高技术通讯 2013 年 第23 卷 第3 期:219 ~227

doi:10.3772/j.issn.1002-0470.2013.03.001

BSCTC:传感网的基于切向约束的B样条等值线查询算法①

郭龙江②*** 孙毅辉*** 刘 勇③*** 段金晟***

(*黑龙江大学计算机科学技术学院 哈尔滨 150001)

(**黑龙江省数据库与并行计算重点实验室 哈尔滨 150001)

摘 要 针对目前传感网的等值线查询方法需要返回较多代表节点,代表节点选取的计算复杂度高且等值线还原结果不理想的情况,提出了一种基于切向约束(TC)的B条等值线(BSC)查询算法,简称BSCTC。该算法基于切向约束的2次B样条插值原理,首先选出等值线中的代表节点并传输代表节点信息给Sink,然后在Sink端对返回的代表节点进行分段还原,形成等值线。理论分析表明:BSCTC算法返回的代表节点期望数只是网络等值线节点数的39%,代表节点选取的计算复杂度为$O(n)$(n为等值线节点个数)。实验结果表明:与目前最好的DABC算法相比,BSCTC算法返回的代表节点数减少了53%,而且可形成更精确的等值线。

关键词 传感器网络,等值线监测,切线约束,B样条,插值

0 引 言

传感器网络主要应用在对物理环境的监测上,如温度、湿度的监测[1-3]。近年来,一些学者将等值线[4-6]技术引入到传感网的环境监测中。等值线监测技术用等值线表示环境监测值的分布,生成的是反映网络区域环境分布的全景图。若是所有节点都将各自的监测信息转发至Sink,往往会造成传输阻塞,损失较多的能量。实际上,等值线查询技术返回部分节点信息就可形成宏观环境分布图,节省了网络能量。等值线图根据等值线监测数值梯度的变化,反映出区域环境的变化,可以用于传感网的事件检测[7]。

等值线查询是指将等值线上的代表节点的信息返回至Sink,Sink根据代表节点的坐标与其它相关信息就可以还原最初的等值线。Sink首先广播查询Q,Q包含感知数值信息。例如,Q包含一个温度值,如90℃。网络中若某个节点的监测值与Q包含的感知值误差小于用户给定的阈值ε,则该节点向Sink返回监测值。这样的节点称为等值线节点。网络中的等值线节点经过代表节点选取算法选出代表节点后,只需代表节点向Sink返回数据。在Sink端,采用等值线还原算法生成拟合等值线。目前的传感网等值线查询方法[8-10]存在两种问题:(1)需要返回较多代表节点;(2)代表节点选取的计算复杂度高,并且等值线还原结果不理想。针对这两种问题,本文提出了一种基于切向约束(tangent constrain, TC)的B样条[11]等值线(B-spline contour, BSC)查询算法,简称BSCTC算法。该算法根据角度判断规则选择代表节点并将代表节点的信息返回至Sink,在Sink端利用基于切向约束的2次B样条插值算法分段还原返回的代表节点,拟合出等值线,是一种高能量效率的精确的等值线查询算法。

1 现有等值线查询算法的问题分析

为了获得网络中等值线分布图,最常规的做法是收集所有节点的监测数据,根据监测值构造各个等值线。然而,当大量监测数据都向Sink转发时,

① 国家自然科学基金(61033015,60803015),教育部新世纪优秀人才支持计划(NCET-11-0955),哈尔滨市青年科技创新人才研究项目(2008RFQXG107,2012RFQXG096),黑龙江省教育厅高校新世纪优秀人才支持计划(1252-NCET-011)和黑龙江省教育厅创新团队项目(2011PYTD002)资助。

② 男,1973年生,博士,教授;研究方向:并行与分布式计算,传感器网络,数据库;E-mail: longjiangguo@gmail.com

③ 通讯作者,E-mail: liuyong001@hlju.edu.cn

(收稿日期:2011-10-08)

网络产生信息阻塞的概率会很大,同时节点会消耗较大的能量。为了解决该问题,研究人员提出了各种等值线聚集查询算法[7,9]。这些算法通过中间节点将相同监测值的节点信息转发至 Sink。尽管这些算法可降低网络中的信息拥塞,但是网络中依然要传输所有等值线节点的信息,所以网络整体的传输量仍为 $O(n)$[8,10]。

为了减少网络传输能量,目前传感器网络主要采用两类等值线查询算法:一类算法将所有等值线节点信息全部传输至 Sink,然后在 Sink 端生成等值线;另一类算法只需返回部分等值线节点信息,等值线节点信息返回后,在 Sink 端拟合出等值线。第一类算法的特点是所有等值线节点都向 Sink 返回信息,其优点是可以形成精确的等值线图。然而,在网络规模较大,节点分布较密的情况下,返回至 Sink 的节点数目太多。ISO-MAP (isobath-contour map)[8,10]算法就是一个例子,该算法不仅返回所有的等值线节点至 Sink,而且每个节点返回的信息还包括监测值、节点坐标以及节点梯度。所有节点都返回信息,势必会消耗大量能量,缩短网络生存周期。第二类算法先将等值线分成若干组,每组选择若干个代表节点返回至 Sink,Sink 根据返回的代表节点信息拟合出该组等值线。基于 Bezier 曲线的数据聚集(data aggregation based on Bezier curves, DABC)算法[12]对每组等值线只返回 4 个代表节点的信息,降低了信息传输量,但是所有等值线节点为了判断是否向 Sink 返回信息,必须运行复杂的 Bezier 插值拟合算法。Sink 端接到所有代表节点的信息后,执行逆 Bezier 差值算法拟合出等值线。该算法虽然降低了返回的节点数,但是每个节点的计算复杂性增加了,而且在 Sink 端生成的等值线效果并不十分理想。针对上述算法存在的问题,本文提出了新的等值线查询算法 BSCTC。

