“蜘蛛机器人”的站立稳定区域计算及运动规划

张竞丹　　董斌　　张华涛

摘要

本文讨论了有腿机器人的运动规划问题。首先提出了蜘蛛模型，并对此模型进行了详细的描述，然后列举了需要解决的问题。在此基础上讨论解决此问题的已有算法。最后，对程序实际采用的算法进行了详细的叙述。

关键词

运动规划　有腿机器人

1.引言

有腿的机器人（legged robots）已经大量存在，目前对于这类机器人的研究主要集中在运动控制上，而这类机器人的运动规划方面的问题却很少有人研究。对于有腿机器人如何在有障碍的危险地区运动的论文基本找不到。但可以看出对这方面的研究是非常有实际意义的，并且必然存在许多有趣的研究课题。

我们有幸找到了“Stable Placements for Spider Robots”和“Motion Planning for Spider Robots”这两篇姊妹文章,对有腿机器人的运动规划问题进行了较深入的研究。论文中建立了有腿机器人的简化模型和运动空间，并在此基础上分别研究了站立稳定区域的计算和运动规划方面的问题。我们在对这两篇文章和其它参考文献的详尽理解的基础上，对这两篇文章所提到的算法进行了一定的简化，并且完成了程序实现。

我们考虑的有腿机器人的名称为蜘蛛机器人（Spider Robots）,它的性质在第二部分详细描述，并且在此模型假定的情况下提出若干有趣的问题；在第三部分，将详细叙述两篇论文中的解题方案和算法；在此基础上，我们在第四部

分讨论了如何对原算法进行简化，并且提出了我们实际采用的算法；第五部分研究了一些计算结果；最后，谈了一些看法和收获。

2.蜘蛛机器人具体描述

蜘蛛机器人的身体是一个质点 G。有若干条腿连接在身体上，这些腿可以在以 G 为圆心，R 为半径的范围内活动，这个圆盘区域称为蜘蛛的活动区域（action range）。

能安全支持蜘蛛的脚的点都在一个平面上，它们构成了由 n 个点构成的集合 M。具体的模型如图 1：

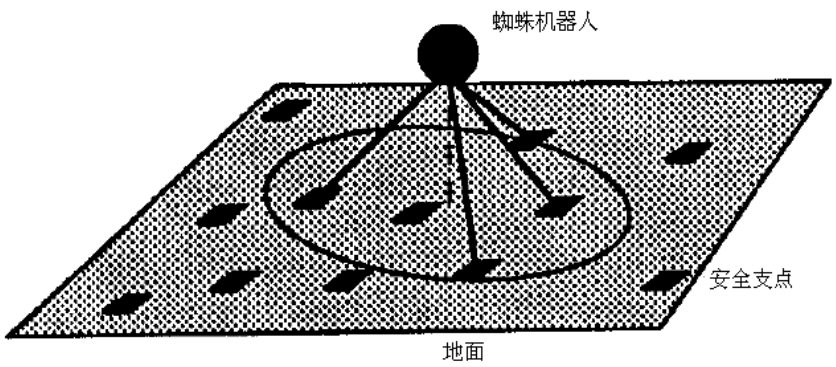

图 1. 蜘蛛机器人

2.1 机器人的站立

我们将蜘蛛身体 G 所在一个位置称为一个形态（configuration）。对于这个站立形态，由集合 M 中的若干个点构成它的站立支点集（placement）。 这个集合被称为 I，满足 Pi∈M，i ∈I，I 中的所有支点构成蜘蛛的一个站立姿势。若是满足| I | =l，我们称这个支点集是l-支点集（l-placement）。

