基于正方形区域的WSAN节点定位算法的研究

徐焕良 李 多 任守纲 方贺贺 王浩云
(南京农业大学信息科技学院 南京 210095)

摘 要 在WSAN中,执行器节点可根据传感器节点感知的数据做出决策并执行相关操作,因此其对感知到事件的传感器节点的准确定位对于实施精确的控制策略至关重要。区别于现有WSN中的非测距定位方式,利用移动的执行器节点代替WSN中的锚节点,提出了一种基于正方形区域的WSAN节点定位算法。首先通过执行器节点的移动确定待定位传感器节点所在区域,然后通过迭代不断缩小该区域,当满足定位精度要求时计算区域的质心作为待定位节点的坐标。仿真实验证明,算法能够在存在RSSI误差和GPS误差干扰的情况下取得较好的定位精度,且使用少量的执行器节点完成定位不仅能节省网络部署成本,还可以克服传统WSN中非测距定位算法严重依赖锚节点密度的不足。

关键词 无线传感执行网络,无线传感器网络,移动执行器,非测距定位,区域划分
中图法分类号 TP393 **文献标识码** A

Research on Node Localization Algrithom Based on Square Area in Wireless Sensor and Actor Network

XU Huan-liang LI Duo REN Shou-gang FANG He-he WANG Hao-yun
(College of Information Science and Technology, Nanjing Agricultural University, Nanjing 210095, China)

Abstract In WSAN, actors make decisions and execute appropriate actions according to the data gathered from sensors. So accurate positioning of sensors that perceive the events is very important to implementing exact control policy. Different from those range-free algrithoms in WSN, a mobile location algrithom based on square area was proposed in WSAN. It used mobile actors instead of anchors to locate sensors. First, the area of the unknown node was determined through the movement of actors, secondly, this area decreased through iteration, when precision was satisfied, the centroid of this area was calculated and treated as coordinate of the unknown node. Via simulation experiments, the algrithom is proven to have a good locating accuracy when RSSI error and GPS error exist, and localizing unknown nodes with a few actors can not only save the cost of network deployment, but also can overcome shortcomings of range-free location algrithoms in WSN, which seriously depend on density of anchors.

Keywords WSAN, WSN, Mobile actor, Range-free position, Region division

1 引言

无线传感器与执行器网络(Wireless Sensor and Actor Network, WSAN)是无线传感器网络(WSN)的衍生物。它由大量传感器节点和一些执行器节点组成,其中传感器节点是微型的,其携带资源和处理能力均有限,而执行器节点能量充足且处理能力较强。通常,WSAN中的节点以人工部署的方式随机抛撒在监测区域中,传感器节点负责对整个区域进行监测,当有事件发生时,传感器节点将事件报告给执行器节点以便其采取相关措施处理该事件。因此在WSAN的许多应用中,传感器节点的位置信息是至关重要的,因为不知道传感器节点位置而感知的数据是没有意义的[1]。例如,在森林防火中,传感器节点感应到火苗,立刻通知本地的执行器(喷水器)进行喷水灭火,防止星火进入不可控制的森林大火状态[2]。又如,矿工在井下作业时随身携带温度和气体传感器来监测周围环境,当事故或灾难发生时,救援队可以根据传感器报告的位置信息进行更有效的人员搜救。此外,网络的管理和运行也需要传感器节点位置信息的辅助,例如对外部目标的跟踪、基于地理位置的路由、统计网络覆盖情况及控制网络负载均衡等。因此,定位技术对于WSAN来说至关重要。

目前全球定位系统(Global Positioning System, GPS)广泛应用于各个领域,但WSAN一般不使用GPS获取所有节

到稿日期:2012-12-27 返修日期:2013-05-30 本文受国家科技部重大科技支撑计划(2011BAK21B05),国家农业科技成果转化资金项目(2011GB2C100001),江苏省科技支撑计划项目(BE2011398),江苏省科技支撑计划项目(BE2011339),南京农业大学青年科技创新基金项目(KJ2010022),中央高校基本科研业务费项目(Y0201100080),南京农业大学2013年国家大学生创新创业训练计划项目(201310307049)资助。
徐焕良(1963—),男,博士,教授,主要研究方向为物联网关键技术;李 多(1989—),女,硕士生,主要研究方向为物联网关键技术,E-mail:2011114004@njau.edu.cn;任守纲(1977—),男,博士,副教授,主要研究方向为农业物联网应用技术;方贺贺(1994—),女,主要研究方向为CPS建模方法;王浩云(1981—),男,博士,讲师,主要研究方向为P2P网络、物联网、CPS相关协议和机制优化设计等,E-mail:wanghy@njau.edu.cn(通信作者)。

点的位置，主要原因有：1）传感器节点携带能量有限，体积较小，不适合配备 GPS 接收机；2）传感器节点数量众多，而网络成本有限，如果为每一个节点配备 GPS 接收机显然是不现实的。因此 WSAN 定位技术不能建立在大规模应用 GPS 的基础上。