2 BSCTC 算法

BSCTC 算法有两部分功能。第一部分是选取代表节点,其核心思想是根据角度判断规则,将大量的等值线节点分组,每组向 Sink 返回 3 个代表节点;第二部分是在 Sink 端根据返回的代表节点信息还原出等值线。

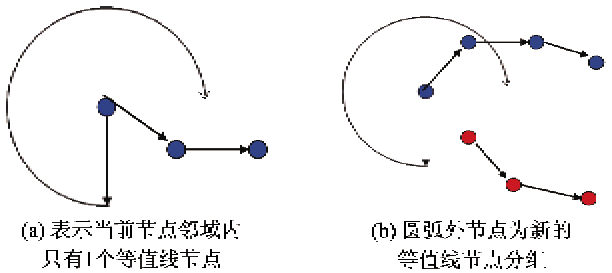
2.1 选取代表节点

Sink 向网络广播一个查询 Q,Q 包含感知数值信息。网络中接收到查询 Q 的节点首先判定自己是否为等值线节点。如果节点的监测值 p 满足 $|p-Q| \leqslant \varepsilon$,$\varepsilon$ 为用户预先制定的阈值,则该节点为一个等值线节点。

选取代表节点的核心思想是将等值线节点分组,每组节点中选取 3 个节点向 Sink 返回数据。这 3 个节点是分组中的起始节点、校正节点和分组中的终止节点。每个节点返回的信息包括 3 个参数 (v, p, s),$v=(x_v, y_v)$ 为节点坐标,p 表示等值线的监测值,表示该点与相邻代表节点之间连线的斜率。

等值线节点确定出来后,每组等值线节点需要选择 3 个等值线节点进行初始化判断。首先,随机的选择一个等值线节点作为第一组节点的起始节点 $v_0=(x_0, y_0)$,起始节点从它正下方开始在它一跳范围内顺时针方向选择最近的节点作为下一个判断点 $v_1=(x_1, y_1)$,计算出起始点的斜率 $s_0=(y_1-y_0)/(x_1-x_0)$。此时,起始节点向 Sink 返回信息,返回数据为 (v_0, p, s_0)。节点 v_1 计算它的斜率 $s_1=(y_1-y_0)/(x_1-x_0)$,将 s_1 和位置信息发送到它的下一个判断点 v_2。最后当前判断节点 v_2 计算出它的斜率 $s_2=(y_2-y_1)/(x_2-x_1)$ 和该段等值线节点的初始角度 α_1($\tan\alpha_1=|s_1-s_2|/|1+s_1s_2|$)以及角度方向,等值线节点选取初始化完成。本文给角度规定两个方向:向上和向下。若直线 v_0v_1 的斜率大于直线 v_1v_2 的斜率,则 v_0v_1 和 v_1v_2 的夹角角度方向为向下;反之角度方向为向上。

如果起始节点一跳范围内仅有一个节点为等值线节点,则该节点为下一个判断点(见图 1(a));如果该节点一跳范围内有 1 个以上的等值线节点,从节点正下方开始选择顺时针方向的第一个节点作为下个判断节点,它邻域的其他等值线节点则作为新的起始点查找它的下一个判断点(如图 1(b))。



(a) 表示当前节点邻域内只有1个等值线节点　(b) 圆弧外节点为新的等值线节点分组

图 1 判断点选择

当等值线节点选取的初始化完成后,将按照以下规则判断节点是否向 Sink 返回数据。

假设 v_i 是当前的判断节点,则 v_i 的斜率为它与

上一个判断点 v_{i-1} 之间连线的斜率,该斜率的计算公式为 $s_i = (y_i - y_{i-1})/(x_i - x_{i-1})$,$(x_i, y_i)$ 为 v_i 的坐标,(x_{i-1}, y_{i-1}) 为 v_{i-1} 的坐标。v_i 是否向 Sink 返回信息,取决于节点 v_{i-1}、v_i 和 v_i、v_{i+1} 之间的角度关系。当 v_{i+1} 向 v_i 返回位置信息,v_i 可得到 v_{i+1} 的斜率 s_{i+1},$s_{i+1} = (y_{i+1} - y_i)/(x_{i+1} - x_i)$,$(x_{i+1}, y_{i+1})$ 为 v_{i+1} 的坐标。节点 v_i 根据公式 $\tan\alpha_i = |s_i - s_{i+1}|/|1 + s_i \cdot s_{i+1}|$,计算出直线 $v_{i-1}v_i$ 和 v_iv_{i+1} 之间的夹角 α_i(见图 2)。

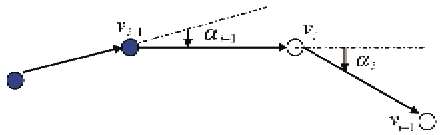
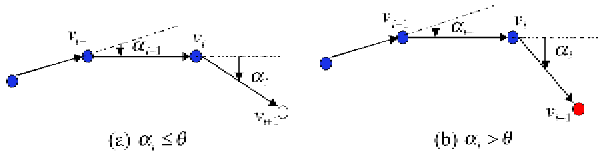


图 2　计算夹角

根据 α_i 与 α_{i-1} 和初始角度 α_1 方向的关系,分为两种情况,判断 v_i 是否向 Sink 返回信息。

(1) α_i,α_{i-1} 和 α_1 方向相等。v_i 比较系统的夹角阈值 θ 与 α_i 的大小,这里 $\theta \in [0, \pi/2]$。θ 能决定选取代表节点的数量和 Sink 所恢复的等值线的精度。θ 的选取取决于节点的距离和等值线节点的平滑分布。实验结果显示,当 $\theta \in [\pi/4, \pi/3]$ 时,可均衡节点返回数量和拟合效果。