对于一个形态 G，一个支点集 I 若是满足如下的定义：

$$\forall i \in I, Pi \in M \wedge d(G, Pi) \le R \wedge |I| \ge l \qquad (1)$$

其中$d(G,Pi)$是平面上两点间的欧拉距离。则我们称这个支点集是l-可站立的（l-feasible）。同时这个形态 G 也称为是l-可站立的。

对于一个形态 G，一个支点集若是满足如下的定义：

$$G \in CH(\{Pi, i \in I\}) \qquad (2)$$

其中$CH(S)$表示点集 S 的凸集。则这个支点集是稳定的。同时这个形态 G 也是个稳定形态。

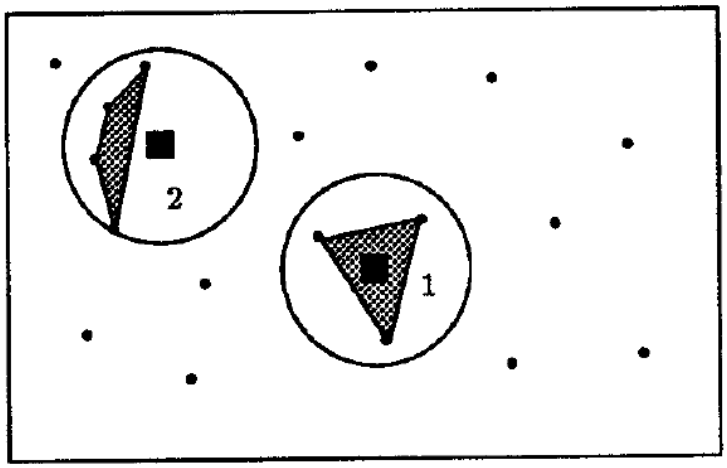

图 2. 一个示例

以图 2 为例，形态 1 是 3-可站立的，形态 2 是 4-可站立的。形态 1 是稳定形态，而形态 2 是非稳定形态。（在图中两个方块分别代表形态 1，2）

对于一个给定支点集合的平面，由活动半径 R 和蜘蛛的腿的个数 l，可以决定一个l-稳定区域（stable region）。这个区域中的任何一个点都是一个l-可站立的且稳定的形态 G。例如对于图 2，点 A 所在的区域是一个 3-稳定区域，而另一个区域是一个 4-稳定区域。

2.2 机器人的运动

在l-稳定区域中的形态（蜘蛛身体）的一条运动轨迹称为l-身体运动轨迹（l-body motion）。同时一个l-运动轨迹（l-motion）是由一个l-身体运动轨迹和蜘蛛在这个运动过程中所有可能的站立支点集构成的。

这里需要注意的是在l-运动中，可能存在一些特殊身体形态，蜘蛛在这些身体形态上需要改变站立支点集。也就是说蜘蛛在这些点上，在不改变身体位置的情况要改变腿的位置，身体才能继续运动。我们可以这样假设蜘蛛模型：蜘

蛛由$l+1$条腿构成，通过额外的脚，它可以保证在始终有l条脚和地面接触的情况下，改变它的落脚点。

2.3 问题

通过前面所描述的模型，我们可以想到若干有趣的问题，其中的几个列举如下：

1. 给定平面上的一组支点，腿的个数l，腿的活动半径 R，求l-稳定区域；
2. 在 1 的基础上给定蜘蛛的初始形态和结束形态，判断是否存在一条从初始形态到结束形态的l-身体运动轨迹；
3. 在 2 的基础上，若存在l-身体运动轨迹，给出l-运动轨迹（即具体给出蜘蛛在运动中每个形态的站立支点集）；
4. 在 3 的基础上，给出最短l-运动轨迹（包括对最短l-运动轨迹的定义，是身体运动距离的最短，还是运动过程中蜘蛛移动腿次数最少意义上的最短）；
5. 若给定的支点集合不存在l-运动轨迹，是否能加入较少的支点，从而形成稳定路径；
6. 对问题 5 是否寻求其最优解的算法是否具有 NP-Hard 的难度；
7. 若考虑对于蜘蛛的每一条腿，可活动的范围大小不等，是否存在算法能在多项式的时间内找到蜘蛛身体能保持稳定的区域；

3.前人的工作

“Stable Placements for Spider Robots”和“Motion Planning for Spider Robots”这两篇分别回答了上面的 1，2，3，4 这四个问题，并且给出了相应的算法，在这里我们将对其进行一个大概叙述。

