的扩展，在构造候选导体列表的算法4中，需要计算空间单元 T 与新插入导体 A 的距离，以及 A 与其它候选导体 B 的遮挡关系。

这意味着需要能在导体 A 是非曼哈顿导体时计算 $dist(A,T)$ 和判断是否有 $A \leq_T B$ 或 $B \leq_T A$ 的算法。

定理 5.2：　对于使用中心点 c 和半边长 r 表示的立方体 $C=(c,r)$，以及导体 A，有：$dist(C,A)=dist(c,A)-r$。

根据上述定理，对于立方体的空间单元 T，利用定义5.4就可以计算 $dist(A,T)$。第3章已经讨论了立方体空间单元的优势以及如何实现，因此这里只考虑立方体空间单元。

遮挡关系的判断则要复杂一些。如果沿用定义3.4和3.5来构造候选导体列表，那么这样的候选导体列表将无法满足找到第二近导体的需要，因为第二近导体很可能被最近导体遮挡。因此需要放宽候选导体列表的要求。

注意到第二近导体不会被其它导体遮挡[①]，并且只在最近导体是非曼哈顿导体的情况下才可能需要第二近导体。因此，定义3.4只适用于导体 A 是曼哈顿导体的情形，当 A 是非曼哈顿导体时，遮挡关系需要重新定义。

当距离当前点 p 最近的导体 X 是非曼哈顿导体时，还需要第二近导体 Y 以构成安全区域。可以利用如下定理：

定理 5.3：　如果 $dist(p,Y) \geq 2dist(p,X)$，则旋转的转移区域一定在安全区域内。

证明　由定理5.1得到 $2dist(p,X) \geq \sqrt{2}dist_r(p,X)$，又公式 (5-1) 中 $|\cos(\theta)|+|\sin(\theta)|$ 的最大值为 $\sqrt{2}$，因此有 $dist(p,Y) \geq 2dist(p,X) \geq \sqrt{2}dist_r(p,X) \geq (|\cos(\theta)|+|\sin(\theta)|)dist_r(p,X)$。 □

这意味着，当 $dist(p,Y) \geq 2dist(p,X)$ 时，不需要知道 $dist(p,Y)$ 的具体值即不需要找到 Y 也可以安全的使用旋转的转移区域。因此对非曼哈顿导体的遮挡关系做出如下定义：

定义 5.7：　非曼哈顿导体 A 在空间单元 T 上遮挡 B：$A \leq_T B \Leftrightarrow \forall p \in T, 2dist(p,A) \leq dist(p,B)$。

根据如上定义，如果第二近导体 Y 被最近导体 X 遮挡，则一定有 $dist(p,Y) \geq 2dist(p,X)$，这时不需要找到 Y；如果 Y 没有被遮挡，则可以找到 Y 并计算出 $dist(p,Y)$。

与定义3.4类似，定义5.7也不能直接用于计算。算法3只适用于曼哈顿导体，因此也需要被扩展。有下述定理，

① 事实上可能被相同距离的导体遮挡，但这并不影响安全距离的计算。

定理 5.4：　对于导体 A 和 B 有：$\forall X \subseteq A, X \leq_T B \Rightarrow A \leq_T B$，$\forall X \supseteq A, B \leq_T X \Rightarrow B \leq_T A$。

证明　$X \leq_T B$ 意味着 $\forall p \in T, dist(p, X) \leq dist(p, B)$，又由 $X \subseteq A$ 可以得到 $dist(p, A) \leq dist(p, X)$，因此有 $dist(p, A) \leq dist(p, B)$，即 $A \leq_T B$。
$X \supseteq A$ 的情形同理可证。　□

算法 10 构造圆柱形 TSV 的内接和外接曼哈顿盒

输入：　圆柱形 TSV $A = (c, r, z_t, z_b)$

输出：　I_A 和 B_A

1: $s \leftarrow \frac{r}{\sqrt{2}}$
2: $I_A \leftarrow (x_c - s, y_c - s, z_b), (x_c + s, y_c + s, z_t))$
3: $B_A \leftarrow (x_c - r, y_c - r, z_b), (x_c + r, y_c + r, z_t))$
4: **return** I_A, B_A

算法 11 构造倾斜导体的内接和外接曼哈顿盒

输入：　倾斜导体 $A = (p_1, \ldots, p_n, p_{n+1} = p_1, z_t, z_b)$

输出：　I_A 和 B_A

1: $x_l \leftarrow \min_{i=1,n}(x_{p_i})), y_l \leftarrow \min_{i=1,n}(y_{p_i}))$
2: $x_r \leftarrow \max_{i=1,n}(x_{p_i})), y_r \leftarrow \max_{i=1,n}(y_{p_i}))$
3: $x_p \leftarrow \sum_{i=1,n}(x_{p_i})), y_p \leftarrow \sum_{i=1,n}(y_{p_i}))$
4: $p \leftarrow (\frac{x_p}{n}, \frac{y_p}{n}), r \leftarrow -dist_r(p, A)$
5: $I_A \leftarrow ((x_p - r, y_p - r, z_b), (x_p + r, y_p + r, z_t))$
6: $B_A \leftarrow ((x_l, y_l, z_b), (x_r, y_r, z_t))$
7: **return** I_A, B_A

对于非曼哈顿导体 A，可以构造两个曼哈顿形体 I_A 和 B_A 满足 $I_A \subseteq A, B_A \supseteq A$。算法10和11分别描述了针对圆柱形 TSV 和倾斜导体构造 I_A 和 B_A 的过程。为了完整性，对于曼哈顿导体 B，也可以定义 $I_B \leftarrow B_B \leftarrow B$。

定理5.4给出一种利用 I_A 和 B_B 判断遮挡关系的充分条件，它不是必要的。这意味着由此得到的候选导体列表中可能会有一些被其它候选导体遮挡但是没有检测出的导体。这在一定程度上会影响后续查询的效率，但是不影响算法的正确性。

注意定义3.4和5.7是有差别，这意味着算法3还需要相应的修改以适用于新的遮挡关系定义。算法12描述了修改后的算法。

算法 12 判断两个导体间的遮挡关系，可以包含非曼哈顿导体

输入： 导体 A，B，空间单元 $T = (l_T, r_T)$

输出： 是否有 $A \leq_T B$

1: $k \leftarrow 1$ **if** A 是曼哈顿导体 **else** 2
2: $A \leftarrow I_A, d_A \leftarrow dist(A, T)$
3: $B \leftarrow B_B, d_B \leftarrow dist(B, T)$
4: **if** $d_A > \frac{d_B}{k}$ **then**
5: **return false**
6: **end if**
7: **for** $w \in \{x, y, z\}$ **do**
8: **if** $w_{l_A} - \frac{d_B}{k} > \max(w_{l_B} - d_B, w_{l_T})$ **then**
9: **return false**
10: **else if** $w_{r_A} + \frac{d_B}{k} < \min(w_{r_B} + d_B, w_{r_T})$ **then**
11: **return false**
12: **end if**
13: **end for**
14: **return true**