若 $\alpha_i \leqslant \theta$,则节点 v_i 不向 Sink 返回信息。此时,节点 v_{i+1} 成为新的当前节点。v_{i+1} 按照图 1 的方法去搜索它的下一个判断节点(见图 3(a));若 $\alpha_i > \theta$,则节点 v_i 作为当前分组的终止节点,并向 Sink 传回信息 (v_i, p, s_i)。节点 v_{i+1} 成为新的等值线节点分组的起始节点去查找它的下一个判断节点(见图 3(b))。



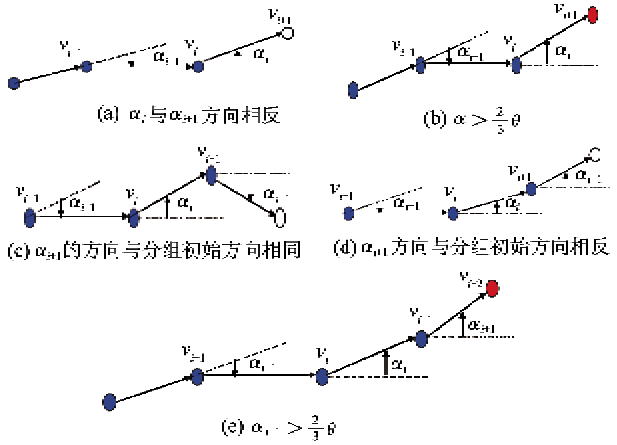
(a) $\alpha_i \leqslant \theta$　　(b) $\alpha_i > \theta$

图 3　代表节点选取规则 1

(2) α_{i-1} 与 α_1 同向,α_{i-1} 与 α_i 反向(见图 4(a))。

若 $\alpha_i > 2\theta/3$,则节点 v_{i+1} 作为等值线节点新分组的起始节点(见图 4(b))。

若 $\alpha_i \leqslant 2\theta/3$,则节点 v_i 不将数据返回至 Sink,节点 v_{i+1} 自己的斜率 s_{i+1} 更改为节点 v_i 的斜率 s_i。节点 v_{i+1} 搜索到它的下一个判断节点 v_{i+2}。若 α_{i+1} 的方向与该分组初始的角度方向相同,则:(i)若 $\alpha_{i+1} < \theta$,节点 v_{i+1} 不将数据返回至 Sink,节点 v_{i+2} 继续查找它的下一判断节点(见图 4(c));(ii)若 $\alpha_{i+1} \geqslant \theta$,$v_{i+1}$ 作为该分组的终止节点向 Sink 转发数据 (v_{i+1}, p, s_{i+1}),节点 v_{i+2} 作为下一分组的起始节点查找它的下一判断节点。若 α_{i+1} 的方向与该分组初始的角度方向相反,则:(i)若 $\alpha_{i+1} \leqslant 2\theta/3$,节点 v_{i+1} 不将数据返回至 Sink,节点 v_{i+2} 的斜率更改为节点 v_{i+1} 的斜率,继续查找下一判断节点(见图 4(d));(ii)若 $\alpha_{i+1} > 2\theta/3$,节点 v_{i+1} 作为该分组的终止节点向 Sink 转发数据 (v_{i+1}, p, s_{i+1}) 数据。节点 v_{i+2} 作为下一分组的起始节点搜索它的下一判断节点,直到所有等值线节点都遍历完成(见图 4(e))。



(a) α_i 与 α_{i+1} 方向相反　　(b) $\alpha > \frac{2}{3}\theta$

(c) α_{i+1} 的方向与分组初始方向相同　　(d) α_{i+1} 方向与分组初始方向相反

(e) $\alpha_{i+1} > \frac{2}{3}\theta$

图 4　代表节点选取规则 2

上述方法可将网络中的等值线节点分组,每组只选择初始节点和终止节点向 Sink 返回数据。但是为了确保在 Sink 端恢复等值线的精确性,需要在每组等值线节点中选择一个校正节点返回 Sink,每组节点向 Sink 返回 3 个节点信息,即初始节点、校正节点和终止节点。校正节点的选择算法如下:

假设 v_i 为当前的判断节点,而且 v_i 所在分组的等值线节点中还没有产生校正节点,则 v_i 依概率 $p = (e^{i\alpha_i} - e^{-i\alpha_i})/(e^{i\alpha_i} + e^{-i\alpha_i})$ 成为校正节点并将数据转发至 Sink,但 v_i 向 Sink 返回的斜率是由上一判断点 v_{i-1} 和下一判断点 v_{i+1} 所确定,其斜率 s_i' 计算公式为:$s_i' = (y_{i-1} - y_{i+1})/(x_{i-1} - x_{i+1})$。因此,校正节点 v_i 向 Sink 返回数据 (v_i, p, s_i')。

至此,等值线节点的每个节点分组均选择出各自的初始节点,校正节点和终止节点作为代表节点向 Sink 返回信息,下一节介绍 Sink 根据返回的代表节点信息拟合出等值线。

代表节点选取算法的伪代码如下:

代表节点选取算法

输入：N 个等值线节点和网络的角度门限 θ

输出：若干代表节点

1. WHILE $N>3$ DO
2. 从 N 个等值线节点中随机的选择一个节点作为该组等值线节点的起始节点 v_i，$N = N-1$，$s = 1$
3. 选择 v_i 的后两个判断节点 v_{i+1} 和 v_{i+2}，计算本组的初始角度 α，$N = N-2$
4. WHILE $N\neq 0$ 且 $s=1$DO
   当前判断节点 v_{i+2} 查找下一个判断节点 v_{i+3}，计算 α_{i+2}，$N = N-1$
5. IF（α_{i+2} 与 α 方向相同）
6. IF（$a_{i+2} <= \theta$ 且 v_{i+2} 不是校正节点）
7. v_{i+2} 不返回信息，i ++
8. ELSE IF（v_{i+2} 是校正节点）
9. v_{i+2} 作为该组校正节点返回信息，i ++
10. ELSEv_{i+2} 作为该组终止节点返回信息 $s = 0$
11. ELSE
12. IF（$a_{i+2} <= 2\theta/3$ 且 v_{i+2} 不是校正节点）
13. v_{i+2} 不返回信息，i ++
14. ELSE IF（v_{i+2} 是校正节点）
15. v_{i+2} 作为该组校正节点返回信息，i ++
16. ELSE v_{i+2} 作为该组终止节点返回信息，$s = 0$
17. END WHILE
18. END WHILE