3.1 l-稳定区域的计算

在“Stable Placements for Spider Robots”中，着重解决了l-稳定区域的计算问题，并且给出了相应的算法，和对算法的复杂度的分析。我们将着重叙述算法实现，并且给出相应的复杂度，但是不作证明。

3.1.1 一些概念和结论

对于任何的$S \subseteq M, |S| \geq 3$，将在欧拉平面定义的满足公式（3）的点集称为由$S$定义的稳定区域（stability region）。

$$R(S) = CH(S) \cap (\bigcap_{p \in S} D(p)) \qquad (3)$$

图 3 显示了由三个点的集合所定义的一个稳定区域。

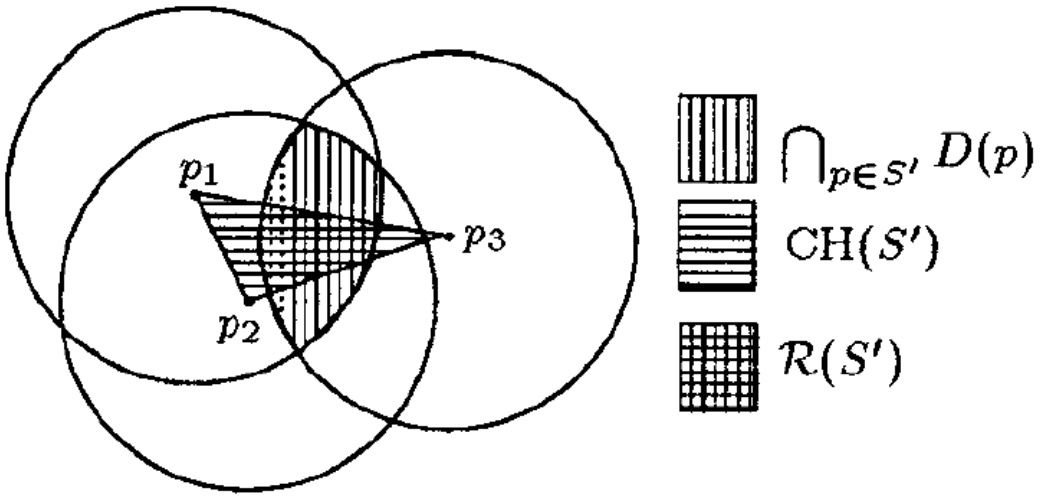

图 3. 由$S = \{p_1, p_2, p_3\}$定义的稳定区域

我们可以得到l-稳定区域的定义：

$$F_l = \bigcup_{T \subset M, |T| = l} R(T) \qquad (4)$$

由这个定义我们就可以得到求l-稳定区域一个方法：从集合M中找出所有有的l个元素的子集T，求$R(T)$，然后做并，就可以求出其稳定区域。但是这种算法的计算量是无法忍受的。需要做优化。

结论一：若有$k > l$, 则有$F_k \subseteq F_l$。

这个结论将不作形式上的证明。但是其结果是比较明显的，一个蜘蛛的腿的个数越少，身体能站立的稳定区域的面积应该越大。也就是说一个区域若是l-稳定区域，则必然是$l-1$-稳定区域，$l-2$-稳定区域，…,3-稳定区域。

对于一个支点集合 M,我们将集合$\{D(p), p \in M\}$称为由若干个圆构成的一个排列 A（arrangement）。其中的每个由圆弧围成的内部不再含有圆弧的小区域称为一个细胞 T（cell）。S(T)定义如下：

$$S(T) \subseteq M, \forall p \in S(T) \Rightarrow T \subseteq D(p) \tag{5}$$

结论二：若T是排列 A 的一个细胞，若有$k = |S(T)|, k \geq l$，则有$F_l \cap T = R(S(T))$。

证明：

由定义知$R(S(T))$是 k一稳定区域，同时$R(S(T)) \subseteq T$，因为$k \geq l$,由结论一可知$R(S(T)) \subseteq F_l$，所以$R(S(T)) \subseteq F_l \cap T$。