在文献 [39] 中，作者将 TSV 的包围盒作为普通导体加入空间管理。并将每个 TSV 的邻近区域看作一个特殊的空间单元，为其建立候选导体列表。当查询点落在 TSV 邻近区域内时，在这个特殊空间单元内查找第二近导体。这种方式无需修改已有的空间管理算法，因而简单易行。但是当 TSV 数量很多时，大量的候选导体列表需要消耗较多的建立时间和存储。另一方面，这种方式要求导体包围盒不能相交，所以并不适用于包含倾斜导线的情形。因此本文在第3章介绍的空间管理的基础上做出上述改进，从而实现一种更一般的、可以将各种非曼哈顿形状导体统一处理的空间管理结构。

以上讨论了如何修改空间管理的建立过程以支持包含非曼哈顿导体的电路。接下来，将讨论修改查询阶段的算法。算法5给出了查找最近导体的过程，只要加入查找第二近导体的过程即可，算法13描述了得到曼哈顿转移区域半边长和安全区域半边长的过程。在此基础上，算法14描述了随机行走中建立转移区域的完整过程。

算法 13 计算最近导体，曼哈顿转移区域半边长和安全区域半边长

输入： 空间单元 T，T 内的点 p

输出： 最近导体 X，曼哈顿转移区域半边长 d_{min}，安全区域半边长 d_{safe}

1: $X \leftarrow null$
2: $d_{min} \leftarrow d_{safe} \leftarrow d_{nb} - dist(p,T)$
3: **for** $C \in list_T$ **do**
4: **if** $dist(C,T) \geq d_{safe}$ **then**
5: **break**
6: **end if**
7: $d \leftarrow dist(p,C)$
8: **if** $d < d_{min}$ **then**
9: $d_{safe} \leftarrow d$ **if** C 是曼哈顿导体 **else** d_{min}
10: $X \leftarrow C, d_{min} \leftarrow d$
11: **else if** $d < d_{safe}$ **then**
12: $d_{safe} \leftarrow d$
13: **end if**
14: **end for**
15: **return** X, d_{min}, d_{safe}

5.4　针对非曼哈顿结构的高斯面的建立和采样

以上讨论了当电路中包含非曼哈顿结构的导体时进行随机行走的过程。然而，如果需要提取电容的主导体包含非曼哈顿导体，还需要为它建立高斯面，并在高斯面上进行采样以作为随机行走的起点。

注意到，如果利用第4章提出虚拟高斯面采样技术，只需要考虑针对一个非曼哈顿导体块建立高斯面并进行采样即可。拒绝采样过程会自然的完成对全线网高斯面的采样。

5.4.1　圆柱形 TSV 的高斯面的建立和采样

圆柱形 TSV 高斯面的建立非常简单。考虑到每个 TSV 都有一个排除区域，在这个范围内不会有其它导体。这意味着，如果 TSV 的曼哈顿包围盒与某个导体相接触，则该导体与 TSV 应属于同一个线网，在建立导体块的高斯面时会被忽略。因此可以使用算法8直接为 TSV 的包围盒建立高斯面。

高斯面作为随机采样的起点，其位置对算法的效率有显著的影响。在第4章中，$factor_G$ 被设定为一个略大于 1 的值。然而，在这里这并不是一个很好的选择。考

算法 14 以当前点为中心建立转移区域

输入： 当前点 p，空间管理结构 S

输出： 转移区域类型和半边长 r

1: 在 S 中定位包含点 p 的空间单元 T。
2: 利用算法13在 T 中找到最近的导体 X，以及 d_{min} 和 d_{safe}
3: **if** $d_{min} = 0$ **then**
4: 　　**return** 终止于导体 X
5: **else if** $X = null \vee X$ 是曼哈顿导体 **then**
6: 　　**return** 曼哈顿转移区域，半边长 d_{min}
7: **end if**
8: $r \leftarrow dist_r(p, X)$
9: 根据定义5.5和5.6计算旋转角度 θ
10: **if** $r < d_{min} \vee r(|\cos(\theta)| + |\sin(\theta)|) > d_{safe}$ **then**
11: 　　**return** 曼哈顿转移区域，半边长 d_{min}
12: **else**
13: 　　**return** 旋转的转移区域，半边长 r
14: **end if**

虑到 TSV 在不同方向上与相邻导体的距离差别很大。例如，在水平方向上 TSV 与侧面的相邻导体的距离是 10μm，而上下两端到相邻导体的距离只有 0.3μm。如果 $factor_G$ 很接近于 1，则得到的高斯面将距离侧面相邻的导体非常远，如图5.3所示。

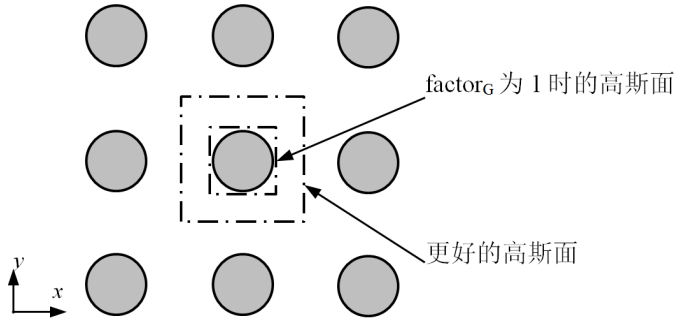


图 5.3　圆柱形 TSV 结构的高斯面俯视图

这样使得随机行走第一步的转移区域远小于它可以的大小。第一个转移区域的大小 $L(r)$ 又影响采样的权值，从式 (2-25) 可以看出，较小的 $L(r)$ 会带来较大的权值变化，从而不利于算法的效率。因此，对于圆柱形 TSV 结构，将 $factor_G$ 设定为足够大的值，即在不同方向上，允许高斯面与主导体块之前的距离有较大的差异，从而得到更好的高斯面（图5.3）。

在算法8中，6 个方向上计算主导体到其它导体的距离 d_s 的方法都是相同的。这意味着如图5.4所示，导体 B 被看作是 $+x$ 方向的导体，并将限制 d_{+x}，因为它位于 $+x$ 和 $+z$ 方向的 45° 分割线的下方。为了得到图中期望的高斯面，将 $+x$ 和 $+z$ 方向的分割线修改为水平方向。