2.2 还原等值线

当 Sink 接收到代表节点后，对各个等值线分组分别进行拟合。拟合算法是采用基于切向约束的二次 B 样条插值每个分组中的 3 个代表节点，算法可以将每个分组的 3 个节点按照其斜率方向拟合成曲线[11]。

二次 B 样条曲线拟合 n 个顶点的式子如下：

$$c(t) = \sum_{i=0}^{n} d_i N_{i,2}(t) \quad t \in [t_2, t_{i+1}] \tag{1}$$

其中 $d_0, d_1, \cdots, d_n$ 为曲线的控制顶点，$N_{i,2}(t)$ 为二次 B 样条的样条基函数，由向量 $\boldsymbol{T} = (t_0, t_1, \cdots, t_{n+3})$ 确定，其中 $t_0 = t_1 = t_2 \leqslant t_3 \leqslant \ldots \leqslant t_{n+3}$。

当 $t_0 = t_1 = t_2, t_{n+1} = t_{n+2} = t_{n+3}$ 时，二次 B 样条曲线通过相应的传感器节点，满足本文算法的要求。

k 次 B 样条的基函数的计算公式如下所示：

$$N_{i,k}(t) = \frac{t-t_1}{t_{i+k}-t_i}N_{i,k-1}(t) + \frac{t_{i+k+1}-t}{t_{i+k+1}-t_{i+1}}N_{i+1,k-1}(t)$$

$$N_{i,0}(t) = \begin{cases} 1, & t_i \leqslant t < t_{i+1} \\ 0, & \text{其他} \end{cases} \tag{2}$$

本算法中用的是基于切向约束的 2 次 B 样条插值，因此可以根据节点向量 $\boldsymbol{T} = (t_0, t_1, \cdots, t_{n+3})$ 和式(2)求出 $N_{i,2}(t)$，计算公式如下：

$$N_{i,2}(t) = \begin{cases} \dfrac{t-t_i}{t_{i+2}-t_i}\cdot\dfrac{t-t_i}{t_{i+1}-t_i}, & t_i \leqslant t < t_{i+1} \\ \dfrac{t-t_i}{t_{i+2}-t_i}\cdot\dfrac{t_{i+2}-t}{t_{i+2}-t_{i+1}} + \dfrac{t_{i+3}-t}{t_{i+3}-t_{i+1}}\cdot\dfrac{t-t_{i+1}}{t_{i+2}-t_{i+1}}, & t_{i+1} \leqslant t < t_{i+2} \\ \dfrac{t_{i+3}-t}{t_{i+3}-t_{i+1}}\cdot\dfrac{t_{i+3}-t}{t_{i+3}-t_{i+2}}, & t_{i+2} \leqslant t < t_{i+3} \\ 0, & \text{其他} \end{cases} \tag{3}$$

控制顶点的求法如下：给定平面内的 n 个数据顶点 $v_1, v_2, \cdots, v_n$，顶点斜率 $s_1, s_2, \cdots, s_n$，序列按下式求得：

$$d_i = \begin{cases} (v_i + v_{i+1})/2, & \beta_i = 0 \\ \text{直线 } l_i \text{ 与 } l_{i+1} \text{ 的交点}, & \beta_i \end{cases} \tag{4}$$

其中 $d_0 = v_1$，$d_n = v_n$，$i = 1, 2, \cdots, n-1$。l_i 是斜率为 s_i 且过顶点 v_i 的直线，β_i 是直线 l_i 和 l_{i+1} 的夹角（见图 5）。

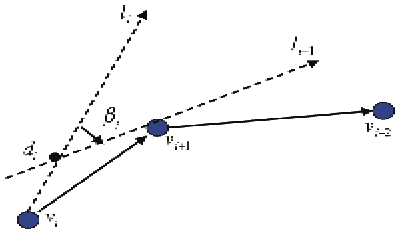


图 5 确定 l_i 和 l_{i+1} 的夹角 β_i

节点向量 $\boldsymbol{T} = (t_0, t_1, \cdots, t_{n+3})$ 按下式计算：

$$\begin{cases} t_0 = t_1 = t_2 = c_1, & t_3 = c_2 \\ t_{i+1} = t_i + \lambda_i(t_i - t_{i-1}), & i = 3, 4, \cdots, n \\ t_{n+3} = t_{n+2} = t_{n+1} \end{cases} \tag{5}$$

其中 $\lambda_i = \dfrac{|d_{i-1} - v_{i-1}|}{|v_{i-1} - d_{i-2}|}$。

式(5)中的 c_1 和 c_2 是满足 $0 \leqslant c_1 \leqslant c_2$ 的任意两常数。为了计算方便，本文将节点向量 $\boldsymbol{T}$ 规范化至 $[0,1]$，则节点向量为

$$\boldsymbol{T} = (0,0,0,\frac{t_3}{t_{n+1}},\cdots,\frac{t_n}{t_{n+1}},1,1,1) \tag{6}$$

把式(5)所求的节点向量带入式(2)得到基函数 $N_{i,2}(t)$，再把 $N_{i,2}(t)$ 和式(4)得到的控制点序列 $d_0, d_1, \cdots, d_n$ 带入式(1)，即可求出经过 n 个顶点的曲线方程（图 6）。