WSN 定位算法主要有以下几种分类方式：基于测距的定位与非测距的定位、分布式定位与集中式定位、绝对定位与相对定位、基于锚节点的定位与无锚节点的定位。现有的 WSN 定位算法并不能完全适用于 WSAN，不能直接移植过来。主要有以下原因：1）WSAN 能够作用于外部世界的特点使得其对定位的精度要求较高。在 WSAN 中，执行器节点需要知道传感器节点的准确位置以便到特定的地点执行任务；2）WSAN 网络由传感器节点与执行器节点构成，这与传统的 WSN 网络不同。网络的异构性注定了 WSAN 定位算法与 WSN 定位算法间的差异；3）WSAN 中执行器节点能量充足、信号发射功率较大、位置已知并且数量通常不多。如何利用这些功能强大的执行器节点帮助传感器节点定位值得思考。故设计出适合 WSAN 的节点定位算法是目前研究的关键。

本文提出的基于正方形区域的移动 WSAN 定位算法（MLAS，Moblie Location Algrithom Based on Square Area）是一种基于区域划分的定位算法，原理为首先确定传感器节点所在的区域（执行器节点协作地移动形成一个正方形），然后通过迭代不断缩小传感器节点的区域，当满足定位精度要求时计算该区域的质心作为传感器节点的坐标。该定位算法可适用于较高精度要求的 WSAN 应用，当执行器节点获得待定位节点的精确位置后，可执行有效的控制策略。同时，算法采用非测距方式定位，对硬件要求较低。针对 WSAN 中执行器节点资源充足、位置已知的特点，使用少量移动的执行器节点来代替 WSN 中的锚节点进行定位，降低网络的部署成本。

2 相关工作

在 WSN 中，通常利用锚节点（通过配备 GPS 或人工部署的方式获得自身位置的节点）来辅助未知节点进行定位。目前最普遍的算法分类方式是根据是否测量距离将算法分为基于测距的定位算法和非测距的定位算法。基于测距的定位算法通过测量节点间的实际距离或角度信息来定位，定位过程一般分为两个阶段：1）获得锚节点与未知节点间的距离或角度；2）利用所测距离或角度通过特定算法（如三边测量法、三角测量法等）计算出未知节点的坐标。基于接收信号强度的方法[3]（Received Signal Strength Indication，RSSI）、基于信号传输时间的方法[3]（Time of Arrival，TOA）、信号到达时间差[3]的方法（Time Difference of Arrival，TDOA）等都是典型的基于测距的算法，它们大多对硬件要求较高，需要节点安装特殊的天线，不适用于大规模网络，并且信号传播距离受限及非视距传播等因素会影响定位的结果。除此之外，基于测距的定位机制使用各种方法来减小测距误差对定位结果的影响，包括多次测量、循环求精定位等方法，这些都要产生大量的计算和通信开销[4]。典型的非测距算法[3-5]有质心法、近似三角形内点测试法（Approximate Point in Triangulation test，APIT）、凸规划算法及基于跳数的距离矢量算法（Distance Vector Hop，DV-Hop）等，这类算法对硬件无特殊要求，定位成本较低，定位精度相对较低，性能依赖于网络拓扑[6]。

WSAN 相比于 WSN 功能更强大，因此越来越多的研究者将目光投向 WSAN 的定位。目前，对 WSAN 定位算法的研究已经取得了一些进展。文献[7]提出了一种基于事件驱动的高效协作定位算法，该算法只对检测到的事件发生的传感器节点进行定位，并且充分利用执行器节点能量不受限的特点来协助定位，有效降低了系统能耗。然而采用 RSSI 测距易受环境影响，从而使算法的实用性降低。文献[8]将 RSSI 方法与粒子滤波相结合对传感器节点进行定位，其与传统 WSN 的定位算法并无多大差别。文献[9]提出了一种半分布式的异步定位机制，算法通过离散双极坐标将每个执行器节点周围的传感器节点组织成一个区域，一部分传感器节点通过接收本区域的执行器节点的信标来计算自己的位置，该过程称为“训练”，这些完成定位的节点进一步训练它们的邻居节点，其中每个传感器节点周期性地唤醒，以节省能量。算法属于粗定位方法，通过升级锚节点来逐步完成全部节点的定位，容易造成误差积累，故定位精度难以保证。文献[10]提出了一种为移动传感器节点进行定位的解决方案，算法利用节点的权值与一跳距离相乘得到距离，然后计算坐标，并且随着节点不断的移动，动态调整权值以适应网络拓扑结构的变化。该算法可扩展性强，适用于大规模网络的定位。其缺点是初始权值的选取对精度影响较大，随着权值的上调，定位精度会有所下降。文献[11]提出的基于时间的定位算法适用于传感器与执行器节点均移动的动态定位，原理类似 TOA 算法。该算法最大的特色是不需要时间同步，通过建立一种时间模型来计算信标的传输时间，继而获得节点间的距离。另外该算法通过控制信标发送频率及传感器节点周期性接收消息的方法控制通信，从而降低了系统能耗。其缺点是时间间隔的划分过于详细，增加了算法的计算复杂度。文献[12]针对动态 WSAN 提出了一种基于 RSSI 测距的分布式事件驱动定位算法，即在 RSSI 测距模型中引入功率衰减因子 a 和高斯随机变量 v，提高了测距结果的精度，但 RSSI 信号始终存在较大误差，故该算法的实用性难以保证。文献[13]对基于半定规划的 WSAN 定位技术进行了研究，分别在测距与非测距方式下提出了新的基于半定规划的 WSAN 定位算法，仿真实验证明，该算法在定位精度和计算复杂度上均取得了不错的效果。