同时由$R(S(T)) = CH(S(T)) \cap (\bigcap_{p \in S(T)} D(p)) = CH(S(T)) \cap T$可以看出，若一个点在$T$内是$l$-稳定的，在必然在$CH(S(T))$的内部（因为它在$S(T)$的子集的凸包的内部），则这个点必然是$k$-稳定的。所以有$R(S(T)) \supseteq F_l \cap T$。

综上，命题得证。

例如对于图 3，由竖线区域构成细胞 T，则$S(T) = \{p_1, p_2, p_3\}$则$F_l \cap T = R(\{p_1, p_2, p_3\}) = R(S(T))$。

3.1.2 基本的计算思路

根据结论二，我们可以得到算法的基本的思路：先求出支点集合 M 的排列 A，然后对排列中的每一个细胞T，若是$|S(T)| \geq l$，则求$R(S(T))$，所有的$R(S(T))$的并集就是l-稳定区域。

同时我们注意到两个相邻细胞的T_1, T_2的$S(T_1), S(T_2)$只相差一个支点（从T_1到T_2，$S(T_2)$要么比$S(T_1)$多一个支点，要么少一个点）。这时求$S(T_2)$的凸包可以充分的利用$S(T_1)$的凸包的信息。(利用对原有凸包增加一个点或减少一个点的凸包动态维护算法。)

3.1.3 算法叙述

由此我们可以得到如下的算法：

（1） 求给定支点集合 M 对于活动半径 R 的排列 A；

这里若是采用参考文献[3]求线段相交的算法的改进算法，可以在时间复杂度为 $O((n+k)\log n)$ 的情况下完成。其中 A 的复杂度，也就是 A 中相交圆弧的数目为 $|A|=\Theta(n+k)$，在最极端的情况下 A 的复杂度为 $\Omega(n^2)$，也就是说这一步的时间复杂度为 $O(n^2\log n)$。

（2） 对每一个细胞 T，进行如下计算：

a. 若包含此细胞的圆的个数小于 l，则这个细胞中没有 l-稳定区域，否则找到包含这个细胞的圆的所有的支点的集合；

b. 求支点集合的凸包；

c. 求凸包和细胞 T 的交集，为这个细胞所包含的 l-稳定区域。

这里若是对细胞的检查按照细胞相邻关系顺次进行，则可以充分利用前一个细胞的凸包的信息。采用参考文献[4]的方法，在 $O(\log n)$ 的时间复杂度的情况下得到新的细胞的凸包。

原论文中证明了第（2）步的时间复杂度不超过第（1）步的，所以整个算法的时间复杂度为 $O(n^2\log n)$，空间复杂度为 $O(n^2)$。

3.2 运动规划

一旦我们计算得到了稳定区域，就可以计算机器蜘蛛 l－运动轨迹。根据 l－运动轨迹的定义，在 l－运动轨迹中的 l－身体运动轨迹应该完全在 l－稳定区域中。这个姿态必然属于一个细胞，包含这个细胞的圆的圆心（支点）就是这个姿态的支点集。

因为在蜘蛛的运动规划中我们不考虑碰撞检测等问题，因此在得到稳定区域的情况下计算运动轨迹，和在这个轨迹上的支点集，不是一个困难的问题。但是这里存在一个如何优化搜索的问题。现在简单介绍一下“Motion Planning for Spider Robots”这篇文章中的思路。

3.2.1 计算过程

计算过程大概有这么几步：

a. 将l－稳定区域划分成若干的小的区域，在这些小的区域内的所有的点的支点集都一致。论文中将这个过程称为梯形分解（trapezoidal decomposition）。如图(4)所示:

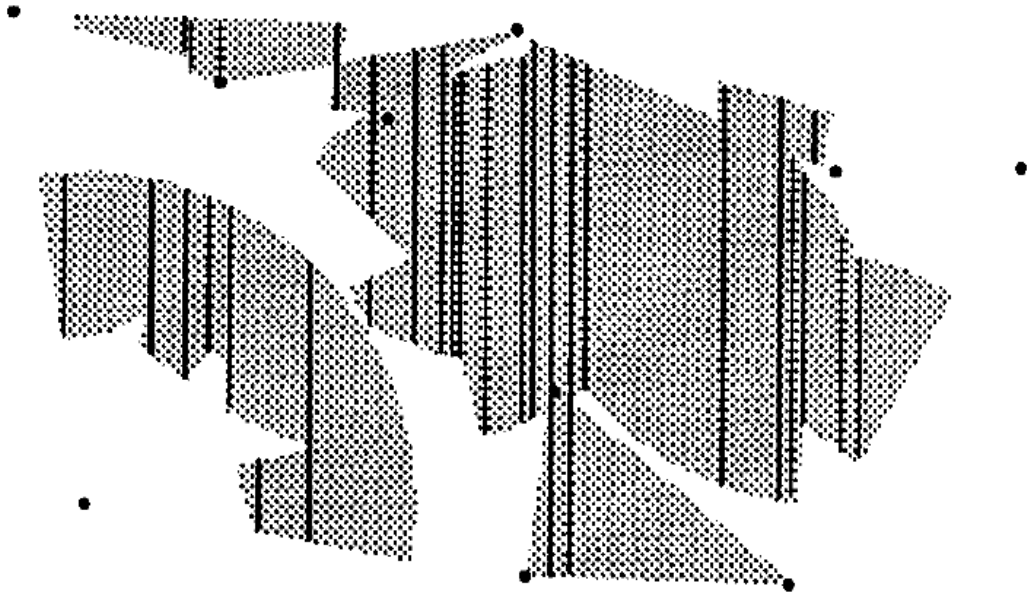

图 4. 梯形分解

b. 将这些小的区域形成相邻的拓扑关系；
c. 通过深度或宽度优先搜索得到蜘蛛的l－身体运动轨迹是由哪几块小区域构成的；
d. 求穿过每块小区域的身体运动的局部最优解，将这些路径段联合起来构成完整的l－身体运动轨迹。

3.2.2 算法评述

对于这个算法的时间复杂度是$O(|F|^2+|A|\log n)$，其中F表示l-稳定区域，A表示以支点为圆心的圆的相交的复杂度。

可以看到这个算法的特点是将稳定区域根据特点进行了数据粗化（分类），这样大大减少了搜索的空间复杂度，可以证明在这样的算法得到的路径是蜘蛛运动需要换支点集最少次数的路径，但不一定是身体移动距离最短的路径。

4.实现算法

在详细阅读了前面论文的基础上，我们对实现算法进行了认真的考虑，当考虑运用论文中所叙述的算法时我们遇到了如下的困难：

(1)原算法的高效率是通过精巧设计算法来实现的，它对算法的设计和数据结构的设计都要有非常高的要求，同时程序也非常的复杂。实际上在原论文中并没有说明如何实现数据结构，但我们在实际考虑的时候发现设计数据结构来满足算法所提的要求是困难的。

(2)原算法的计算在实数域上面进行，其中有大量的集合运算，并且这些集合的边是由圆弧和直线围成，这样导致集合的基本运算非常的复杂，需要写大量的程序，而这不是在短时间内就能完成的。

(3)稳定区域实现梯形划分和小区域拓扑关系的形成实现起来也非常的麻烦。

4.1 简化方式

在考虑了实际遇到的困难和我们能用来实现算法的时间的情况下，我们对原来的算法进行了简化，主要的简化思路如下：

(1)我们对整个问题的计算不在实数域上，而将其离散化，在一个大的点阵上（3000×3000）进行计算，同时对集合的表示将采用点阵上的一些特殊的值的方式来表示，这虽然没有实数计算那么精确，但是数据结构的设计变得比较的简单；

(2)对于凸包的计算，我们采用一般的计算方法，也就是说我们将不利用相邻细胞的凸包动态维护算法；

(3)对于稳定路径的计算将采用广度优先的算法进行计算，因为这个问题现在是在一个固定大小的区域内计算，而且是用点阵形式表示，用广度优先算法虽然不是太优化，但是完全可行，而且是完全可接受的。