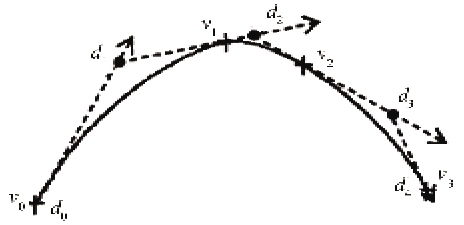


图 6　4 个顶点的曲线拟合

本节对从网络接收到的等值线节点的分组分别进行基于切向约束的 2 次 B 样条插值拟合。由于每个分组只返回 3 个节点信息,因此式(1)中的 $n = 3$;再把这 3 个节点返回的位置信息和斜率带入式(4)可以计算出每段分组的控制顶点序列 d_0, d_1, d_3, d_4,把顶点序列带入式(5),可得到该分组的节点向量 $\boldsymbol{T} = (t_0, t_1, \cdots, t_6)$,最后带入式(6)将节点向量规范化到[0,1]区间。

将计算得到的节点向量 $\boldsymbol{T} = (t_0, t_1, \cdots, t_6)$ 带入式(3)可以求出该分组的基函数 $N_{i,2}(t)$。根据得到的基函数 $N_{i,2}(t)$ 和控制顶点序列 d_0, d_1, d_3, d_4,带入式(1)可以计算出此等值线节点分组对应的拟合的等值线。此时式(1)中的 $n = 3$。

图 7(a)和 7(c)分别为网络真实等值线节点,图 7(b)表示图 7(a)经过等值线节点选择后,向 Sink 返回 3 个方块节点 v_0、v_5、v_9,其中 v_0 为该段分组的起始节点,v_5 为校正节点,v_9 为终止节点,曲线为本文算法根据这 3 个节点产生的等值线。图 7(d)表示图 7(c)经过等值线节点选择后,将图 7(c)中的等值线节点分为两组,第一组向 Sink 返回 v_0, v_3, v_5,第二组向 Sink 返回 v_6, v_9, v_{11}。每组 3 个节点分别对应各组的起始节点,校正节点和终止节点。曲线为本文算法根据这 6 个节点产生的等值线。

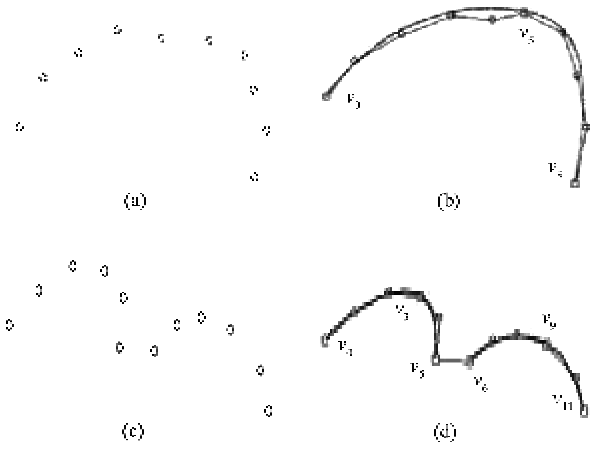


图 7　拟合等值线

在 Sink 端等值线还原算法的伪代码如下:

Sink 等值线还原算法	
输入:M 段分组代表节点	
输出:分段等值线	
1	WHILE $M > 0$ DO
2	用当前分组返回的 3 个等值线节点带入式(4),求出该组的控制顶点。
3	利用式(5)求出节点向量。
4	利用式(3)求出该组的基函数。
5	将得到的控制顶点序列和基函数带入式(1)得到该组的等值线,$M = M - 1$。
6	END WHILE

3　BSCTC 算法理论分析

经过代表节点选取后,网络中的等值线节点将会被分为若干组,每组向 Sink 返回 3 个节点,从而减少传输节点的数量,节省了网络能量。而且网络中的等值线节点只需计算节点在分组内的角度并与角度阈值 θ 作比较,判断是否返回信息,所以网络中的等值线节点计算代价很低。

定理 1 当相邻节点之间距离为节点传播半径时,BSCTC 算法返回的节点数最多。

证明:图 8(a)显示了相邻节点之间距离小于节点传播半径时的情况。设 v_i 为当前判断节点,v_{i+1} 为它的下一判断点,v_{i-1} 为它的上一判断点。v_{i-1} 与 v_i 之间距离小于传播半径。按照等值线节点的选取规则,如果 $v_i v_{i+1}$ 在区域 s_4 中,则 v_i 不返回数据。下面证明 v_{i+1} 不会在区域 s_3 和 s_5 中。因为按照代表节点选取规则,v_{i-1} 从它正下方开始在一跳范围内顺时针方向选择最近的节点作为下一个判断点,如果 v_{i+1} 在扇形区域 s_3 或 s_5 中,那么 v_{i+1} 应为 v_{i-1} 的下一判断点,而不是 v_i,所以 v_{i-1} 不会在区域 s_3 和 s_5 中。设以 v_i 为圆心的圆为 S_c。如果 v_i 返回数据,则 v_{i-1} 一定处在区域 $s - (s_3 + s_5 + s_4 - s_3 \cap s_4)$ 中。

图 8(b)显示了相邻节点之间距离等于节点传播半径时的情况。类似于上面的证明,如果 v_i 返回数据,则 v_{i+1} 处在区域 $s - s_1 - s_2$ 中。

因为 $s_3 > s_1, s_2 = s_4$ 且 $s_5 > s_3 \cap s_4$,所以 $s - (s_3 + s_5 + s_4 - s_3 \cap s_4) < s - s_1 - s_2$。因此,当相邻节点之间距离为节点传播半径时,$v_i$ 返回数据的概率大。所以当相邻节点之间距离为节点传播半径时,返回节点数最多。

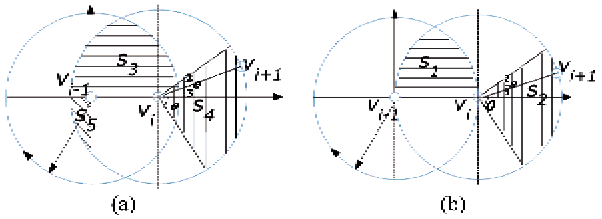


图 8 引理 1 和定理 1 中的证明图

定理 2 当相邻节点之间距离为节点传播半径时，代表节点的期望返回数为 $(\frac{2}{3}+\frac{\sqrt{3}\pi}{4}-\frac{5\theta\pi}{6})\cdot n$，其中 n 为等值线节点数，θ 为节点选取时的夹角阈值。