本文提出的 MLAS 算法使用少量移动的执行器节点代替锚节点，克服了 WSN 非测距算法定位精度严重依赖锚节点密度的不足，例如避免了 APIT 算法中的误判现象，在计算及通信方面的开销也优于 DV-Hop 算法。对于文献[7,8,12]来说，只是在 RSSI 算法的基础上进行改进或结合其它算法，而 MLAS 算法基于区域迭代求精的思想在定位精度上明显优于它们。文献[10]原理类似 DV-Hop，其执行器节点静止的模型使应用场合受到限制。基于以上原因，本文在非测距方式下设计了 MLAS 算法。

利用执行器节点上的 GPS 数据来直接获得待定位节点位置的方法并不适合对精度有较高要求的 WSAN 应用[14]。一般当执行器节点要靠近待定位节点时需要利用 RSSI 读数来判断距离，而 RSSI 易受干扰，且误差较大，再加上 GPS 自身的误差，这样靠单执行器节点的 GPS 数据来拟合待定位节点的位置会存在定位误差较大、结果不稳定等问题。本文算法中利用 4 个辅助执行器协助 1 个主执行器进行精确定位，使得执行器节点能够在获得待定位节点的准确位置信息后，

进行精确的控制操作。

3 基于正方形区域的移动 WSAN 定位算法

3.1 网络模型

网络中随机部署若干传感器节点和执行器节点(算法至少要求 5 个执行器节点,本文将以 5 个为例对算法进行阐述)。传感器节点静止,负责对整个区域进行监测并将数据报告给执行器节点,执行器节点是移动的,根据接收到的传感器节点的数据执行相关任务;传感器节点的信号强度有限,通讯半径设为 R,执行器节点的信号强度则覆盖整个网络;另外执行器节点配备 GPS,可随时获得自身位置信息。

3.2 算法设计

3.2.1 形成正方形区域

待定位节点 S 发起定位请求,第一个接收到该请求的执行器节点 A_0 移动到节点 S 附近。具体过程如下:节点 A_0 在向节点 S 靠近的过程中有间隔地向节点 S 发射信号,节点 S 计算接收到的信号强度值,当信号强度达到一定阈值时(即两个节点之间的距离足够小,$RSSI(d_{as})\geqslant a$,a 代表某个较小阈值,且 a 满足 $a>RSSI(R)$),它向节点 A_0 发送一个反馈信号,节点 A_0 收到该信号后立刻停止移动。

这里我们选择 RSSI 信号强度值来进行测量,假设理想情况下 RSSI 信号强度随距离增大而减小,且信号强度值与距离可利用理论或经验模型进行转化,通常使用式(1)(对数-常态分布模型)来计算节点间的距离[15]:

$$RSSI(d)=RSSI(d_0)-10\beta\log\left(\frac{d}{d_0}\right)+X_\sigma \tag{1}$$

式中,$RSSI(d)$为在离发射源距离为 d 处接收到的 RSSI 强度值,单位为 dBm;$RSSI(d_0)$为对应 d_0 米处待定位节点接收到的 RSSI 强度值,单位为 dBm;d 为发射端与接收端的距离;d_0 为参考距离,单位为 m,通常取为 1m;β 为路径衰减指数,与周围环境和障碍物密切相关,在理想传播模型下,$\beta=2$;X_σ 表示标准偏差为 σ 的正态随机变量,单位为 dBm,σ 在不同部署环境下取值为 4～12 不等[16]。本文实验中,统一取 $\beta=3$,$\sigma=6$。

节点 A_0 移动至某个位置停止,向其它执行器节点广播自己的位置信息(X_0,Y_0),其它执行器节点根据接收到的(X_0,Y_0)数据,分别调整自己的位置,最终形成一个正方形区域如图 1 所示,另外 4 个执行器节点预期位置的坐标为:

$$(X_1,Y_1)_{A_1}=(X_0-d_{as},Y_0+d_{as}) \tag{2}$$

$$(X_2,Y_2)_{A_2}=(X_0+d_{as},Y_0+d_{as}) \tag{3}$$

$$(X_3,Y_3)_{A_3}=(X_0+d_{as},Y_0-d_{as}) \tag{4}$$

$$(X_4,Y_4)_{A_4}=(X_0-d_{as},Y_0-d_{as}) \tag{5}$$

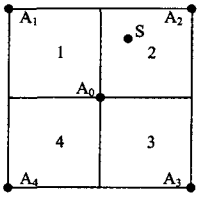

图 1 执行器节点位置关系

3.2.2 缩小所在区域

通过上述步骤,执行器节点形成了一个边长为 $2d_{as}$ 的正方形,能够保证节点 S 一定处于其中。接下来判断节点 S 处于 1、2、3、4 哪个子区域中,进一步缩小待定位节点所在的区域。判断方法为节点 S 接收来自 A_1,A_2,A_3,A_4 的发射信号,并记录下信号强度进行比较,按如下规则判断:

(1)RSS_1 最大,RSS_3 最小,节点 S 属于区域 1;

(2)RSS_2 最大,RSS_4 最小,节点 S 属于区域 2;

(3)RSS_3 最大,RSS_1 最小,节点 S 属于区域 3;

(4)RSS_4 最大,RSS_2 最小,节点 S 属于区域 4;

若节点 S 处于边界上,则理论上两侧区域的执行器节点的信号强度相同(同时最大或最小),这时判断节点 S 处于哪个区域对定位并无影响,随机处理即可。

节点 S 确定自己所在的子区域后,向节点 A_0 发送消息,例如节点 S 判断自己处于 2 号正方形区域中,随后向节点 A_0 发送消息,节点 A_0 收到消息后计算 2 号正方形中心的坐标,移动至该点,并向其它执行器节点广播自己的坐标,其它执行器节点收到消息后按照 3.2.1 节介绍的机制计算新的坐标并进一步调整自己的位置,如图 2 所示。重复上述步骤判断节点 S 所在的区域,通过迭代将待定位节点所在的区域缩小至很小的范围中,在每次迭代过程中,测量正方形顶点处的执行器节点间的信号强度,当其达到一定阈值时,停止迭代,并计算该正方形的质心,将其作为待定位节点 S 的坐标。

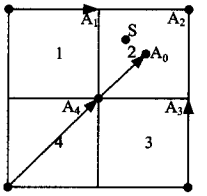

图 2 执行器节点移动形成新的正方形

坐标的计算公式如下:

$$(x,y)_{A_0}=\begin{cases}(X_0-1/4d_i,Y_0+1/4d_i) & S\text{ in area1}\\(X_0+1/4/d_i,Y_0+1/4d_i) & S\text{ in area2}\\(X_0+1/4d_i,Y_0-1/4d_i) & S\text{ in area3}\\(X_0-1/4d_i,Y_0-1/4d_i) & S\text{ in area4}\end{cases} \tag{6}$$

$$(x,y)_{A_1}=\begin{cases}(X_1,Y_1) & S\text{ in area1}\\(x_{A_0}-1/4d_i,y_{A_0}+1/4d_i) & S\text{ in area2}\\(x_{A_0}-1/4d_i,y_{A_0}+1/4d_i) & S\text{ in area3}\\(x_{A_0}-1/4d_i,y_{A_0}+1/4d_i) & S\text{ in area4}\end{cases} \tag{7}$$

$$(x,y)_{A_2}=\begin{cases}(x_{A_0}+1/4d_i,y_{A_0}+1/4d_i) & S\text{ in area1}\\(X_2,Y_2) & S\text{ in area2}\\(x_{A_0}+1/4d_i,y_{A_0}+1/4d_i) & S\text{ in area3}\\(x_{A_0}+1/4d_i,y_{A_0}+1/4d_i) & S\text{ in area4}\end{cases} \tag{8}$$

$$(x,y)_{A_3}=\begin{cases}(x_{A_0}+1/4d_i,y_{A_0}-1/4d_i) & S\text{ in area1}\\(x_{A_0}+1/4d_i,y_{A_0}-1/4d_i) & S\text{ in area2}\\(X_3,Y_3) & S\text{ in area3}\\(x_{A_0}+1/4d_i,y_{A_0}-1/4d_i) & S\text{ in area4}\end{cases} \tag{9}$$

$$(x,y)_{A_4}=\begin{cases}(x_{A_0}-1/4d_i,y_{A_0}-1/4d_i) & S\text{ in area1}\\(x_{A_0}-1/4d_i,y_{A_0}-1/4d_i) & S\text{ in area2}\\(x_{A_0}-1/4d_i,y_{A_0}-1/4d_i) & S\text{ in area3}\\(X_4,Y_4) & S\text{ in area4}\end{cases} \tag{10}$$

式中,$(x,y)_{A_i}$,$i=0,1,2,3,4$ 代表节点预期位置的坐标,(X_i,Y_i),$i=0,1,2,3,4$ 代表节点当前位置的坐标,d_i 代表当前形成正方形的边长,i 是当前正方形的迭代次数。

