An Off-line Algorithm to Estimate Trajectories of Mobile Nodes Using Ad-hoc Communication

Sae Fujii Akira Uchiyama Takaaki Umedu Hirozumi Yamaguchi Teruo Higashino

Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871 Japan
{s-fujii,utiyama,umedu,h-yamagu,higashino}@ist.osaka-u.ac.jp

***Abstract*—In this paper, we propose an off-line algorithm called TRACKIE to estimate trajectories of mobile nodes based on encounter information. This method only assumes reasonable number of landmarks and ad-hoc wireless communication facility of mobile nodes, and does not rely on multi-hop ad-hoc networks nor global positioning systems. The method is new and practical because it achieves low-cost estimation of mobile trajectories and provides accurate solution (the average estimation error was less than 40% of wireless range in simulations). We have evaluated TRACKIE with MICAz Mote and shown that estimation error is about $2m$ in real environments where wireless range is $3m$.**

I. Introduction

Trajectory information has been recognized significant. For example, in wireless public LAN station deployment planning, high-performance stations may be placed along the flows of pedestrians in order to provide seamless connections. Also in evacuation planning for fire disaster or earthquake, we are able to make appropriate plans for emergency exit sign installation, announcement and so on. GPS receivers are useful for collecting trajectories, however they do not work indoors and underground like in large stations, airport terminals, convention halls and underground cities where we sometimes need to collect trajectories of mobile nodes for such purposes.

Positioning techniques using ad-hoc communication facility may be used in such situations. However, most of the methodologies assume multi-hop ad-hoc wireless networks which are sometimes partitioned and instable. Thus we should not rely on multi-hop networks; instead we should use more opportunistic communication style as studied more recently. In addition, we would like to determine trajectories of mobile nodes *more accurately* in an off-line manner, for mobility analysis purposes. Any existing on-line localization does not fit for this purpose. We discuss this issue in detail in Section II.

In this paper, we propose an offline algorithm called TRACKIE to estimate trajectories of mobile nodes using encounter information. Each encounter information is a record of two mobile nodes' encounter or a record of a mobile node and a landmark's encounter. For mobile nodes, we do not assume computation facility nor GPS receivers. Instead we only require short-range wireless communication devices and a small amount of memory which accumulates encounter information.

We have evaluated the proposed method using network simulator MobiREAL [1], [2]. From the experimental results, we have shown that our two-phase strategy of heuristic and SA could compensate for each other's disadvantage and consequently achieve in average the reasonable estimation error which was less than 40% of the wireless range with realistic mobility and geography. Also, we have evaluated TRACKIE with MICAz Mote and have shown that estimation error was about $2m$ where the wireless range was $3m$.

II. Related Work and Contribution

Most of existing positioning methods assume the knowledge about landmarks and the measured or estimated ranges between nodes. They are categorized into two types: (1) range-based methods that measure the ranges using ultra sound, RSS or some other technologies [3]–[5], and (2) range-free methods that only use wireless connectivity information [6]–[15]. The range-based methods may be able to achieve higher accuracy than range-free methods but require higher costs for measurement equipment. On the contrary, range-free methods are cost-efficient, but less accuracy is demerit. Centroid [7] and MCL [8] directly receive location information from landmarks and estimate positions. Amorphous [9], DV-HOP [10], APIT [12] and HCRL [11] receive location information in multi-hop manners from landmarks and estimate positions. In Sextant [13] and UPL [6], each node uses other nodes' estimated positions to estimate its own position. Some methods like MDS [14] and Sweeps [15] calculate relative positions based on connectivity information, and then decide absolute positions by landmark information.

Also localizing and tracking mobile nodes in sensor networks has been investigated so far [16]–[18]. These methods assume that sensors are deployed in high density and these sensors send the sensed events to base stations. Then the tracking of objects is done using the collected information by centralized, off-line computation.

Our method is different from the above methods since our method is low-cost and off-line, and is targeting mobile nodes. Object tracking in sensor networks also assumes centralized off-line computation, but the concept is very different because we target mobile nodes and use their encounter information. Also we do not need special hardware like ultra-sound transmitters and RSS indicators, nor multi-hop ad-hoc networks. Also, taking the merit of off-line, our method allows estimating trajectories more accurately using both spatial and temporal relationship compared with existing positioning algorithms that focus on spatial relationship only.

III. Problem Definition and Algorithm Overview

A. Target Environment

In this paper, we assume that landmark nodes (or simply landmarks) are deployed sparsely, and a mobile node corresponds to a pedestrian who has a portable terminal. All the portable terminals and landmarks are capable to communicate with each other via a personal area wireless communication devices such as ZigBee and Bluetooth. We assume that the communication range R is common for all the mobile nodes and landmarks, and their clocks are roughly synchronized.

Each mobile node or landmark has a unique ID, which includes a flag to distinguish its node type ("landmark" or "mobile node"). Each node i broadcasts "hello messages" at a regular interval to its neighbors. Every time node i receives a hello message from its neighbor node j, it stores the tuple (t, ID_i, ID_j) to its memory space where t denotes the time when node i received the hello message and ID_i (ID_j) is the ID of node i (node j). Here, we call this tuple "*encounter record*". When node i encounters a landmark, it sends all the collected encounter records to the server via the landmark.

Our objective is to estimate trajectories of N mobile nodes using the encounter records and position information of landmarks. Here, we assume that the timestamp t contained in an encounter record is an integer of $[0, T]$ without loss of generality (T is an integer constant). Therefore, we consider the discrete time domain $[0, T]$.

Each trajectory is a sequence of "*footprints*". Hereafter, the footprint of node i at time t is denoted as $p_{i,t}$, and the trajectory of node i is denoted by $p_{i,0}$, $p_{i,1}$,...,$p_{i,T}$. Our task is to determine the position of $p_{i,t}$ for any pair of $i \in N$ and $t \in [0..T]$ as accurate as possible.