证明：如图 8(b)所示，设以 v_i 为圆心，半径为传播半径的圆为 S，其面积为 $A(S)$，区域 S_1 和 S_2 的面积分别为 $A(S_1)$ 和 $A(S_2)$。因此，v_i 不返回数据的概率为 $(A(S_1)+A(S_2))/A(S)=(\frac{1}{3}-\frac{\sqrt{3}\pi}{4})+\frac{5\theta\pi}{6}$。所以，$v_i$ 返回数据的概率为 $p=1-(\frac{1}{3}-\frac{\sqrt{3}\pi}{4}+\frac{5\theta\pi}{6})$。设 X 为返回到 Sink 的节点数，X_i 为概率 p 的 0－1 分布。$E(X)=E(\sum_{i=1}^{n}X_i)=p\cdot n=(\frac{2}{3}+\frac{\sqrt{3}\pi}{4}-\frac{5\theta\pi}{6})\cdot n$。

根据定理 2 当 $\theta=\pi/2$，$(2/3+\sqrt{3}\pi/4-5\theta\pi/6)\cdot n$ 取最小值 0.39 n，即返回的代表节点期望数只是网络等值线节点数的 39%。

根据定理 1 和定理 2 可知，当相邻节点之间距离小于节点传播半径，并且 $\theta=\pi/2$ 时，返回的代表节点数要小于网络等值线节点数的 39%。

定理 3 运用 BSCTC 代表节点选取算法，网络所有等值线节点的计算代价为 $O(n)$。

证明：设 v_i 为当前的判断节点，v_i 根据下一判断点 v_{i+1} 的位置信息，计算出斜率 s_i，根据上一判断节点 v_{i-1} 传递的斜率信息 s_{i-1}，可计算出 v_i 的角度 α_i。$\alpha_i=\arctan(|s_i-s_{i-1}|/|1+s_{i-1}s_i|)$。所以节点的整个计算都是在常数时间内完成，即每个等值线节点的计算代价为 $O(1)$。设网络中的等值线节点数为 n，则网络所有等值线节点的计算代价总和为 $O(n)$。

4 实 验

4.1 实验准备

本研究首先进行模拟实验。分别产生了等值线节点数为 60，72 和 56 的 3 组随机节点，并且这 3 组中节点之间的平均距离相同。每组中的空心节点代表网络中的等值线节点。分别使用 BSCTC 算法和 DABC[12]算法对等值线节点进行选取并发送至 Sink，Sink 端对返回的节点进行拟合输出等值线。实验对比了 DABC 算法与 BSCTC 算法返回的等值线节点数，拟合等值线精度和节点本身计算量。

本文随后在 Berkeley 研究所采集的真实数据[13]上进行等值线查询，对比查询结果。

4.2 实验结果

图 9(a)为随机产生的传感器节点，其中空心节点为监测值相同的等值线节点；图 9(b)中的曲线是 DABC 算法输出的拟合等值线，分段线段为准确的等值线；图 9(c)中的曲线是 BSCTC 算法输出的拟合等值线，分段线段为准确等值线。另外两组模拟实验结果如图 10 和图 11 所示。

针对这 3 组模拟实验，对比 ISO-MAP[10]、DABC[12]和 BSCTC 算法返回的等值线节点数量，如图 12所示。可以看出，与 ISO-MAP 和 DABC 算法相比，BSCTC 算法减少了返回至 Sink 的等值线节点数量。与 DABC 相比，BSCTC 返回的节点数平均减少 53%。

利用公式

$$t=\frac{1}{n-1}\sum_{1}^{n-1}\int_{x_i}^{x_{i+1}}\frac{|f(x)-\hat{f}(x)|}{|x_i-x_{i+1}|}\mathrm{d}x \qquad (7)$$