3.2.3 执行器节点移动机制

算法利用执行器节点的移动特性来辅助定位，本小节具体说明执行器节点的移动机制。

从上面的算法描述可以看出，执行器节点每次工作都要从当前位置移动到一个预期位置，设预期位置的坐标为(X_n, Y_n)，当前位置的坐标为(X_0, Y_0)，可以计算出斜率 $k_n=(Y_n-Y_0)/(X_n-X_0)$和距离 $D=\sqrt{(X_n-X_0)^2+(Y_n-Y_0)^2}$。

执行器节点只需按照斜率 k 移动距离 D 即可。然而在节点移动过程中，由于节点自身的原因及障碍物等环境因素的影响，移动可能出现偏差，导致节点最终停止的位置与预期位置有所差异。作者希望通过一些方法来降低这种误差。

其中一种方法是节点一次性移动距离 D 后启用校准机制：将节点停止移动后的当前位置的坐标与预期位置的坐标进行比较，如果超出误差允许范围，则进一步调整位置，即计算新的斜率及距离并移动，直到误差达到允许范围为止。这种方法简单易实现，缺点是如果斜率出现较大偏差会使得执行器节点移动代价过高。

因此笔者希望采用一种策略使得节点在移动过程中及时修正误差，选择一条接近最短距离的路径进行移动。大致思想是执行器节点每移动一段固定距离后就进行矫正，如果存在误差则调整斜率，否则按当前角度继续前进。最后当节点终止移动后，再启用校准机制进行位置校准。

下面对该过程进行详细说明：

首先，计算矫正次数 $Num=k\times[\sqrt{(Y_n-Y_0)^2+(X_n-X_0)^2}-b]$，其中 k,b 为常量，k 表示次数随着距离增长的幅度，b 表示需要执行矫正的最短距离，可根据实际情况自行设置；若 Num 为小数，则向下取整。其次，进行移动，令(X_i, Y_i)为执行器节点第 i 次移动后的位置坐标，k_i 为第 i 次移动后的斜率，S_i 表示第 i 次移动的距离，且移动距离 S_i 后需要进行一次矫正，$\Delta\theta$ 表示斜率 k_i 与 k_n 之间的角度差。计算移动距离 $S_i=\sqrt{(X_n-X_{i-1})^2+(Y_n-Y_{i-1})^2}/(Num-(i-1))$，$i=1,2,\cdots,Num$，并进行移动，移动一次过后计算当前斜率 $k_i=Y_i-Y_0/X_i-X_0$，并根据当前斜率计算角度差 $\Delta\theta=|\arctan k_i-\arctan k_n|$。

此时需要利用斜率来判断是否需要真正执行矫正，当 $\Delta\theta>\sigma$(σ 为角度阈值)时，我们就认为移动偏差较大，需要矫正，并执行矫正处理，否则按照当前斜率继续移动。

移动机制的伪代码如表 1 所列。

表 1 移动机制伪代码

```
Input:k_n,Num    Output:k_wheel
InitializeNum,k_n;
k_wheel←k_n;
for(i=0;i<Num;i++)
{
  S←√((X_n−X_i)²+(Y_n−Y_i)²)/(Num−i);
  Move(k_wheel,S); //按照斜率 k_wheel 移动距离 S
  k_{i+1}←Y_{i+1}−Y_0/X_{i+1}−X_0;
  Δθ←|arctank_{i+1}−arctank_n|;
  if(Δθ>σ)//角度偏差过大
  k_wheel←Y_n−Y_{i+1}/X_n−X_{i+1};//矫正:调整下次移动斜率
}
Start correction; //启用校准机制
```

校准机制的伪代码如表 2 所列。

表 2 校准机制伪代码

```
Input:r,(X_f,Y_f)    Output:(X_tmp,Y_tmp)
r←√((X_n−X_f)²+(Y_n−Y_f)²);
  //(X_f,Y_f)为节点终止移动后的坐标
k_wheel←Y_n−Y_f/X_n−X_f;
while(r>R) //(X_f,Y_f)超出预期坐标的误差范围
{
  Move(k_wheel,r);
  getX_tmp,Y_tmp;//(X_tmp,Y_tmp)为本次调整后的坐标
  r←√((X_n−X_tmp)²+(Y_n−Y_tmp)²);
  k_wheel←Y_n−Y_tmp/X_n−X_tmp;
}
Stop correction;
```

3.2.4 误差允许范围

如上所述，执行器节点通过移动形成一个正方形来辅助传感器节点进行定位，然而要形成一个规则的正方形有一定难度，3.2.3 小节中曾提到由于节点自身的原因及障碍物等环境因素的影响，移动过程中可能出现偏差，使得最终的坐标与预期坐标之间存在误差，从而影响定位的精度。因此需要启用校准机制对位置进行校准。

这里我们将误差允许范围设定为一个误差半径 R，如图 3 所示，当启用校准机制进行位置校准时，只要移动后的位置在预期位置的误差半径范围以内，就是定位可以容忍的误差范围。即 MLAS 算法允许形成一个近似正方形的四边形。