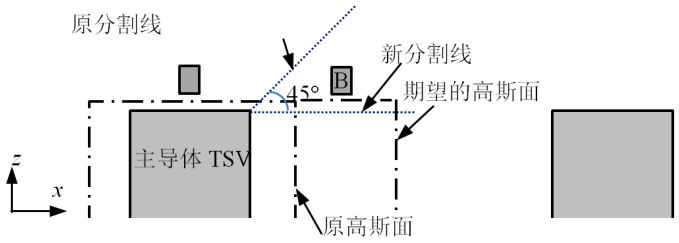


图 5.4　圆柱形 TSV 结构的高斯面侧视图

经过上述修改，得到了更适合圆柱形 TSV 几何结构的高斯面。它的第一步转移区域尽可能较大，从而减少达到指定精度所需的采样数。然而，这时的高斯面在不同方向上与主导体的距离差异很大，即第一个转移立方体的大小差异很大。这同样会使得权值（式 (2-25)）变化较大。通常，起始于圆柱形 TSV 高斯面上下表面的采样 r 的 $L(r)$ 较小而权值较大。因此利用重要性采样技术可以减小这种不利影响。

高斯面上的采样概率函数有 4 种基本的选择，其中式 (4-13) 和 (4-14) 因为包含有 $D(r)$ 项从而可以减小 $L(r)$ 的影响。由于 $f(r) \propto \frac{1}{D(r)}$，因此，高斯面的上下表面的采样点将会较多。从物理意义的角度来看，由于这里是在计算电荷，即法向电场强度的积分，理想的重要性采样应该是近似于法向电场强度的。而 $f(r) \propto \frac{1}{D(r)}$ 正好体现了电场强度反比于距离的特性，因此可以有效的改善随机采样的收敛速度。

5.4.2　倾斜导线的高斯面的建立和采样

倾斜导线的高斯面的建立则更为复杂。如图5.5(a) 所示，由于没有排除区域的保护，倾斜导线的包围盒是可能与其它导体相接触的，因此在建立高斯面时必须考虑倾斜导体本身的形状。此时得到的高斯面也很可能不再是曼哈顿形状的（图5.5(b)），因此如何在上面进行采样也需要考虑[①]。

与一般的高斯面生成算法类似，生成倾斜导线的高斯面的过程中，也需要计算主导体块到其它导体的距离。在垂直方向上，考虑到倾斜导体是棱柱结构，因此可以沿用定义4.1中的 $dist_{+z}$ 和 $dist_{-z}$，并以此设定高斯面的上下表面位置。接下来只需要考虑在水平方向上如何建立多边形导体的高斯面。

① 只需实现均匀采样即可，拒绝采样技术会完成任意概率密度的采样。

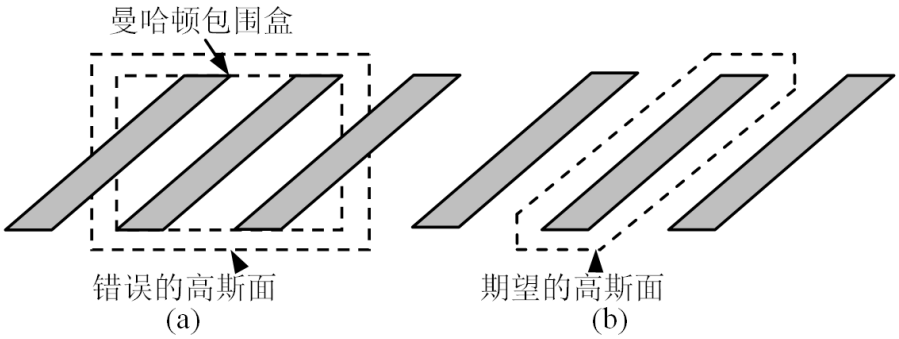


图 5.5　倾斜导体和高斯面的俯视图

对于两个不相互接触的多边形，可以利用如下定义计算距离:

定义 5.8：　多边形 $P = (p_1, \ldots, p_n)$ 和 $Q = (q_1, \ldots, q_m)$ 的水平距离：$dist_h(P, Q) \leftarrow \min(\min_{i=1,n}(dist_h(p_i, Q)), \min_{j=1,m}(dist_h(q_i, P)))$。

在生成高斯面的过程中，圆柱形 TSV 被其曼哈顿包围盒替代，同时曼哈顿形体的横截面长方形是多边形的一种特例。因此任何导体间的水平距离都可以由定义5.8来计算[①] 。

在得到多边形主导体到其他导体的最小距离 $2d_h$ 之后，可以在主导体周围描绘高斯面。如图5.5(b) 所示，对于多边形主导体 A，其高斯面 G_A 应尽量满足 $\forall p \in G_A, dist_h(p, A) = d_h$。可以看出，对于 n 边形主导体 A，G_A 也是多边形并且其边数介于 n 和 $2n$ 之间，算法15描述了生成 G_A 的过程，其基本思路是将 A 的每个边向外扩张 d_h。

倾斜导线完整的高斯面也是棱柱结构，这意味着虽然其顶面和底面是不规则的凸多边形，但其每个侧面都是长方形，因此只需要关心如何在凸多边形上进行均匀采样。先考虑最简单的多边形，三角形。用三个点来表示三角形 A，即 $A = (a, b, c)$，则 A 内的任何一点 u 可以由两个 $[0, 1]$ 区间内的数 λ 和 μ 表示，即 $u(\lambda, \mu) = c + \mu(a + \lambda(b - a) - c)$。

定理 5.5：　如果 $\lambda \leftarrow random()$ 并且 $\mu \leftarrow \sqrt{random()}$，其中 $random()$ 每次返回一个 $[0, 1]$ 区间内均匀分布的随机数 [36]，则 $u(\lambda, \mu)$ 在三角形 $A = (a, b, c)$ 均匀分布。

证明　当 $\lambda \leftarrow random()$ 时，易得 $v(\lambda) = a + (b - a)\lambda$ 在线段 $[a, b]$ 上均匀分布。
当 $\mu \leftarrow \sqrt{random()}$ 时，$p(\mu \in [0, x]) = p(random() \in [0, x^2]) = x^2$，即 $p(\mu = x) = 2x$。μ 在 $[0, 1]$ 内线性分布意味着 $u(\lambda, \mu) = c + \mu(v(\lambda) - c)$ 在线段 $[c, v(\lambda)]$ 上线性分布，即 $u(\lambda, \mu)$ 在三角形内均匀分布。　□