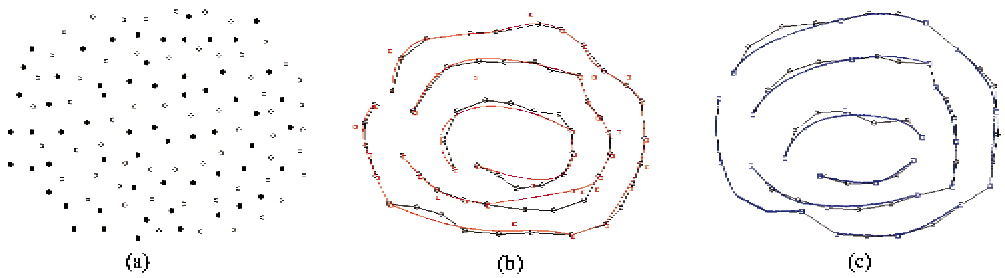


图 9 实验 1 网络中等值线节点个数为 60

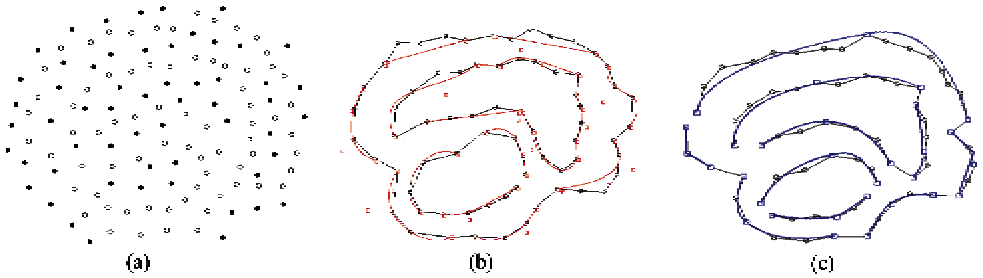


图 10　实验 2 网络中等值线节点个数为 72

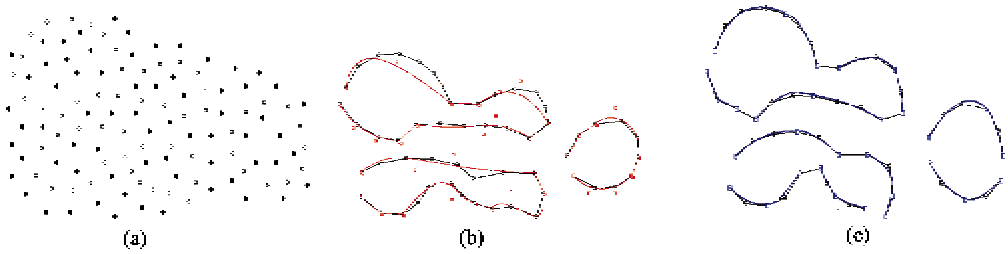


图 11　实验 3 网络中等值线节点个数为 56

可以计算不同算法生成的等值线的误差 t，其中 n 为等值线节点个数，x_i 为等值线节点 i 的 X 轴坐标，$f(x)$ 为等值线节点 i 与节点 $i+1$ 之间连线的直线方程，$\hat{f}(x)$ 为拟合等值线方程。

根据式(7)计算 BSCTC 算法和 DABC 算法在 3 组模拟实验下生成的等值线误差，如表 1 所示。可以看出，BSCTC 算法比 DABC 算法能更准确地还原等值线。

表 1　实验误差对比

	实验一误差	实验二误差	实验三误差
DABC	1.82	2.43	1.86
BSCTC	1.21	1.56	0.62
误差减小率(%)	33.5%	35.8%	66.6%

图 12 给出了不同算法返回的节点数。通过图 12和表 1 可得出如下结论:BSCTC 算法在节点传输量和等值线拟合效果两方面均优于 DABC 算法。

根据这 3 组模拟实验，下面分析阈值 θ 的最佳取值范围。图13(a)显示了不同θ取值下产生的等

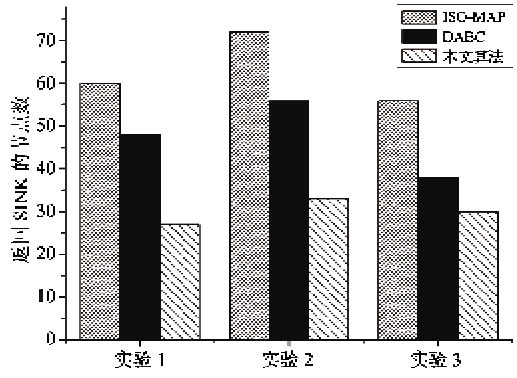
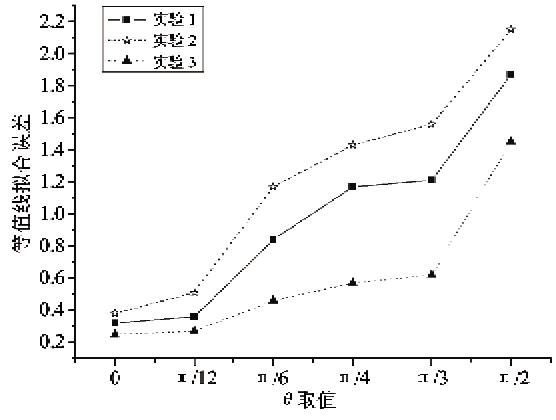
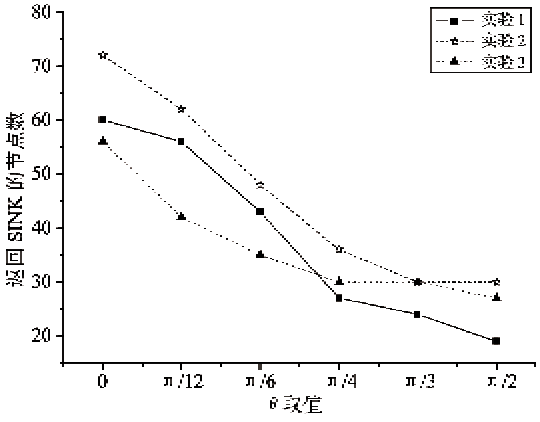


图 12　不同算法返回的节点数



(a) 对拟合误差的影响



(b) 对返回节点数的影响

图 13　BSCTC 算法的取值产生的影响

值线误差。图 13(b)显示了不同 θ 取值下返回的节点数。可以看出，θ 在[$\pi/4$,$\pi/3$]范围内，三组实验的拟合误差较小且稳定，而且可以返回较少的等值线节点数。

通过大量实验证实，当等值线节点分布趋势较平稳时，θ 在[$\pi/4$,$\pi/3$]内取值可保持误差较小且返回节点数较少；当等值线节点分布趋势不平稳，拐点较多时，应适当降低 θ 取值，保证拟合的精确度。

假定网络中存在 n 个等值线节点。用 BSCTC 算法时，因为每个等值线节点的计算复杂度为 $O(1)$，所以整个 BSCTC 算法的计算复杂度为 $O(n)$。表 2 给出了 BSCTC，DABC，ISO-MAP 三种算法的计算复杂度。可以看出，与 DABC 算法相比，BSCTC 算法的节点计算量也大大降低了。

表 2 网络中所有节点计算复杂度对比

算法	DABC	ISO-MAP	BSCTC
计算复杂度	$O(n^2)$	$O(n)$	$O(n)$

下面我们将 BSCTC 算法和 DABC 算法应用到伯克利研究中心的真实实验数据上。该实验数据是 54 个节点在室外一个月内所采集的关于光照、温度、湿度、电压的数据。我们提取这 54 个节点在某个时间采集的一组温度数据，然后分别用 BSCTC 算法和 DABC 算法对这些数据进行处理，产生这一时刻的温度等值线。图 14 为 BSCTC 算法的实验结果。图 15 为 DABC 算法的实验结果。