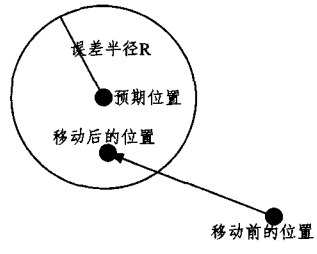

图 3 误差半径示意

4 算法仿真

本章采用 Java 语言编写仿真实验，对本文提出的 MLAS 算法的性能进行了验证。实验在 50m×50m 的网络中进行，部署 50 个传感器节点和若干执行器节点。传感器节点的通信半径设为 15m，执行器节点通信范围可覆盖整个网络，且移动速度为 10m/s。实验就信号强度阈值 a、误差半径 R 及执行器节点个数 Na 对 MLAS 算法性能的影响做了分析与总结。仿真数据为多次实验的平均值。

由于 WSN 中常见的定位算法，例如 APIT、质心算法等并不适用于 WSAN，且目前该领域没有与 MLAS 原理类似的定位算法，故仿真环节没有设计与其它定位算法的对比实验。

4.1 性能评价指标

在实验中，为了验证 MLAS 算法的各项性能，本节选择和制定了如下评价指标。

平均定位误差[17]：

$$\epsilon=\frac{\sum_{i=1}^{n}\sqrt{(x_i-xr_i)+(y_i-yr_i)}}{n}$$

式中，(xr_i, yr_i)表示第 i 个传感器节点的实际位置，(x_i, y_i)表示第 i 个传感器节点通过定位算法计算得到的估计位置，n 表示节点总数。

平均定位时间 $t=\frac{t\_total}{n}$，其中 t_total 表示所有节点定位所需的总时间，n 表示节点总数。一个节点定位所需的时

间主要包括节点移动时间、消息传输时间及节点处理时间。

平均定位开销 $o=\frac{o\_total}{n}$，o_total 表示所有节点定位的总开销，n 表示节点总数。这里使用发送消息个数来衡量定位开销。

4.2 仿真结果及分析

仿真结果为以上 3 个评价指标分别随信号强度阈值 a、误差半径 R 及执行器节点个数 Na 的变化。

4.2.1 信号强度阈值 a 对结果的影响

图 4—图 6 分别为在误差半径 R 及执行器节点个数 Na 恒定的情况下，平均定位误差、平均定位时间以及平均定位开销与信号强度阈值 a 的曲线关系，其中每幅图有条曲线分别表示 R 取 0.2m 到 1.0m 时的情况。平均定位误差单位为 m，平均定位时间单位为 s，平均定位开销单位为个。

从 3 幅图中可看到，在 R 及 Na 恒定的条件下，平均定位误差随着 a 取值的增大而减小；当 a 增大到 −68dBm 时，误差的减小幅度开始变小；当 a 取 −66dBm 时，平均定位误差可减小到 0.8m 左右。而平均定位时间及平均定位开销均随着 a 取值的增大而增大，这意味着在定位误差减小的同时会带来时间及开销方面的增长。

分析实验结果和曲线情况，信号强度阈值 a 越大，执行器节点 A_0 需要越靠近传感器节点才能停止移动，因此随着信号强度阈值 a 的增大，MLAS 算法的定位结果将更加精确。

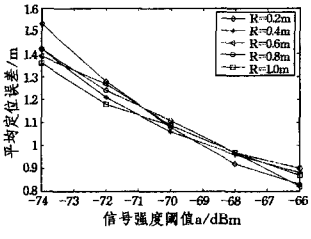

图 4 平均定位误差随信号强度阈值 a 变化曲线（$Na=5$）

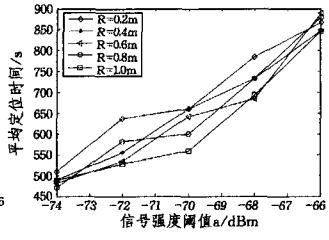

图 5 平均定位时间随信号强度阈值 a 变化曲线（$Na=5$）

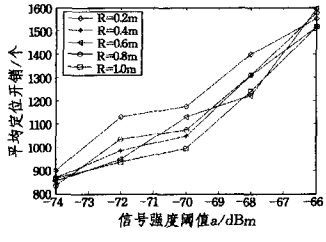

图 6 平均定位开销随信号强度阈值 a 变化曲线（$Na=5$）

4.2.2 误差半径 R 对结果的影响

图 7—图 9 分别为在信号强度阈值 a 及执行器结点个数 Na 恒定的情况下，平均定位误差、平均定位时间以及平均定位开销与误差半径 R 的曲线关系，其中每幅图有 4 条曲线分别表示 Na 取 5 到 8 时的情况。