4.2 算法思路

在考虑了简化后，我们的算法的思路将变得比较的简单。其中求稳定区域的计算还是按照原论文中的思路，只是对每个细胞的计算不需要按照相邻的关系顺次进行。对于求运动路径，用一般的广度优先的算法。但是因为所计算的空间比较大，要注意程序中对内存的优化，防止内存溢出。同时要注意采用宽度优先，得到的路径是蜘蛛身体运动的最短路径，但是不能保证是蜘蛛需要改变支点次数最短的路径。

4.3 详细算法

最终我们设计了实现算法，在经过简化后，确实算法的实现变得简单了，但是因为采用点阵的方式进行计算，计算量必然比矢量方式的大，因此提高算法效率，减少内存的利用成了设计时的主要的目标。

4.3.1 稳定区域的计算

输入参数：点阵的规模（P×P），支点集 M，活动半径 R，腿的条数l。

计算步骤：

a. 初始化，包括对一个 P×P 的矩阵全部置零；

b. 在一个 2*R+1 的矩阵中，以 R+1 为圆心，构建一个圆形的模板，在圆形内部的点置 1，在外部的置 0；

c. 模板填充：在 P×P 的矩阵中，以每个支点为中心，将 b 中形成的模板中的每一个元素和矩阵中的相应元素相加，得到的新的值存入此矩阵元素中；

d. 此时矩阵中的元素的值，就是包含这个元素的圆的个数；所有相同的值形成的一块区域是就是前面定义的一个细胞，对每一细胞做如下的操作：

(1)找包含这个细胞的圆的支点所形成的集合，若是支点的个数小于 l，则将这个细胞的值全部设为零；

(2)否则构造这个支点集合的凸包，将凸包内部的点加上一个特殊的值（10000）；

(3)将属于凸包内部，同时又属于细胞内部的点进行特殊的标定，表示属于稳定区域。对细胞中不属于凸包的点的值全部设为零（通过矩阵中点的值进行判断）；

(4)将凸包中不属于细胞的点的值减掉刚才所加上的特殊值；

(5)若此细胞内存在稳定的点，则记录支点集，这个支点集将构成蜘蛛的落脚点；

以上五步完成了求细胞内稳定区域的过程；

a, b, c, d 四步完成后，特殊标定的点就是稳定区域；

4.3.2 路径规划

路径规划采用宽度优先算法，简单叙述如下：

a．对输入的起始点和结束点，判断是否在稳定区域内，若是不在则退出；

b．建立一个队列，将起始点放入队列中；

c．顺次对队列中每一个点进行如下的操作：

（1） 将这个点从队列中取出；

（2） 考察这个点的八个方向的点，对于还没有扩展到的点，若这个点是稳定点，则将这个点进行标记，并将这个点放入到队列的最后，同时记录新得到的点是如何由原来点扩展得到的；若新的点就是结束点，则结束；

d．第 c 步的结束方式有两种，若是扩展完队列中的点仍然无法扩展到结束点，则不存在起始点到结束点的路径；否则找到了运动路径，根据扩展信息得到完整的路径；

4.3.3 算法分析

分析可知上面的算法的复杂度是和求解的点阵的大小紧密相关的。求稳定区域的计算是反复的对点阵中的每一个点进行赋值和检测工作，算法的时间复杂

度是$O(n^2)$级的，空间复杂度也是$O(n^2)$的。求路径的宽度优先算法，在最坏的情况下需要对整个点阵进行遍历，因此复杂度也是$O(n^2)$级的。

因此整个程序计算复杂度是由点阵的大小决定的，经过测试，发现程序对于2000×2000 的点阵的计算，在时间和空间上都是可接受的。

5. 具体实现

在上面讨论的基础上，我们进行了程序实现，采用了 Visual C++ 6.0 作为实现工具。所用的机器的配置为 P3－650，内存 128M。

5.1 软件功能说明

在我们实现的软件中，主要包括如下的几个功能：

1. 点阵的编辑功能：能对点阵进行编辑，包括随机增加若干个点，增加特定的点，删除特定的点；
2. 点阵的存取功能：能对编辑完成的点阵存磁盘，以及从磁盘中将事前编辑好的点阵取出；
3. 稳定区域的计算功能：能针对所给定的数据计算蜘蛛可站立的稳定区域（其中蜘蛛的脚的个数和活动半径可变）；
4. 路径规划功能：能根据给定的起始点和结束点，计算蜘蛛的运动轨迹，并且指出在运动过程中的每一个位置的支点集合；