B. Basic Idea of TRACKIE Algorithm

In general, deploying many landmarks increases accuracy, but it is often costly. Thus we do not assume many landmarks but assume that mobile nodes encounter landmarks at reasonable intervals. In such environments, the challenging problem is how we can accurately estimate the trajectory of each mobile node *between two landmarks*. An intuitive and simple solution is to give all encounter records to constrain the distance between nodes (*i.e.* the distance must not be greater than radio range R) and derive a solution using a linear programming problem solver (or some heuristic algorithms). However, since the number of constraints may become huge as the number of mobile nodes increases, it may not be realistic. Additionally, such a solution may have a fatal problem. Since there may be many feasible solutions that satisfy the constraints, it is not easy to find a solution which reproduces natural trajectories of pedestrians.

In general, all pedestrians do not walk straightly (assume they are in shopping malls). Some of them may stay for a while in front of their favorite shops, and some may sit on benches talking with each other. However, if the number of pedestrians becomes large, we can expect that at least one pedestrian walks straightly between landmarks. The most primary key point of our TRACKIE is to identify these straight trajectories and use them as "quasi-landmarks" because such trajectories can be estimated easily and with high accuracy.

Based on this idea, we design a three phase algorithm. The first phase called *initialization phase* estimates the trajectories by only using encounter records with landmarks to obtain an initial, simple solution. We assume that between two landmarks each mobile node moved straightly with a constant speed. Therefore, an estimated trajectory is a polygonal line where vertexes correspond to landmark positions (Fig. 1(a)).

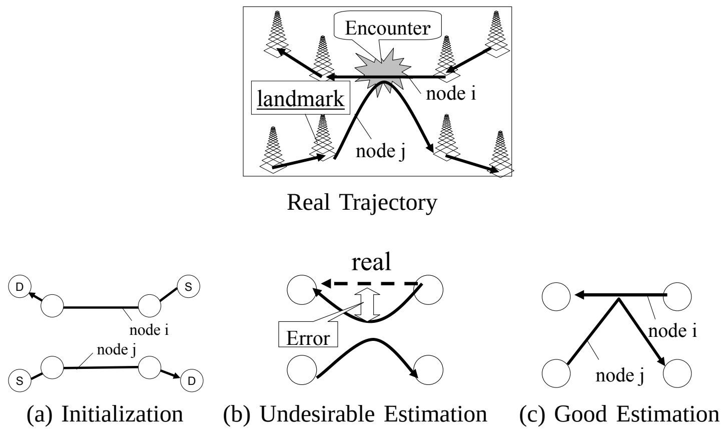


Fig. 1. Estimation of Trajectories by TRACKIE

The second phase is called *iterative modification phase* and uses encounter records between mobile nodes to modify the form of trajectories. For each encounter record that indicates encounter of two mobile nodes i and j at time t, the positions of their "footprints" at time t must be within the distance R, where R is the radio range. Here, to satisfy this constraint (called *encounter constraint* hereafter), it is one possibility to bend the straight trajectories of the two nodes to make the footprints close to each other. However, if one trajectory is actually a straight line, this clearly affects the estimation accuracy (Fig. 1(b)). Thus in this phase, we first identify such trajectories that are straight lines with high possibility, and use them to modify other trajectories. For example, in Fig. 1(c), we can see that the node i's trajectory is determined first, and node j's trajectory is stretched by the encounter constraint. Also, in order to avoid big modification of trajectories that may lead to unnatural trajectories, we iteratively localize the footprints so that trajectories are stretched gradually.

Finally, given the modified trajectories, we should check if they never traverse obstacles such as buildings and walls (called *obstacle constraints*) and if they never exceed the maximum speed of pedestrians (called *velocity constraints*). Then we should adjust the trajectories so as to satisfy these constraints as well as the encounter constraints. Since such adjustment needs to check all the constraints, we apply a global optimization technique based on Simulated Annealing (SA) in the third phase. This phase is called *SA-Based modification phase*.

IV. Algorithm Description

In the algorithm, for each footprint, a "state" is associated. The state of a footprint is either of *anchored*, *quasi-anchored*, *constrained* or *free*. Footprint $p_{i,t}$ is said to be *anchored iff* node i encountered a landmark at time t. These states are explained in the algorithm description. The three phases of the algorithm are explained below.

Iterative Modification Phase

```
initialize NQ as set of all the free footprints
do{
//determine footprints to quasi-anchored
  cand ← fastest_footprints(NQ)
  for (each $p_{i,t}$ ∈cand) {
    $p_{i,t}$.state ← "quasi-anchored"
    NQ ← NQ − {$p_{i,t}$}
  }
//localize non-quasi-anchored footprints
  for (each $p_{i,t}$ ∈ NQ)
    for (each $p_{j,t}$ ∈encounter($p_{i,t}$)){
      if($p_{j,t}$.state = "quasi-anchored")
        EQ ← EQ ∪ {$p_{j,t}$}
      $p_{i,t}$.pos = centroid($p_{i,t-1}$, $p_{i,t+1}$,EQ)
    }
  for (each constrained footprint $p_{i,t}$)
    modifyFreeFootprint($p_{i,t}$, nextConstrained($p_{i,t}$))
}while (NQ ≠ $\phi$)
```

Fig. 3. Algorithm Description (Iterative Modification Phase)

A. Initialization Phase

For each encounter of node i with a landmark at time t, the initialization phase makes footprint $p_{i,t}$ *anchored* and sets its position to the position of the landmark. For the other footprints, their states are set to *free*. Then for each pair of two subsequent *anchored* footprints $p_{i,t}$ and $p_{i,t+l}$ of node i ($l > 0$), we determine the positions of the *free* footprints $p_{i,t+1}$, ..., $p_{i,t+l-1}$ so that they are aligned along the line between $p_{i,t}$ and $p_{i,t+l}$ with equal spaces.