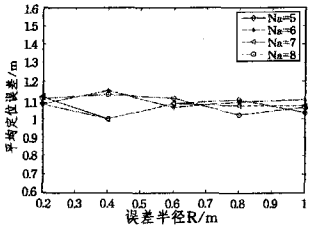

图 7 平均定位误差随误差半径 R 变化曲线（$a=-70$dBm）

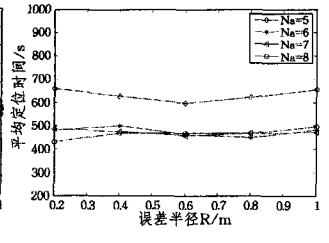

图 8 平均定位时间随误差半径 R 变化曲线（$a=-70$dBm）

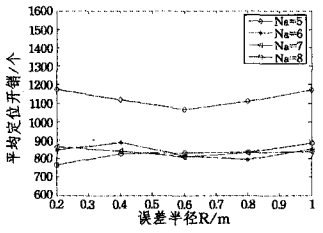

图 9 平均定位开销随误差半径 R 变化曲线（$a=-70$dBm）

从 3 幅图中可看到，在 a 及 Na 恒定的条件下，平均定位误差并未随着 R 取值的增大而呈现较大变化，只在 1m 处上下波动。R 取值的变化对平均定位时间及平均定位开销的影响也不大。

根据实验结果和曲线情况分析，误差半径 R 主要用于校准节点移动位置，使得执行器节点移动后的实际坐标与预期坐标间的误差在一个可以承受的范围内，实际当中对于最终传感器节点的定位结果产生的影响并不明显。影响虽不大，但误差半径 R 是保证传感器节点能够正确定位的前提，是算法的一个重要参数。

4.2.3 执行器节点个数 Na 对结果的影响

图 10—图 12 分别为在误差半径 R 及信号强度阈值 a 恒定的情况下平均定位误差、平均定位时间以及平均定位开销与误差半径 Na 的曲线关系，其中每幅图有 4 条曲线分别表示 a 取 −74dBm 到 −66dBm 时的情况。

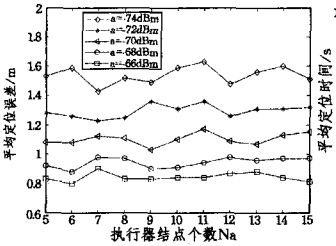

图 10 平均定位误差随执行器节点个数 Na 变化曲线（$R=0.2$m）

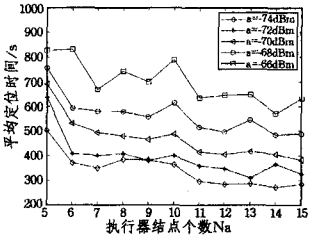

图 11 平均定位时间随执行器节点个数 Na 变化曲线（$R=0.2$m）

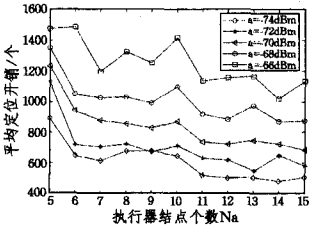

图 12 平均定位开销随执行器节点个数 Na 变化曲线（$R=0.2$m）

从 3 幅图中可看到，在 R 及 a 恒定的条件下，Na 取值的变化对平均定位误差未造成较大影响，曲线只是在较小范围内波动，无规律可言。而随着 Na 取值的增大，平均定位时间及平均定位开销的整体变化呈现下降趋势，特别是 Na 取值从 5 增加到 6 时，平均定位时间和平均定位开销开始大幅减少，之后取值趋于平稳，直到 $Na\geqslant 10$，平均定位时间和平均定位开销又出现小幅度下降，且曲线趋于平缓，只是偶尔出现小波动。

从实验结果和曲线图情况分析，执行器节点 Na 个数的增加使得距离传感器节点最近的一些执行器节点参与定位，在这样的情况下，定位时间和定位开销也将有所降低。当 Na 取 6 时最为理想。

综上所述，信号强度阈值 a 对 MLAS 算法性能影响最

大，a 取值越大，算法定位精度越高，而精度的提高也增加了定位时间及开销。执行器节点个数 Na 对定位精度影响不大；Na 取值越大，定位时间及定位开销越少，特别是取值大于等于 6 时这种变化更加明显。误差半径 R 对 MLAS 算法性能影响较小，R 值的变化对定位精度、时间和开销所引起的变化很小。通过仿真实验，给出最理想的参数设置为 $a=-68$dBm，$R=0.2$m，$Na=6$，此时平均定位误差仅为 0.88 米，平均定位时间不到 600 秒，而平均定位开销约为 1000 个消息。当然，在实际操作中，用户可根据具体应用环境及需求对 MLAS 算法的相关参数进行设置，对定位精度、定位时间及定位开销进行综合考虑，以期达到最佳效果。例如，对于精度要求高但实时性要求较低的环境，可以适当调高信号强度阈值 a，使得算法能够精确地定位到传感器节点。

4.2.4 讨论

通过移动的执行器节点辅助传感器节点进行定位，不但降低了部署成本，且充分利用了网络的异构性及有效资源。但节点移动过程中会产生误差并占用时间。MLAS 中节点 A_0 首先移动到待定位节点附近的某一位置，这个移动过程对算法以后的执行至关重要，因此引入信号强度阈值 a 对移动进行控制，实验证明，该参数对算法性能有很大影响。迭代过程中引入的斜率矫正及坐标校准机制，都是算法为进一步优化移动过程所做的努力。