① 定义5.8与3.3计算方法不同，但对于不相交的两个长方形，其结果是相同的。

算法 15 生成多边形导体的高斯面

输入： 多边形主导体 $A = (p_1, \ldots, p_n)$，其它导体 Bs

输出： 高斯面 G_A

$d_h \leftarrow \min_{B \in Bs}(dist_h(A, B))/2$
for $i = 1, n$ **do**
    $u = (sign(y_{p_{i+1}} - y_{p_i}), sign(x_{p_i} - x_{p_{i+1}}))$　　▷ *sign* 来自于公式 (2-21)
    if $x_u = 0 \vee y_u = 0$ **then**
        $a_{2i} \leftarrow (y_u + x_u, y_u - x_u), a_{2i+1} \leftarrow (x_u - y_u, x_u + y_u)$
    else
        $a_{2i} \leftarrow a_{2i+1} \leftarrow u$
    end if
end for
$a_1 \leftarrow a_{2n+1}$
$m = 0$
for $i = 1, n$ **do**
    $u \leftarrow a_{2i-1}, v \leftarrow a_{2i}$
    $m \leftarrow m + 1, q_m \leftarrow (x_{p_i} + x_u * d_h, y_{p_i} + y_u * d_h)$
    if $u = v$ **then**
        continue
    else if $u + v = (0, 0)$ **then**
        $m \leftarrow m + 1, q_m \leftarrow (x_{p_i} - y_u * d_h, y_{p_i} + x_u * d_h)$
    end if
    $m \leftarrow m + 1, q_m \leftarrow (x_{p_i} + x_v * d_h, y_{p_i} + y_v * d_h)$
end for
$G_A = (q_1, \ldots, q_m)$
return G_A

对于一般的凸 n 边形 $P = (p_1, \ldots, p_n)$，连接 $(p_1, p_3), (p_1, p_4), \ldots, (p_1, p_{n-1})$，可以将其分割成 $n - 2$ 个三角形。因此在多边形上均匀采样可以先根据面积选择三角形，再在三角形内根据定理5.5选择均匀分布的点。

5.5　实验结果和分析

本文在 RWCap2 的基础上实现了本章中提出的各种算法。首先通过与 Raphael、FastCap[3] 和 QBEM[5] 横向对比来验证本章提出的算法的正确性与精度，然后通

过纵向对比验证旋转转移区域和空间管理技术等带来加速。

在这一节使用到的例子有：

- 测例 6：如图5.6所示，包含 9 个 TSV。TSV 的直径和间距均为 5μm，高为 70μm。普通导线宽为 0.2μm，高为 0.36μm。
- 测例 7：在测例 6 的基础上去掉了 TSV 之间的导线，只留上下两层导线。
- 测例 8：在测例 7 的基础上，进一步去掉了 1、3、7 和 9 号 TSV（图5.6）和部分导线。
- 测例 9：随机布局的 400 个 TSV 和平行导线。每个 TSV 直径为 4μm，高为 40μm，TSV 间距不小于 4μm。导线几何尺寸与测例 6 相同。

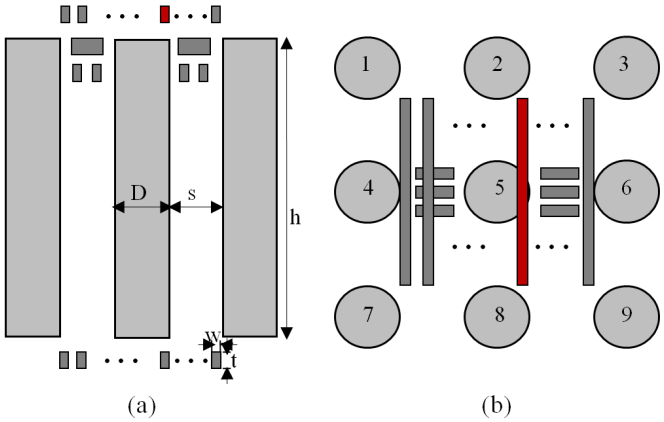


图 5.6　测例 6 示意图，侧视图 (a)，俯视图 (b)

- 测例 10：包含倾斜导线的例子。包含 492 个线网，521 个导体块，其中 310 个为非曼哈顿导体块。导线最小线宽为 25nm，外形尺为 38μm×114μm×32μm。
- 测例 11、12，13：与测例 10 类似。图5.7展示了每个例子中包含主导体的金属层。

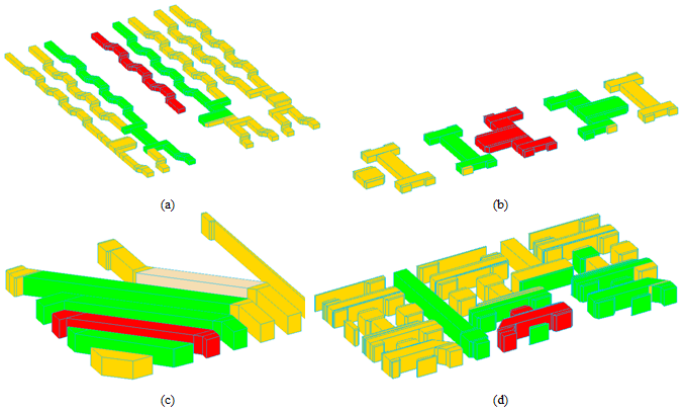


图 5.7　测例 10(a)，11(b)，12(c)，13(d) 中包含主导体（红色）的金属层

5.5.1　验证算法准确性

对于包含 TSV 的例子，将分别提取中心 TSV 的主电容和一个顶层导线（如图5.6所示红色导线）到中心 TSV 的耦合电容 [40]。使用随机行走提取电容时，设定误差限为主导体总电容 0.5%，耦合电容 1%。这里没有使用测例 9，因为它对于基于线性方程组的解法而言太大了。

首先验证使用正方形横截面的曼哈顿导体近似替代圆柱形 TSV 所带来的误差。为了尽量减小误差，让正方形横截面与原圆形横截面的面积相等，即正方形边长 $a = \sqrt{\pi}r$，其中 r 为圆柱的半径。实验中还尝试了其它的近似方式，如近似正方形与原圆形周长相等即 $a = \frac{\pi}{2}r$，使用圆的外接正方形即 $a = 2r$ 或内接正方形即 $a = \frac{r}{\sqrt{2}}$。实验表明使用等面积正方形进行替代时误差最小，使用外接或内接正方形时误差最大（20% 以上）。

表5.2展示了分别使用 Raphael，RWCap2 和 RWCap3（在 RWCap2 的基础上了实现本章提出的算法）提取中心 TSV 的总电容的结果。表5.2展示了提取耦合电容的结果。Raphael 的结果包含精确描述的圆柱形结构和近似的正方形结构两部分，RWCap2 只能得到正方形近似的结果，RWCap3 可以对圆柱形结构进行提取。