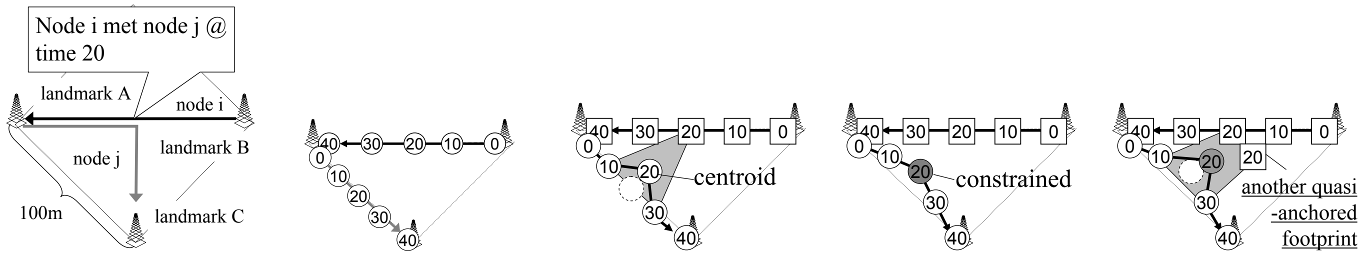
The set of footprints after this phase is called initial solution. In this initialization phase, the trajectories are regarded as straight lines between landmarks. Thus any encounter information between mobile nodes is ignored.

For readability, we use an example shown in Fig 2(a) in the following sections. Fig 2(b) shows the initial solution where $p_{i,0}$, $p_{i,40}$, $p_{j,0}$ and $p_{i,40}$ are anchored.

B. Iterative Modification Phase

Here we introduce a new state of footprints, *quasi-anchored*. Once a footprint becomes *quasi-anchored*, its position is fixed throughout this modification phase. Thus a *quasi-anchored* footprint is treated as *anchored* footprint and is used to estimate the other footprints' positions.

The iterative modification phase modifies the initial solution considering the encounter constraints. This is done by gradually increasing *quasi-anchored* footprints and iteratively localizing the other footprints using the encounter information with the *quasi-anchored* footprints. This terminates when all the footprints except



(a) real trajectories (b) initial trajectories (c) modification of initial trajectories

Fig. 2. Examplification of Iterative Modification Phase

the *anchored* footprints become *quasi-anchored*. In Fig. 3, we give the formal description of this iterative modification phase. The codes from 3 to 8 correspond to *Determining Candidate to be Quasi-anchored (DCQ)* policy that determines the ordering to make footprints *quasi-anchored*. The codes from 9 to 17 correspond to *Localizing Non-Quasi-anchored footprints (LNQ)* that localizes the positions of non-*quasi-anchored* footprints. These policies are explained below.

Here we address the DCQ policy. We let $\vec{v_{i,t}}$ denote node i's estimated velocity vector at time t in the initial solution, and it is defined as $|\overrightarrow{q_{i,t}q_{i,t+1}}|$ where $q_{i,t}$ denotes the position of footprint $p_{i,t}$ in the initial solution. After obtaining the initial solution, we compute the velocity vectors for all pairs of $i \in N$ and $t \in [0..T]$. Using these velocity vectors, we continue the following round until all the footprints (except the *anchored* ones) become *quasi-anchored*. First, we choose the sequence of footprints $p_{i,t+1}$,...,$p_{i,t+l-1}$ where (i) $p_{i,t}$ and $p_{i,t+l}$ are *anchored*, (ii) $p_{i,t+1}$,...,$p_{i,t+l-1}$ are not *quasi-anchored*, and (iii) the velocities of $p_{i,t+1}$,...,$p_{i,t+l-1}$ (they must be equal due to the property of the initial solutions) are not less than any of the others. Intuitively, this indicates to choose the fastest movement between two landmarks in the initial solution. Secondly, we let $p_{i,t+1}$,...,$p_{i,t+l-1}$ be *quasi-anchored* and localize the positions of the other footprints by the policy LNQ explained later in this section. We exemplify the above by an example. In Fig. 2(b), we choose the sequence $p_{i,10}$, $p_{i,20}$, $p_{i,30}$, set their states to be *quasi-anchored* and localize the other footprints. After that, we set the states of $p_{j,10}$, $p_{j,20}$, $p_{j,30}$ to be *quasi-anchored*, and at this moment the algorithm terminates.

Finally, we address the policy LNQ that localizes footprints. Here, we introduce an additional state, *constrained*, for this localization process. First, we identify each footprint $p_{i,t}$ that is either *free* or *constrained*, and has encounter constraint(s) with some quasi-anchored footprints. Secondly, we set the position of $p_{i,t}$ to the centroid of the positions of its successor $p_{i,t+1}$, predecessor $p_{i,t-1}$ and each quasi-anchored $p_{j,t}$ where node i and j encountered at time t. Thirdly, we set the state of $p_{i,t}$ to be *constrained* if it is *free*. After these steps, free footprints $p_{i,t-1}, ..., p_{i,t-k+1}$ and $p_{i,t+1}, ..., p_{i,t-k'-1}$ where $p_{i,t-k}$ and $p_{i,t+k'}$ are footprints that are not *free*, we set the positions of these free footprints so that they are aligned along the line between $p_{i,t-k}$ and $p_{i,t}$ and the line between $p_{i,t}$ and $p_{i,t+k'}$. Intuitively, the above process attempts to satisfy encounter constraints using the quasi-anchored footprints as new landmarks. By taking the centroid of positions of the predecessor and successor as well as the quasi-anchored footprints, we avoid big change of positions by one attempt of localization. By iteratively applying this localization process, the positions of footprints are gradually adjusted so as to satisfy the encounter constraints. The example of this localization is shown in Fig. 2(c). First, we set the position of $p_{j,20}$ to the centroid of $p_{j,10}$, $p_{j,30}$ and $p_{i,20}$ because $p_{i,20}$ has already been quasi-anchored and node i met node j at time t. Second, we set the state of $p_{j,20}$ to be *constrained*, then align $p_{j,10}$ and $p_{j,30}$ along the line between $p_{j,0}$ and $p_{j,20}$ and along the line between $p_{j,20}$ and $p_{j,40}$. Moreover if $p_{j,20}$ gets another quasi-anchored footprint (say $p_{a,20}$), we set the position of $p_{j,20}$ to the centroid of $p_{j,10}$, $p_{j,30}$, $p_{i,20}$ and $p_{a,20}$.