BSCTC 算法返回 36 个节点信息到 Sink，拟合等值线的平均误差为 2.10。DABC 算法返回 40 个节点信息到 Sink，拟合等值线的平均误差为 3.56。由此可见，BSCTC 算法明显优于 DABC 算法。

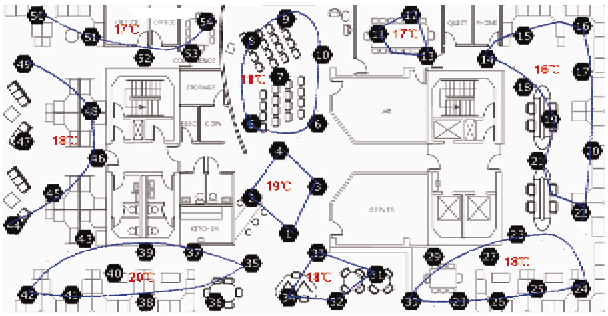


图 14 BSCTC 算法真实效果图

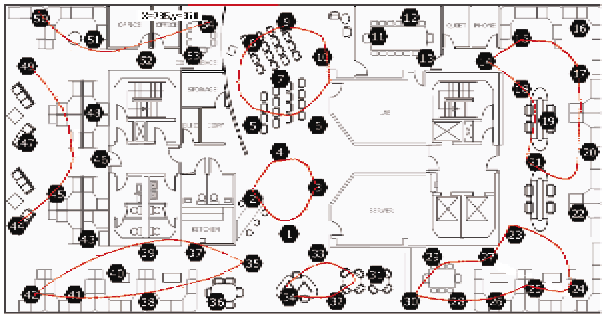


图 15 DABC 真实效果图

5 结 论

本文提出了一个新的传感网等值线查询的算法 BSCTC。与目前最好的算法 DABC 相比，BSCTC 算法减少了上传至 Sink 的节点数，且拟合出更加准确的等值线。同时，BSCTC 算法选取代表节点的计算复杂度减小到 $O(n)$。理论分析和实验结果均表明 BSCTC 算法是一种能量有效的精确的等值线查询算法。

参考文献

[1] 李建中，高宏. 无线传感器网络的研究进展. 计算机研究与发展，2008，45(1)：1-15

[2] Gedik B, Liu L, Yu P S. ASAP: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 2007, 18(12): 1766-1783

[3] He T, Stankovic J A, Marley M, et al. Feedback control-based dynamic resource management in distributed real-time systems. *IEEE Real-Time Systems*, 2007, 80(7): 997-1004

[4] Hellerstein J M, Hong W, Madden S, et al. Beyond average: Toward sophisticated sensing with queries. In: Proceeding of the Information Processing in Sensor Networks, California, USA, 2003. 63-79

[5] Li M, Liu Y, L. Chen. Non-threshold based event detection for 3D environment monitoring in sensor networks. *IEEE Transaction Knowledge and Data Engineering*, 2008, 20(12), 1699-1711

[6] Mainwaring A, Polastre J, Szewczyk R, et al. Wireless sensor networks for habitat monitoring. In: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks & Applications, Atlanta, USA, 2002. 88-97

[7] Xue W, Luo Q, Chen L, Liu Y. Contour map matching for event detection in sensor networks. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Illinois, USA, 2006. 145-156

[8] Liu Y, Li M. Iso-Map: Energy-efficient contour mapping in wireless sensor networks. In: Proceeding of the IEEE International Conference on Distributed Computing Systems, Washington, DC, USA, 2007. 36-42

[9] Meng X, Nandagopal T, Li L, et al. Contour maps: Monitoring and diagnosis in sensor networks. *IEEE Computer Networks*, 2006, 50(15): 2820-2838

[10] Li M, Liu Y. Iso-map: energy-efficient contour mapping in wireless sensor networks. *IEEE Transaction Knowledge and Data Engineering*, 2010, 22(5): 699-710

[11] 潘日晶. 满足数据点切向约束的二次 B 样条插值曲线. 计算机学报, 2007, 30(12): 2132-2141

[12] Yuan A, LIU Y, Ma C, et al. Energy-efficient contour mapping aggregation in wireless sensor networks. In: Proceedings of the IEEE Sensors Applications Symposium, Limerick, Ireland, 2010. 79-83

[13] http://db.csail.mit.edu/labdata/labdata.html

BSCTC: a contour query algorithm based on B-spline curves with tangent constraints for wireless sensor networks

Guo Longjiang[*,**], Sun Yihui[*,**], Liu Yong[*,**], Duan Jinsheng[*,**]

([*]School of Computer Science and Technology, Heilongjiang University, Harbin 150001)

([**]Key Laboratory of Database and Parallel Computing of Heilongjiang Province, Harbin 150001)

Abstract

Considering that existing contour query approaches for wireless sensor networks suffer from heavy transmission due to the large number of representative nodes, large computational overheads and low recovery accuracy, this paper proposes a new contour query algorithm based on B-spline curves with tangent constraints, named the BSCTC algorithm for short. The BSCTS algorithm firstly elects some representative nodes in contours and sends relative information of them to the Sink according to the interpolation principle for secondary B-spline curves with tangent constraint, and then the Sink recovers contours according to the information of representative nodes. The theoretical analysis shows that the expectation of the representative nodes returned by the BSCTC algorithm is only 39% of the number of the nodes in contours of sensor networks, and the complexity of representative node selection is $O(n)$. The experimental results show that when using the BSCTC algorithm, the number of the returned representative nodes can be reduced by 53% compared with the state-of-the-art DABC algorithm, and the more accurate contour map can be formed.

Key words: wireless sensors networks, contour monitoring, tangent constraints, B-spline, interpolation