表 5.1　使用 Raphael，RWCap2 和 RWCap3 提取中心 TSV 的总电容

测	Raphael			RWCap2		RWCap3		
例	圆柱形 (aF)	正方形 (aF)	误差 *	电容 (aF)	时间 (s)	电容 (aF)	误差 +	时间 (s)
6	3866	4065	5.1%	4056	2.01	3890	0.6%	2.39
7	3740	3962	5.9%	3930	2.06	3794	1.4%	1.88
8	3718	3939	5.9%	3916	2.58	3747	0.8%	2.41

* 正方形近似替代相对与圆柱形描述的误差。

+ RWCap3 相对于 Raphael 对圆柱形描述的误差。

表 5.2　使用 Raphael，RWCap2 和 RWCap3 提取顶层导体到中心 TSV 的耦合电容

测	Raphael			RWCap2		RWCap3		
例	圆柱形 (aF)	正方形 (aF)	误差 *	电容 (aF)	时间 (s)	电容 (aF)	误差 +	时间 (s)
6	48.2	58.6	21.6	58.2	4.2	48.3	-0.2	4.38
7	49.9	60.2	20.6	59.6	3.5	50.0	0.2	4.74
8	50.0	60.4	20.8	60.0	3.9	49.9	-0.2	5.18

* 正方形近似替代相对与圆柱形描述的误差。

+ RWCap3 相对于 Raphael 对圆柱形描述的误差。

从表5.1和5.2可以看出，使用正方形近似替代引入的主导体总电容误差在

5%~6% 之间，耦合电容误差更大，可以达到 20%。这充分说明了在电容提取过程中，精确地描述圆柱形 TSV 是很有必要的。算法 RWCap3 得到的结果与 Raphael 的圆柱形结果间的误差小于 1.5%，其中耦合电容误差更小。至于运行时间，RWCap3 相比与 RWCap2 所需的时间只增加了约 30%。为了 5% 甚至更多的精度而牺牲 30% 的时间是值得的。

表5.3和5.4分别展示了使用 FastCap 和 QBEM 提取中心 TSV 主导体总电容和顶层导线耦合电容的结果。从中可以看出，FastCap 在计算主导体总电容时精度很好，但是计算耦合电容的精度很差。注意到在这些例子里，TSV 的几何尺寸都非常大并且还有大量的平行线，因此 FastCap 的离散化可能不够细致。但是如果增加离散密度，FastCap 会因为内存限制而无法运行。另一方面，QBEM 能够较精确地得到主导体总电容和耦合电容，但是计算时间和存储的使用量都非常大。

表 5.3　使用 FastCap，QBEM 和 RWCap3 提取中心 TSV 的总电容

测	FastCap				QBEM				RWCap3			
例	电容 (aF)	误差 (%)*	时间 (s)	存储 (GB)	电容 (aF)	误差 (%)*	时间 (s)	存储 (GB)	时间 (s)	加速 1+	加速 2+	存储 (MB)
6	3736	-3.4	79.0	1.9	3708	-4.1	404	7.7	2.39	33	169	~1
7	3710	-0.8	67.3	1.8	3603	-3.7	402	7.6	1.88	36	214	~1
8	3691	-0.7	50.1	1.1	3547	-4.6	271	5.3	2.41	21	112	~1

* 相对于 Raphael 的误差。

+ 加速 1 和加速 2 分别为 RWCap3 相对于 FastCap 和 QBEM 的加速。

表 5.4　使用 FastCap，QBEM 和 RWCap3 提取顶层导体到中心 TSV 的耦合电容

测	FastCap				QBEM				RWCap3			
例	电容 (aF)	误差 (%)*	时间 (s)	存储 (GB)	电容 (aF)	误差 (%)*	时间 (s)	存储 (GB)	时间 (s)	加速 1+	加速 2+	存储 (MB)
6	64.5	33.8	79.1	1.9	46.1	-4.4	299	6.0	4.38	18	68	~1
7	64.9	30.1	66.8	1.8	48.0	-3.8	298	5.9	4.74	14	63	~1
8	65.1	30.3	51.2	1.1	47.9	-4.2	192	4.2	5.18	10	37	~1

* 相对于 Raphael 的误差。

+ 加速 1 和加速 2 分别为 RWCap3 相对于 FastCap 和 QBEM 的加速。

运行时间方面，RWCap3 比快速 BEM 算法快 10 倍以上，甚至可以达到 214 倍，并且能够精确的提取主导体总电容和耦合电容。一般的，随机行走算法相对于 BEM 算法的加速会随着问题规模的增大进一步增加。

对于包含倾斜导体的例子，由于主导体是一个由很多不同形状的导体块组成

的线网，因此只提取主导体的总电容。表5.5展示了 RWCap3 与 QBEM 算法的对比。可以看到，对于复杂的包含倾斜导体的线网，RWCap3 能够精确的提取线网电容，误差被控制在 1.5% 以内。同时，与 QBEM 相比可以获得 4~7 倍加速。对于更大的例子，加速效果将会更好。

表 5.5　使用 QBEM 和 RWCap3 和提取包含倾斜导体的线网的总电容

测例	QBEM		RWCap3			
	电容 (aF)	时间 (s)	电容 (aF)	误差	时间 (s)	加速比
10	451.0	12.83	444.5	-1.4%	1.69	7.6
11	282.3	5.78	285.5	1.1%	0.86	6.7
12	242.9	11.68	246.5	1.5%	3.07	3.8
13	402.3	20.15	405.8	0.9%	4.59	4.4

5.5.2　验证算法效率

本章在 RWCap2 的基础上，提出了一系列的扩展，以便将随机行走算法适用到包含非曼哈顿形状导体的结构中。具体的改进有：扩展点到导体的距离定义，可以生成与非曼哈顿导体接触的立方体转移区域（M_1）；旋转立方体转移区域使得它能更好的与非曼哈顿导体接触（M_2）；适用于非曼哈顿导体的空间管理（M_3）；适合于圆柱形 TSV 的高斯面的建立（M_4）和重要性采样（M_5）。其中 M_4 和 M_5 只针对于主导体是圆柱形 TSV 的情形。表5.6和5.7纵向的对比了每个扩展所带来的性能提升。