C. SA-based Modification Phase

Simulated Annealing (SA) techniques are used to find a global minimum without falling into a local minimum. They compare the current solution and a new candidate, and accept the new candidate with the following acceptance ratio $P(rand(0,1) < e^{-\Delta Cost/T})$. Here, $\Delta Cost$ denotes the cost of the new candidate minus the cost of the current solution. If $\Delta Cost \leq 0$, the new candidate is always accepted because $e^{-\Delta Cost/T}$ is more than 1. On the other hand, if $\Delta Cost > 0$, the new candidate is accepted with some probability. As $\Delta Cost$ is smaller, the probability to accept candidates becomes higher.

In the SA-based modification phase, we modify the trajectories modified by the iterative modification phase using the SA technique. To generate a new candidate, we first choose a footprint randomly, and move its position within a circle centered at the current position with radius r_{sa}. We increase/decrease r_{sa} based on the following formula so that the acceptance ratio p approaches to 0.5;

$$r_{sa} = \begin{cases} r_{sa} \times (1 + c_u \dfrac{p-0.6}{0.4}), & p > 0.6 \\ \dfrac{r_{sa}}{(1 + c_u \frac{0.4-p}{0.4})}, & p < 0.4 \\ r_{sa}, & otherwise \end{cases} \quad (1)$$

where c_u is a constant number and we set $c_u = 2$ empirically.

As the cost function, we consider the following three functions $Cost_a$, $Cost_b$ and $Cost_c$, and we execute SA optimization for each function. Therefore, the SA optimization is executed three times.

a) Minimizing Encounter Constraint Errors: If two nodes i and j encountered at time t, the distance between the two footprints $|p_{i,t} - p_{j,t}|$ must be less than or equal to wireless range R. It is natural to minimize the violation of encounter constraints for better solutions. Thus as the first cost function $Cost_a$, we define the total sum of “encounter constraint errors” for all the encounter records. Formally,

$$Cost_a = \sum_i \sum_j \sum_t e_{i,j,t} \quad (2)$$

$$e_{i,j,t} = \begin{cases} 0, & |p_{i,t} - p_{j,t}| \le R \\ |p_{i,t} - p_{j,t}| - R, & otherwise \end{cases} \quad (3)$$

If we can reduce the value of $Cost_a$, we can mitigate inconsistency with encounter records. This can make the derived trajectories more accurate.

b) Minimizing Obstacle Constraint Errors: If we can obtain obstacle information, we would like to derive trajectories which do not traverse obstacles. Therefore we define “obstacle constraint error” $o_{i,t}$ as the distance from the borderline of an obstacle to footprint $p_{i,t}$ if $p_{i,t}$ is located inside the obstacle. As the second cost function, we use $Cost_b = \sum_i \sum_t o_{i,t}$.

c) Minimizing Speed and Direction Fluctuation: We would like to derive trajectories such that nodes' speeds and moving directions do not change suddenly. We define $\overline{V_i}$ as node i's average speed, which is defined as $\overline{V_i} = \sum_{k=0}^{T-1} |\vec{v_{i,t}}|/T$. We also define $s_{i,t} = |\vec{v_{i,t}}| - \overline{V_i}$. Here, $\sum_t s_{i,t}$ denotes fluctuation of speeds of node i. We also define $a_{i,t}$ in the formula (4). Then, $\sum_t a_{i,t}$ denotes fluctuation of moving directions of node i. Therefore, we define $\sum_i \sum_t (s_{i,t} + a_{i,t})$ as the total speed and direction fluctuation, and treat it as the third cost function $Cost_c$. If we can reduce the value of $Cost_c$, we can obtain smooth trajectories with small speed fluctuation.

$$a_{i,t} = \frac{360}{2\pi} \times \arccos\left(\frac{\vec{v_{i,t+1}} \cdot \vec{v_{i,t}}}{|\vec{v_{i,t+1}}||\vec{v_{i,t}}|}\right) \quad (4)$$

Using those three cost functions, we apply SA techniques in order to obtain more accurate and natural trajectories.

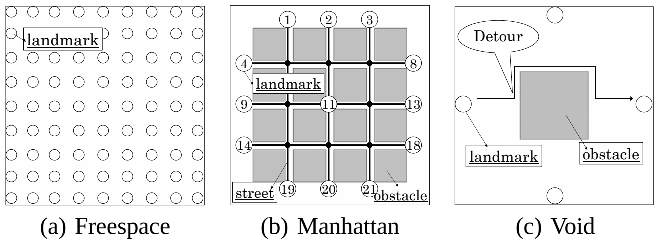


Fig. 4. Maps of Simulated Areas (circles represent landmarks)

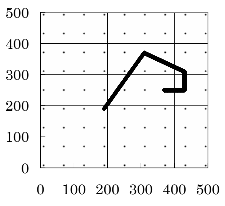
V. SIMULATION

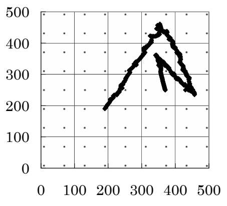
We have conducted simulations using the network simulator MobiREAL [1], [2] to see the accuracy of trajectories and to compare TRACKIE with the other methods.

A. Performance Evaluation of TRACKIE in Different Scenarios

To evaluate the accuracy of the estimated trajectories of different shapes in several environments, we have arranged three types of scenarios. The *zigzag scenario* assumes the Random WayPoint(RWP) model in the free space map of $500m \times 500m$ shown in Fig. 4(a) to see the reproducibility of zigzag trajectories. It is challenging to estimate trajectories that contain “stay” and “turn” observed in the RWP model. The *street walking scenario* assumes the Random Street Decision (RSD) model in the Manhattan map of $500m \times 500m$ where six streets of $20m$ width are deployed (Fig. 4(b)). In the RSD model, each node moves along a street, and at each intersection it randomly chooses a street (except the backward direction). This scenario is likely in city sections. Finally, the *detour scenario* assumes four trajectories that detour the $25m \times 25m$ square void in the center of the field of $100m \times 100m$ as shown in Fig. 4(c). Obviously this scenario is very likely and general in many situations.