6. 数据结果分析

通过程序的计算，我们可以得到一些直接的结果：对于蜘蛛的半径固定的情况，蜘蛛的腿越多，计算得到的稳定区域的面积越小；在蜘蛛的腿的个数固定的情况下，蜘蛛的活动半径越大，则稳定区域的面积越大。

对于计算中的一些特殊值的考虑，如蜘蛛腿的数目为 2 等，我们通过在输入界面上做一些限制来防止这样的情况出现。

经过测试，我们发现程序对于 3000×3000 点阵，有 30000 个支点的情况，程序可以在 10 分钟左右的时间得到计算结果。

7. 结束语

有腿机器人的运动规划存在许多值得研究的问题，而且许多都是有现实意义的，对于这方面需要有好的抽象模型，好的算法。

可以看到我们所完成的程序在许多方面有值得改进的地发，如在写程序的过程中我们发现许多方面都直接受到了点阵不能过大的限制，因此若是能实现在连续空间计算的算法，将能更有效的解决这个问题。

同时对于论文中所讨论的这个模型，还有许多问题可以研究，包括前面提到但是没有解决的问题。另外，在这个模型中没有考虑存在障碍物和身体不是一个点的情况，也就是说回避了碰撞检测等问题。若是模型中加上这些因素，整个问题将会变得更加有挑战性。

参考文献

[1] J-D. Boissonnat, O. Devillers, L. Donati, and F. Preparata ,"Stable Placements for Spider Robots" ,In 8th ACM Symposium on Computational Geometry in Berlin, June 1992.

[2] J-D. Boissonnat, O. Devillers, L. Donati, and F. Preparata. "Motion planning for a spider robot",In IEEE International Cortference on Robotics and Automation, May 1992.

[3] F.P. Preparata and M.1. Shames, "Computational Geometry : an Introduction", Springer-Verlag, 1985.

[4] J. Hershberger and S. Suri, "Offline maintenance of planar configurations", In ACM-SIAM Symposium on Discrete Algorithms, pages 3241, 1991.

后记

做这个计算几何的大作业真是一个痛苦的过程，查资料，讨论，写论文，写程序，忙得不亦乐乎。还好，总算是弄完了，还以为这一天永远都不会到来。

但是，毕竟大家都还是有不少收获的。首先，大家增加了相互之间的了解，在做试验之前，我们虽然是一个大组的，但是有两个同学是外校考进来的，基本上都不太了解。现在彼此都相互熟悉了，也许过个十年八年，大家学过什么，做过什么都忘了，但是这段友谊还能延续下去。其次，通过做这个试验，我发现团结和责任心是如何的重要，一个团队要能成功，需要每个人能无私的奉献，能积极为团队贡献自己的力量，这一点是我在做试验中深刻体会到的。最后，我们通过做这个试验，对计算几何的一些概念也有了更多的了解，并且学习了如何将理论运用到实际问题中去，这对我们的发展是相当有好处的。

感谢我们组的同学这几天的辛勤努力，特别是要感谢董斌同学，感谢他在期末考试繁忙中抽出时间写程序，没有他整个程序就不能完成。同时也希望大家能把这样的团队精神用到以后的工作学习中去。

也要感谢邓老师对我们的辛勤教导，在他的指导下，我们确实对计算几何这门学科有了必要的了解。同时，我感觉这门课是组织比较成功的，包括对网络的充分的利用，这门课也是这学期我出席最多的课（好像缺席了一次）。真心希望这门课能够越办越好。

张竞丹

1/12/01