表 5.6　算法 M_1，M_2 和 M_3 提取非曼哈顿结构主导体总电容的性能对比

测	M_1 曼哈顿转移区域			M_2 旋转转移区域				M_3 使用空间管理			
例	采样数 (K)	跳转次数	时间 (s)	采样数 (K)	跳转次数	时间 (s)	加速 *	采样数 (K)	跳转次数	时间 (s)	加速 *
6	2182	37.4	51.48	2186	10.8	17.94	2.9	2216	11.3	10.17	1.8
7	2293	37.6	45.43	2261	10.7	15.37	3.0	2289	11.3	9.93	1.5
8	2290	37.5	37.09	2309	10.8	13.64	2.7	2323	13.8	11.70	1.2
9	5656	36.0	1906	5605	10.4	584.2	3.3	5642	10.9	23.60	24.8
10	176	11.5	30.36	171	9.7	24.11	1.3	173	9.7	1.69	14.3
11	173	10.2	2.66	170	10.0	2.57	1.0	174	10.9	0.86	3.0
12	280	15.9	33.52	280	12.4	26.16	1.3	274	13.1	3.07	8.5
13	384	28.9	46.93	376	13.0	23.90	2.0	384	14.9	4.59	5.2

* 相对于前一组时间的加速。

表 5.7　算法 M_4 和 M_5 提取圆柱形 TSV 主导体总电容的性能对比

测	M_4TSV 高斯面位置优化				M_5 高斯面重要性采样			
例	采样数 (K)	跳转次数	时间 (s)	加速 *	采样数 (K)	跳转次数	时间 (s)	加速 *
6	1560	13.9	8.63	1.2	440	11.6	2.39	3.6
7	957	15.5	5.48	1.8	277	12.9	1.88	2.9
8	976	26.7	8.19	1.4	282	20.3	2.41	3.4
9	1716	15.1	9.62	2.5	389	12.1	3.00	3.2

* 相对于前一组时间的加速。

从表5.6中 M_2 与 M_1 的对比可以看出，旋转转移区域能够大幅度的减少平均每次采样所需的跳转次数。对于以 TSV 为主导体的例子，它可以带来 2~3 倍的加速。对于包含倾斜导体的例子，加速比相对较少，这主要是因为倾斜的面占所有面的比例较低。从 M_3 与 M_2 的对比中可以看出，空间管理能够带来较大的加速，并且例子中导体越多加速效果约明显。如测例 13、10 和 9 分别包含 383、521 和 1816 个导体块，空间管理为它们带来的加速分别为 5.2、14.3 和 24.8 倍。

从表5.7可以看出，针对 TSV 结构的高斯面和在高斯面上的重要性采样等减小方差的技术能够显著的减少采样次数。后者还能减少平均每次采样的跳转次数，这是因为使用重要性采样后，高斯面上距离导体较近的区域采样概率增加，而从这些区域出发的采样点相对会在比较少的跳转次数后到终止于导体表面。使用针对 TSV 的高斯面和重要性采样技术能够带来 5~8 倍的加速。

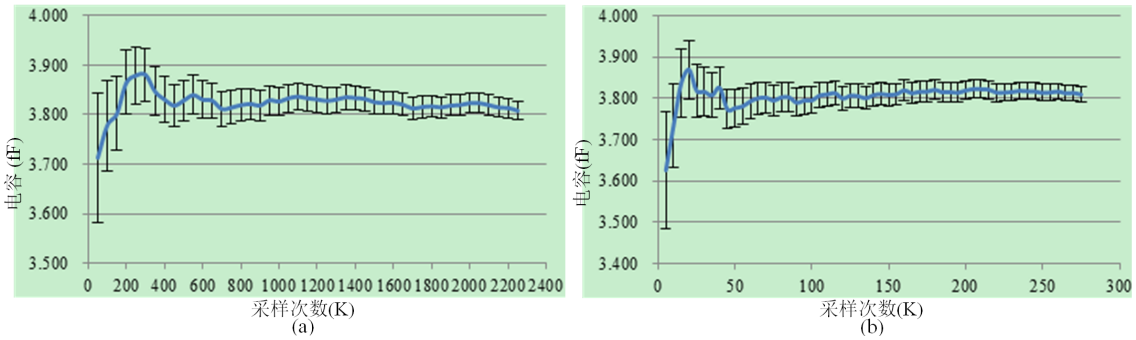
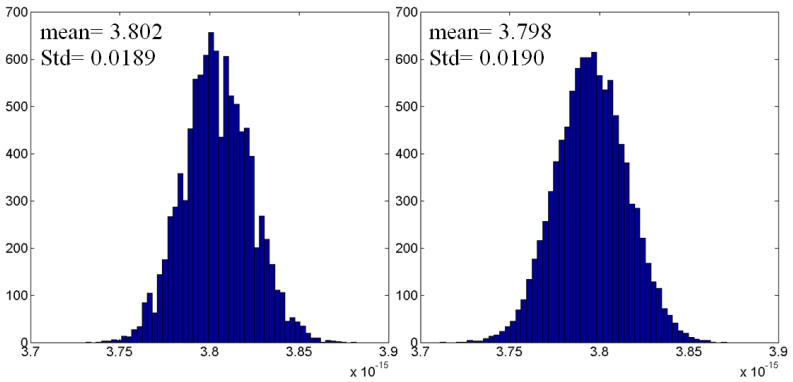


图 5.8　使用 M_3(a) 和 M_5(b) 提取测例 7 中心 TSV 总电容的收敛情况

旋转转移区域和空间管理分别在减少公式 (2-14) 中的 N_{hop} 和 T_{hop} 两项，它们对于电容的计算结果没有直接的影响，因此表5.6也没有对比电容值。高斯面的位置和重要性采样技术主要减小 N_{walk}。以测例 7 为例比较 M_3 和 M_5 的收敛情况，从图5.8可以明显的看出，优化的高斯面和采样概率能够显著的加快收敛速度。

为了验证优化高斯面和采样概率不会影响结果的准确性，分别用 M_3 和 M_5 提

取测例 7 中心 TSV 总电容 10000 次，并将结果的分布情况画在图5.9中。可以看到，两个图都基本符合正态分布，并且计算出的标准差（小于均值的 0.5%）表明本文提出的方差减小技术不会影响结果的准确性。



(a)　(b)

图 5.9　使用 M_3(a) 和 M_5(b) 提取测例 7 中心 TSV 总电容 10000 次的结果分布

5.6　本章小结

本章针对以圆柱形 TSV 和倾斜导线为代表的非曼哈顿导体提出了旋转的转移区域，改进的空间管理结构和高斯面生成等技术。这些技术使得悬浮随机行走算法可以直接应用于准确描述的几何形体从而避免近似替代所带来的误差。实验表明，使用本文提出的技术提取圆柱形 TSV 和倾斜导线的电容误差小于 1.5%。同时，各项改进也显著的加速了非曼哈顿导体电容的提取。具体的，旋转的转移区域能加速 3 倍，空间管理技术能加速近 25 倍，高斯面和采样概率的优化则能加速近 8 倍。