We have used the LoS (Line of Sight) radio propagation model where only two nodes that are visible from each other can communicate. The radio range was set


(a)


(b)

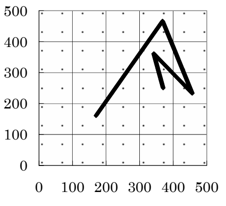

(c)

Fig. 5. Estimated Trajectories ((a) after initialization phase (b) after iterative modification phase) and (c) Real Trajectory (in Zigzag Scenario)

to $10m$. The velocities of nodes were set to follow the normal distribution with mean $2.0m/s$ and variance 0.1. The pair of the number of nodes and the number of landmarks was set to (3,000, 81) in the zigzag scenario, (1,000, 13) in the street walking scenario and (200, 4) in the detour scenario. A hello packet was sent from each node for every second.

In the above settings, we have evaluated two parameters, *Average Position Error* (APE in short) which is the average of position errors for all the nodes and times, and *Average Angle Error* (AAE in short) which is the average difference of the angles of two velocity vectors from the real and estimated trajectories. Formally, these are defined as follows;

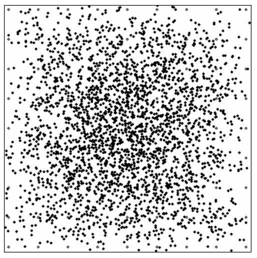
$$APE = \frac{\sum_i \sum_t |p_{i,t} - \hat{p_{i,t}}|}{T \cdot N}$$

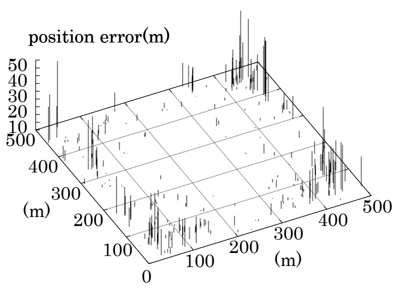
$$AAE = \frac{\sum_i \sum_t \arccos\left(\frac{avgv_{i,t} \cdot av\hat{g}v_{i,t}}{|avgv_{i,t}||av\hat{g}v_{i,t}|}\right)}{T \cdot N}$$

where $p_{i,t}$ and $\hat{p_{i,t}}$ are estimated and real positions of node i at time t, respectively. Similarly, we let $avgv_{i,t}$ and $av\hat{g}v_{i,t}$ denote the average velocity vectors from time t to time $t+l$ in the estimated and real trajectories respectively (l is a constant time duration). $avgv_{i,t} = \frac{\overrightarrow{p_{i,t}p_{i,t+l}}}{l}$ and in all the experiments we have assumed $l = 10$.

In Table I, we have shown APEs and AAEs. We have measured those values after the three phases of the algorithm. For comparison purpose, we have also measured those values without the iterative modification phase (case (iii')).

From the results in Table I, in the zigzag scenario, APE (in the whole region) was more than $5m$, which was rather larger than those in the other two scenarios. On the other hand, in the center region, APE was less than $4m$, which was close to those in the other scenarios. This is because node distributions observed in the RWP model are such that many nodes concentrate in the center of the region and they are hard to encounter the other nodes near the boundary. Therefore, the above results indicate that TRACKIE could accurately estimate the trajectories that are irregular (and thus complex), if nodes had enough chance to encounter each other. Also in both of the street walking and zigzag scenarios, the APE (or AAE) after the iterative modification phase and the SA-based modification phase are identical. From this fact, the iterative modification phase could archive enough accuracy by itself.


(a)



(b)

Fig. 6. Positioning Errors at a Moment ((a) node distribution (b) node positions vs. positioning errors) (in zigzag scenario)

TABLE II
SIMULATION PARAMETERS

# of Nodes	**3,000**, 4,000 or 5,000
Hello Packet interval (sec.)	**1**, 10 or 20
Wireless range(m)	**10**, 20 or 30
# of Landmarks	57, 69 or **81**
# of Mobile Landmarks	**0**, 100, 200 or 300

Then we focus on the detour scenario where the iterative modification phase is supposed to have larger errors because no node is able to move straightly between two landmarks in this scenario. As expected, APE was still $7.69m$ after the iterative modification phase, but after the SA-based modification phase it became $4.76m$. In addition, we can see considerable improvement of AAE by the SA-based modification phase. These results indicate the necessity of the SA-based modification phase to cope with various environments and situations.

An interesting observation is that without the iterative modification (*i.e.* case (iii')) accuracy could not be improved. Thus this indicates that both modification phases help each other to accomplish high accuracy.

B. Comparison with Existing Methods

We have compared the performance of TRACKIE with MCL [8] and Amorphous [9] under different parameter

TABLE I

AVERAGE POSITION ERRORS / AVERAGE ANGLE ERRORS (BOLD FONT INDICATES THE BEST THROUGHOUT SCENARIO)

	zigzag (m) / (rad)		street walking (m) / (rad)	detour (m) / (rad)
	Whole Region	Center Region		
(i) After initialization phase	36.75 / 0.48	30.33 / 0.40	41.96 / 0.61	11.34 / 0.49
(ii) After iterative modification phase	**5.42** / **0.22**	**3.97** / **0.17**	3.77 / 0.13	7.69 / 0.45
(iii) After SA-based modification phase	**5.42** / **0.22**	**3.97** / **0.17**	**3.71** / **0.12**	**4.76** / **0.38**
(iii') After SA-based modification phase (iterative modification phase was not applied)	26.93 / 0.42	19.28 / 0.32	19.29 / 0.42	6.52 / 0.48