MLAS 算法基于非测距的思想，与测距算法相比，降低了网络的能耗和开销；且原理简单，无需复杂的计算。经实验证明，MLAS 算法的平均定位误差较小，当 a 取 -66dBm 时，误差只有 0.8m 左右。缺点是平均定位时间较长，一般在 400s 以上。网络规模较小时，该算法能够取得不错的精度，特别是室内环境中。另外 MLAS 算法适用于实时性要求不高的场合。

结束语 WSAN 自身的特征及网络中节点的多样性使得 WSN 定位算法不能直接用于其中。本文提出的 MLAS 算法，使用移动的执行器节点来帮助传感器节点进行定位，充分利用了 WSAN 网络的异构性，具有以下优点：1）通过确定待定位节点所在的区域并不断减小它，最后求得该区域质心的方式来估算节点位置，不但原理简单，并且同测距算法需要添加额外的硬件设备相比，大大降低了成本；2）改进了 WSN 定位算法对锚节点数量要求较高的缺点，较大程度地节省了网络部署成本；3）经仿真实验证明，通过合理设置参数，算法取得了不错的定位效果，定位精度能够达到 0.88 米，且定位开销和定位时间均控制在一定范围内。接下来将 MLAS 算法与覆盖算法相结合，以期达到用最少的执行器节点覆盖最大的网络范围的效果。

此外，根据 WSAN 具体应用场景，执行器节点在实际环境中的移动会受到各种限制。这对执行器节点移动策略的设计提出了更高的要求，可利用机器学习等人工智能方法来进行路径规划、障碍规避、编队策略等方面的研究。我们将在本文工作的基础上对上述问题进行进一步的深入研究。

参 考 文 献

[1] 邱岩，赵冲冲，戴桂兰，等. 无线传感器网络定位技术研究[J]. 计算机科学，2008，35(5)：47-50

[2] 马建庆，钟亦平，张世永. ServLoc：无线传感反应网络的安全位置服务机制[J]. 软件学报，2008，19(10)：2628-2637

[3] 彭宇，王丹. 无线传感器网络定位技术综述[J]. 电子测量与仪器学报，2011，25(5)：389-396

[4] 王福豹，史龙，任丰原. 无线传感器网络中的自身定位系统和算法[J]. 软件学报，2005，16(5)：857-868

[5] 黄中林，邓平，梁甲金，等. 无线传感器网络定位技术研究进展[J]. 传感器与微系统，2009，28(11)：4-7

[6] 毛科技，赵小敏，何文秀，等. WSN 中基于区域划分的半自动 DV-Hop 定位算法[J]. 计算机科学，2012，39(3)：39-42

[7] Han P，Gao C S，Yang M，et al. ECLS：An Efficient Cooperative Localization Scheme For Wireless Sensor and Actor Networks [C]// Shanghai. Proceedings of the fifth International Conference on Computer and Information Technology. Los Alamitors：The Institute of Electrical and Electronics Engineers，2005：396-400

[8] El-Osery A I，Abd-Almageed W，Youssef M. Wireless corner Calibration-free RF-based localization algorithm for sensor actuator networks using particle filters[J]. IEEE Antennas and Propagation Magazine，2006，48(4)：166-173

[9] Giacomo G，Cristina M P，Sajal K D. A Semi-Distributed Localization Protocol for Wireless Sensor and Actor Networks[C]// Mannheim. 8th IEEE International Conference on Pervasive Computing and Communications. Los Alamitors：The Institute of Electrical and Electronics Engineers，2010：438-443

[10] Mustafa I A，Matthias R B，Damla T. Local Positioning for Environmental Monitoring in Wireless Sensor and Actor Networks [C]// Denver，CO. Proceedings of the 35th Annual IEEE Conference on Local Computer Networks. Los Alamitors：The Institute of Electrical and Electronics Engineers，2010：806-813

[11] Ghalib A S，Ozgur B A. Timing-Based Mobile Sensor Localization in Wireless Sensor and Actor Networks[J]. Mobile Networks and Applications，2010，15(5)：664-679

[12] 任守意. 动态无线传感器反应网络事件驱动定位算法[J]. 传感器与微系统，2009，28(12)：89-92

[13] 何国钢. 基于半定规划的 WSAN 定位技术研究[D]. 成都：西南交通大学，2012

[14] 张正勇，孙智，王刚，等. 基于移动锚节点的无线传感器网络节点定位[J]. 清华大学学报，2007，47(4)：534-537

[15] 朱登科. 基于 RSSI 的无线传感器网络测距和定位技术研究[D]. 长沙：国防科技大学，2010

[16] 李响. 无线传感网络节点自定位技术的研究[D]. 合肥：中国科学技术大学，2009

[17] 吕睿，阳宪惠. 减少无线传感器网络节点定位误差的方法[J]. 清华大学学报，2008，48(S2)：1839-1843