第 6 章　总结与展望

本文提出的技术从三个方面使悬浮随机行走电容提取算法得到了性能提升和功能加强。首先研究了针对大规模结构的空间管理技术。改进了空间管理结构的建立和查询过程，所提出的技术能将八叉树结构空间管理的建立加速数千倍，并能将随机行走过程加速 2 倍。另外还对比了多种不同的索引组织形式，并以此为基础提出了网格-八叉树混合结构，获得存储开销和计算性能更好的平衡。实验结果表明，对于包含近 50 万块导体的 VLSI 结构，混合空间管理的建立时间少于 20 秒，而随机行走过程也快于工业界最先进的技术。

本文还研究了全线网电容提取问题，提出了虚拟高斯面采样技术。它不需要复杂的几何计算即可生成由多个导体块和通孔组成的完整线网的高斯面，并且能够与重要性采样技术配合。在研究高斯面的位置对随机行走过程收敛速度的影响的基础上，本文提出了优化的高斯面位置和高斯面上采样概率密度函数，从而减少达到指定精度所需的采样数。实验表面，优化的高斯面位置和采样概率密度函数能够带来 1.5 倍加速。

最后，本文将现有的随机行走算法扩展到能处理包含非曼哈顿形体的结构。为此设计了旋转的转移区域以便与非曼哈顿形体更好的接触，扩展了空间管理结构，使它能够很好的索引非曼哈顿形体。同时还研究了非曼哈顿形体的高斯面的构造技术，并针对 TSV 结构进一步优化了高斯面的位置和采样概率。实验表明，本文提出的算法能够精确的提取非曼哈顿导体的电容（误差小于 1.5%），并且相比于快速 BEM 算法加速数百倍。

本文中的实验虽然只针对了单介质的情形，但是事实上提出的各项改进算法都是与介质无关的，它们可以自然地应用于多介质的环境中，详细结果可以参考[32,35,39]。

在本文工作的基础上，仍然有许多值得进一步研究和改进的方面。针对非曼哈顿结构的几何计算方面，有许多检测算法（如遮挡关系的判断和旋转的转移区域的使用等）会有漏判的情形，它们对整个随机行走过程的性能有着不利的影响。接下来的工作可以考虑如何使用高召回率的算法替代它们。

此外，现有的随机行走算法还有一些不足，例如在针对触屏电路的电容提取问题中，快速准确地提取耦合电容往往也是非常重要的，而目前的算法在提取耦合电容时收敛较慢，效率偏低。另一方面，现有的算法也不能有效地处理很多实际工艺中包含的悬浮哑元、保形介质等结构。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。

本文提出的算法提升了悬浮随机行走电容提取算法在面对大规模结构时的性能，并且使得该算法能够更好更准确的适应更为复杂的电路结构。这大大的拓宽了 FRW 算法的应用场景，为其进一步地发展和应用打下了坚实的基础。

参考文献

[1] Meijs N, Genderen A J. An efficient finite element method for submicron ic capacitance extraction. Proceedings of the 26th ACM/IEEE Design Automation Conference. ACM, 1989. 678–681.

[2] Chen G, Zhu H, Cui T, et al. Parafemcap: a parallel adaptive finite-element method for 3-d vlsi interconnect capacitance extraction. Microwave Theory and Techniques, IEEE Transactions on, 2012, 60(2):218–231.

[3] Nabors K, White J. Fastcap: A multipole accelerated 3-d capacitance extraction program. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 1991, 10(11):1447–1459.

[4] Shi W, Liu J, Kakani N, et al. A fast hierarchical algorithm for three-dimensional capacitance extraction. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2002, 21(3):330–336.

[5] Yu W, Wang Z. Enhanced qmm-bem solver for three-dimensional multiple-dielectric capacitance extraction within the finite domain. Microwave Theory and Techniques, IEEE Transactions on, 2004, 52(2):560–566.

[6] Yan S, Sarin V, Shi W. Sparse transformations and preconditioners for 3-d capacitance extraction. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2005, 24(9):1420–1426.

[7] Yu W, Zhang M, Wang Z. Efficient 3-d extraction of interconnect capacitance considering floating metal fills with boundary element method. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2006, 25(1):12–18.

[8] Chai W, Jiao D, Koh C K. A direct integral-equation solver of linear complexity for large-scale 3d capacitance and impedance extraction. Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE. IEEE, 2009. 752–757.

[9] Le Coz Y, Iverson R. A stochastic algorithm for high speed capacitance extraction in integrated circuits. Solid-State Electronics, 1992, 35(7):1005–1012.

[10] Le Coz Y, Greub H, Iverson R. Performance of random-walk capacitance extractors for ic interconnects: a numerical study. Solid-State Electronics, 1998, 42(4):581–588.

[11] Iverson R B, Le Coz Y L. A floating random-walk algorithm for extracting electrical capacitance. Mathematics and computers in simulation, 2001, 55(1):59–66.

[12] Brambilla A, Maffezzoni P. A statistical algorithm for 3-d capacitance extraction. IEEE microwave and guided wave letters, 2000, 10(8):304–306.

[13] Batterywala S H, Desai M P. Variance reduction in monte carlo capacitance extraction. VLSI Design, 2005. 18th International Conference on. IEEE, 2005. 85–90.

[14] Batterywala S, Ananthakrishna R, Luo Y, 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. 6–pp.

[15] Kamon M, Iverson R. High-accuracy parasitic extraction. EDA for IC implementation, circuit design, and process technology. CRC Press/Taylor & Francis Group, Boca Raton, 2006..

[16] El-Moselhy T A, Elfadel I M, Daniel L. A hierarchical floating random walk algorithm for fabric-aware 3d capacitance extraction. Proceedings of the 2009 International Conference on Computer-Aided Design. ACM, 2009. 752–758.

[17] Zhuang H, Yu W, Hu G, et al. Fast floating random walk algorithm formulti-dielectric capacitance extraction with numerical characterization of green's functions. Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific. IEEE, 2012. 377–382.

[18] Yu W, Zhuang H, Zhang C, et al. Rwcap: A floating random walk solver for 3-d capacitance extraction of very-large-scale integration interconnects. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2013, 32(3):353–366.

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

[20] Yu W, Zhai K, Zhuang H, et al. Accelerated floating random walk algorithm for the electrostatic computation with 3-d rectilinear-shaped conductors. Simulation Modelling Practice and Theory, 2013, 34:20–36.