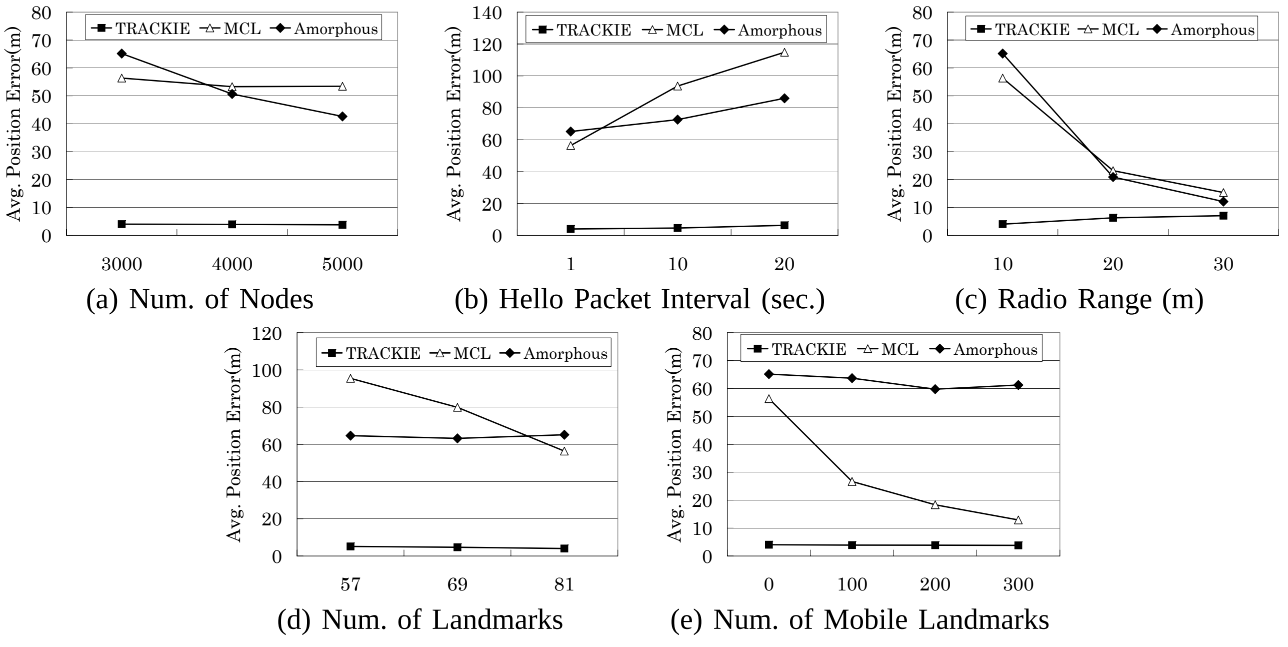


Fig. 7. Average Position Error (APE) of TRACKIE, MCL and Amorphous under Different Parameter Settings

settings. Of course, this may not be fair comparison because MCL and Amorphous are on-line (real-time), distributed localization techniques. However, we would like to know to what extent our centralized off-line algorithm works better than those well-known on-line methods.

MCL stands for the Monte Carlo Localization and is a range-free localization where each node manages its area of presence and refines it whenever it encounters a landmark. Amorphous is also a range-free localization where position information of landmarks are propagated through ad-hoc networks composed by mobile nodes and each node estimates its position by using the number of hops from these landmarks as well as the received position information of landmarks.

We have set the ranges of parameters as shown in Table II. In each experiment, we have varied one of these parameter values and let the others remain the default values emphasized by a bold font. The zigzag scenario introduced in Section V-A was used, and we let the velocities follow the normal distribution with mean 2.0m/s and variance 0.1. We have measured the average position error (APE) in the center region.

a) Number of Nodes: Fig. 7(a) shows APE with different number of nodes. In all the three methods, errors were identical or improved as the number of nodes became large. In particular, Amorphous could accomplish considerable improvement by the increase of nodes because it could have more chances to get landmark position information by well-connected ad-hoc networks. We note that the average number of neighbors were 5.16, 6.89 and 8.62 in the cases of 3,000, 4,000 and 5,000 nodes, respectively.

b) Hello Packet Interval: Since all the methods rely on hello packets among nodes, larger transmission intervals may result in less information about neighbors. This property is clearly shown in Fig. 7(b). However, the important thing is that in TRACKIE, if a node receives a hello packet, we can constrain the distance between its sender node and receiver node in the algorithm. Therefore, this information helps to increase the accuracy of the sender node's position as well as that of the receiver node. On the other hand, in the other two methods, only the receiver can take this benefit because the sender node does not know the receiver node's presence. As a result, TRACKIE is robust to longer intervals.

c) Radio Range: From the result in Fig. 7(c), the errors in MCL and Amorphous became smaller as the radio range became longer. This is natural because larger radio range leads to more changes to encounter landmarks in MCL and the larger number of neighbor nodes in Amorphous.

On the other hand, in TRACKIE, the error slightly increased as the radio range became larger. Of course we obtain more encounter information for larger radio range, but distant constraints between two encounter nodes are relaxed. The result shows that the effect of the latter dominates the former. However, even in the case of large radio range R, the error was less than $0.4R$, which is still reasonable enough.

d) Number of Landmarks: Fig. 7(d) shows the effect of the number of landmarks. We can see that MCL was affected much more than the others because in MCL nodes need to encounter landmarks to localize themselves. In Amorphous, deploying more landmarks does not directly improve the errors because it needs stable ad-hoc networks that propagate landmark information. It is natural that the errors in TRACKIE were improved with the larger number of landmarks, but even with fewer landmarks (*e.g.* 57 landmarks), it could achieve enough accuracy.

e) Number of Mobile Landmarks: Sometimes, landmarks are not stationary because some mobile nodes can track its exact positions by GPS devices and provide the information to neighboring nodes [8]. Here we assume such a situation where these mobile landmarks are also available in addition to the stationary landmarks. The result in Fig. 7(e) shows that MCL could improve the position errors by these mobile landmarks, and Amorphous could not improve the errors because of the same reason as the case of Fig. 7(d). Since in this experiment we deployed 81 stationary landmarks, additional mobile landmarks did not help to improve the errors in TRACKIE any longer. Thus we can say that we do not need many landmarks, but need some landmarks (2.7% of mobile nodes in this case) and less number of mobile nodes than that required by existing methods like Amorphous.

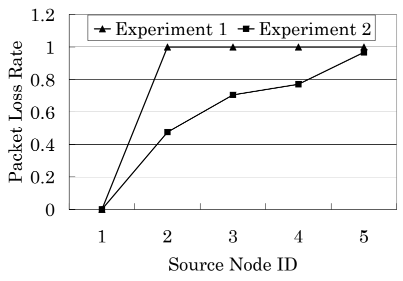
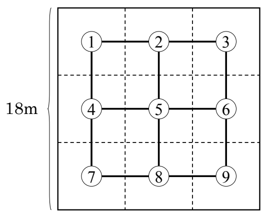
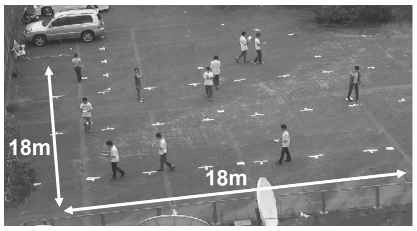


Fig. 8. Packet Loss Rate





(a) Experiment Map (b) Scene of Experiment

Fig. 9. Experiment Environment

VI. Experiment Using MICAz Mote

We conducted experiments to evaluate TRACKIE using MICAz Mote. As preliminaries, we have measured radio range of Motes in the open air in the following two environments.

- Experiment1: six nodes ($ID_0..ID_5$) were deployed on the ground along a straight line at one meter interval in this order.
- Experiment2: same as experiment1 except that ID_0 was at one meter height from the ground

We set Mote's radio power to -15dBm and the interval of hello packet transmission to 1 sec. Then we let the nodes ID_i ($i = 1..5$) transmit hello packets and have measured the number n of received hello packets at ID_0.

Fig. 8 shows packet loss rates defined as $(T - n)/T$. We can see that in Experiment 1, node ID_0 has received hello packets only from ID_1. Also, in Experiment 2 where ID_0 had stayed off the ground, packet loss rate was improved.

Then we have conducted the experiment to collect encounter records in real environment using Motes. In this experiment, students carried Motes and accumulated encounter records on Motes. All the encounter records accumulated on Motes were collected to a host computer through a base station after the experiment, and we estimated the trajectories by TRACKIE. Then we have computed the average position error (APE). Fig. 9 shows

TABLE III

AVERAGE POSITION ERROR (METERS)

	In Simulations		In Real Env. (m)
	IDEAL	DOI	
Initial Solution	2.88	3.94	3.51
TRACKIE	1.79	2.57	3.19
TRACKIE_EX	1.79	2.09	2.24

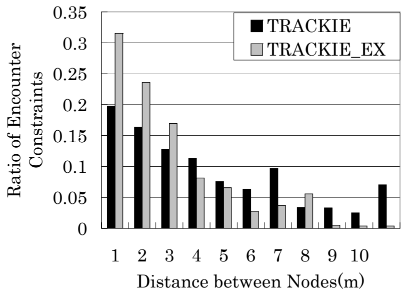


Fig. 10. Relation between Encounter Information and Distance

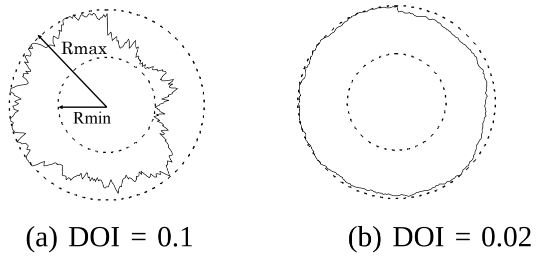


Fig. 11. Degree Of Irregularity Model

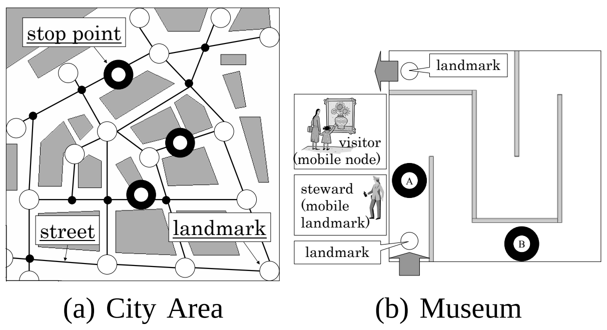


Fig. 12. Application Examples

the area map and a photo from the experiment. The area was $18m \times 18m$ free space with nine intersections. There are four landmarks placed at the intersections 1, 3, 7 and 9. Ten students carried Motes and moved following the RSD model. Motes transmitted hello packets for every second. We set Motes' radio power to -15dBm and the TRACKIE algorithm assumed $R = 3m$.

The last column of Table III shows APEs measured in this experiment. We can see that APE of TRACKIE was better than the initial solution, but it is still $1.06R$. This is because Motes sometimes received hello packets from others which were more than $3m$ away due to radio propagation fluctuation. To see this influence, in Fig. 10, we show the normalized distribution of transmitted distances of hello packets (see the case of "TRACKIE"). We can see that in about half of the cases, distances were over $3m$ and clearly this made estimation errors larger.

To regulate such fluctuation, we introduce a filter to verify the propriety of encounter records. It accepts an encounter record (t, ID_i, ID_j) if and only if (t, ID_j, ID_i) exists, in order to exclude encounter records created by irregular radio ranges. Hereafter, we call the TRACKIE algorithm with this filter TRACKIE_EX. In Fig. 10, we can see the case of TRACKIE_EX, and also we show APE of TRACKIE_EX in Table III ($2.24m$), which was quite better than TRACKIE.