[21] Liu C, Song T, Cho J, et al. Full-chip tsv-to-tsv coupling analysis and optimization in 3d ic. Proceedings of the 48th Design Automation Conference. ACM, 2011. 783–788.

[22] Savidis I, Friedman E G. Closed-form expressions of 3-d via resistance, inductance, and capacitance. Electron Devices, IEEE Transactions on, 2009, 56(9):1873–1881.

[23] Katti G, Stucchi M, De Meyer K, et al. Electrical modeling and characterization of through silicon via for three-dimensional ics. Electron Devices, IEEE Transactions on, 2010, 57(1):256–262.

[24] Kim D H, Mukhopadhyay S, Lim S K. Fast and accurate analytical modeling of through-silicon-via capacitive coupling. Components, Packaging and Manufacturing Technology, IEEE Transactions on, 2011, 1(2):168–180.

[25] Peng Y, Song T, Petranovic D, et al. On accurate full-chip extraction and optimization of tsv-to-tsv coupling elements in 3d ics. Proceedings of the International Conference on Computer-Aided Design. IEEE Press, 2013. 281–288.

[26] Takagi M, Yamaguchi K, Chida H, et al. Layout and reticle verification for fpd. Photomask and NGL Mask Technology XIX. International Society for Optics and Photonics, 2012. 84410M–84410M.

[27] Uchida Y, Tani S, Hashimoto M, et al. Interconnect capacitance extraction for system lcd circuits. Proceedings of the 15th ACM Great Lakes symposium on VLSI. ACM, 2005. 160–163.

[28] Vetterling W T, Teukolsky S A, Press W H. Numerical recipes: example book (C). Press Syndicate of the University of Cambridge, 1992.

[29] Bentley J L. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975, 18(9):509–517.

[30] Bansal N. Randomized algorithms for capacitance estimation[D]. Citeseer, 1999.

[31] Samet H. Applications of spatial data structures. 1990..

[32] Zhang C, Yu W. Efficient techniques for the capacitance extraction of chip-scale vlsi interconnects using floating random walk algorithm. ASP-DAC, 2014. 756–761.

[33] Deschacht D, De Rivaz S, Farcy A, et al. Keep on shrinking interconnect size: is it still the best solution? Electronic Manufacturing Technology Symposium (IEMT), 2010 34th IEEE/CPMT International. IEEE, 2010. 1–4.

[34] G Rollins, Online presentation of Synopsys, Inc. Rapid3d 20x performance improvement[EB/OL]. [2010]. http://www.synopsys.com/Community/UniversityProgram/Pages/Presentations.aspx.

[35] Zhang C, Yu W. Efficient space management techniques for large-scale interconnect capacitance extraction with floating random walks. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2013, 32(10):1633–1637.

[36] Matsumoto M, Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation (TOMACS), 1998, 8(1):3–30.

[37] Hwang C O, Given J A, Mascagni M. The simulation–tabulation method for classical diffusion monte carlo. Journal of Computational Physics, 2001, 174(2):925–946.

[38] Mascagni M, Hwang C O. ϵ-shell error analysis for "walk on spheres" algorithms. Mathematics and computers in simulation, 2003, 63(2):93–104.

[39] Yu W, Zhang C, Wang Q, et al. Random walk based capacitance extraction for 3d ics with cylindrical inter-tier-vias. Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design. IEEE Press, 2014. 702–709.

[40] Peng Y, Petranovic D, Lim S K. Fast and accurate full-chip extraction and optimization of tsv-to-wire coupling. Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference. ACM, 2014. 1–6.

致　谢

首先，我要衷心地感谢我的导师喻文健老师。在我从本科毕设到硕士毕业的整个过程中，喻老师一直对我的工作给予着鼓励和支持，帮助我克服了期间遇到的各种困难。喻老师严谨的治学态度、敏锐的学术眼光、勤奋的工作态度和深厚的学术功底也是我学习的榜样，并将积极影响我今后的学习和工作。

此外，我还要感谢大学到硕士这七年间所有的老师，身边的同学，还有我的家人。在我的学习和成长过程中，他们所一直给予我的鼓励与支持是我不断前进的动力。

最后，我要感谢清华大学计算机系七年来对我的培养，是清华让我成为了今天的我。

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

1989 年 12 月 1 日出生于青海省西宁市。

2008 年 9 月考入清华大学计算机科学与技术系计算机科学与技术专业，2012 年 7 月本科毕业并获得工学学士学位。

2012 年 9 月免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] **Chao Zhang**, Wenjian Yu, Efficient space management techniques for large-scale interconnect capacitance extraction with floating random walks, *IEEE Trans. Computer-Aided Design*, 32(10): 1633-1637, 2013 (SCI 收录，影响因子 1.5)

[2] Wenjian Yu, **Chao Zhang**, Qing Wang, and Yiyu Shi, Random walk based capacitance extraction for 3D ICs with cylindrical inter-tier-vias, in *Proc. International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2014, pp. 702-709. (CCF 推荐 B 类国际会议)

[3] **Chao Zhang**, Wenjian Yu, Efficient techniques for the capacitance extraction of chip-scale VLSI interconnects using floating random walk algorithm, in *Proc. IEEE ASP-DAC*, Singapore, Jan. 2014, 756-761. (CCF 推荐 C 类国际会议)

[4] **Chao Zhang**, Wenjian Yu, Qing Wang, and Yiyu Shi, Fast random walk based capacitance extraction for the 3D IC structures with cylindrical inter-tier-vias, *IEEE Trans. Computer-Aided Design*, 2015 (SCI 收录，影响因子 1.5) (accepted)

[5] Wenjian Yu, Hao Zhuang, **Chao Zhang**, Gang Hu, and Zhi Liu, RWCap: A floating random walk solver for 3-D capacitance extraction of VLSI interconnects, *IEEE Trans. Computer-Aided Design*, 32(3): 353-366, 2013 (SCI 收录，影响因子 1.5)

[6] Bolong Zhang, Wenjian Yu, and **Chao Zhang**, 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*, 2015, DOI: 10.1002/jnm.2042. (SCI 收录，影响因子 0.6)

[7] 齐明, 赵陈粟, **张超**, 喻文健, 面向高精度寄生参数提取与时延分析的集成电路版图数据转换方法, *计算机辅助设计与图形学学报*, 第 27 卷, 第 6 期, 2015 年

(EI 收录) (to appear)

研究成果

[1] 喻文健, **张超**, 面向集成电路互连电容提取的线网高斯面采样方法与系统, 专利号: 201410016439.3

[2] 喻文健, **张超**, 面向集成电路互连电容参数提取的空间管理数据生成方法, 专利号: 201310388975.1