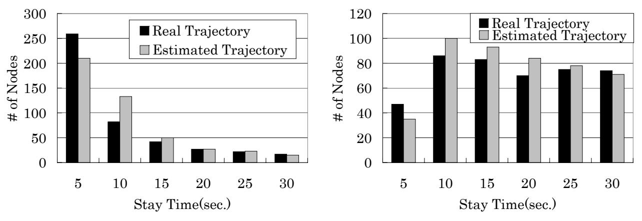
Second, we compare the experimental results in simulations and real environments. We conducted simulation in the same settings as this experiment, using the ideal radio propagation model ($R = 3m$). In the second column of Table III, the "IDEAL" subcolumn shows APE of TRACKIE and TRACKIE_EX in this simulation experiments. We can see considerable difference compared with the ones in the real environment. Considering this fact, we have introduced the Degree Of Irregularity (DOI) model proposed in Ref. [12]. The DOI model considers irregular radio pattern by varying the communication distance for each angle θ (θ = 0..360°). The parameter DOI denotes irregularity of the radio pattern. Fig. 11 shows the radio patterns with $DOI = 0.1$ and at $DOI = 0.02$. The lager DOI is, the lager irregularity of the radio pattern. We used this DOI model with $R_{min} = 3m$, $R_{max} = 8m$ and $DOI = 0.1$ and conducted simulations. The subcolumn "DOI" of Table III shows the result. The difference from the real environment became smaller, and in particular the difference in TRACKIE_EX was less than $0.2m$.

From the results above, we can say that the position errors of TRACKIE_EX was small enough and the validity of the experiment in real environment was proved by simulations with the realistic radio propagation model.

VII. APPLICATION

We give two application examples and show the results of simulations of TRACKIE_EX using the DOI model.

Trajectories of pedestrians in city regions can be used to analyze their behavior and are useful for many cases



(a) at point A (b) at point B

Fig. 13. Stay Time at each point

including evacuation planning and commercial use.

We used the real map of $500m \times 500m$ region in front of the Osaka train station like Fig. 12(a) with 17 landmarks. We put 3,000 nodes and let 60 % of them move by the RSD mobility and let 40 % move by "customer mobility" where each node visits several points and stops for several tens of seconds. The speeds of nodes followed the normal distribution with average 1.5m/s and variance 0.1. The DOI model with $R_{min} = 7.5m$, $R_{max} = 12.5m$ and $DOI = 0.1$ was used but the TRACKIE_EX algorithm used $R = 10m$. Then APE was $4.05m$ and this is accurate enough for the purposes mentioned above.

Also, trajectories in museums or shops let us know which exhibits or products each person was interested in and how long he/she stayed in front of them. We used the map of $50m \times 50m$ shown in Fig. 12(b) with just two stationary landmarks at gateways. In addition, we also put 17 stewards as "mobile" landmarks, since they walked along the scheduled routes in scheduled time with constant speed $2m/s$. We exploited "visitor mobility" where each node stops for randomly decided time durations in front of the exhibits deployed at $10m$ intervals. We used the same velocity distribution of the former example. The DOI model was used with $R_{min} = 3m$, $R_{max} = 8m$ and $DOI = 0.1$. TRACKIE_EX used $R = 3m$. In these settings, APE was $2.2m$, which is accurate enough to identify the trajectories. Also, we show the distribution of residence times at points A and B in Fig. 13. This result indicates that we could estimate with enough accuracy their residence motion, and such information is very useful to learn the popularity of exhibits and so on.

VIII. Conclusion

In this paper, we have proposed an algorithm to estimate the trajectories of mobile nodes. This is a low-cost, simple and accurate method because we do not need many landmarks. Also we do not require that ad-hoc networks are connected (we assume opportunistic communication). Analyzing the upper bound of position errors and the impact of landmark deployment to position errors is part of our future work.

References

[1] "MobiREAL simulator web page," tools are available on `http://www.mobireal.net`.

[2] K. Maeda, K. Sato, K. Konishi, A. Yamasaki, A. Uchiyama, H. Yamaguchi, K. Yasumoto, and T. Higashino, "Getting urban pedestrian flow from simple observation: Realistic mobility generation in wireless network simulation," in *Proc. of ACM/IEEE Int. Symp. on MSWiM 2005*, 2005, pp. 151–158.

[3] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proc. of IEEE INFOCOM*, 2000, pp. 775–784.

[4] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket location-support system," in *Proc. of MobiCom 2000*, 2000, pp. 32–43.

[5] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proc. of MobiCom 2001*, 2001, pp. 166–179.

[6] A. Uchiyama, S. Fujii, K. Maeda, T. Umedu, H. Yamaguchi, and T. Higashino, "Ad-hoc localization in urban district," in *Proc. of IEEE INFOCOM Mini-Symposium*, 2007, pp. 2306–2310.

[7] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 28–34, 2000.

[8] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proc. of MobiCom 2004*, 2004, pp. 45–57.

[9] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network," in *Proc. of IPSN 2003*, 2003, pp. 333–348.

[10] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proc. of IEEE GlobeCom 2001*, 2001, pp. 2926–2931.

[11] S. Yang, J. Yi, and H. Cha, "HCRL: a hop-count-ratio based localization in wireless sensor networks," in *Proc. of IEEE SECON*, 2007, pp. 31–40.

[12] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proc. of MobiCom 2003*, 2003, pp. 81–95.

[13] S. Guha, R. Murty, and E. G. Sirer, "Sextant: a unified node and event localization framework using non-convex constraints," in *Proc. of MobiHoc 2005*, 2005, pp. 205–216.

[14] Y. Shang, W. Ruml, and Y. Zhang, "Localization from mere connectivity," in *Proc. of MobiHoc 2003*, 2003, pp. 201–212.

[15] D. K. Goldenberg, P. Bihler, M. Cao, J. Fang, B. D. O. Anderson, A. S. Morse, and Y. R. Yang, "Localization in sparse networks using sweeps," in *Proc. of MobiCom 2006*, 2006, pp. 110–121.

[16] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, 2003.

[17] F. Zhao, J. Liu, J. Liu, L. Guidas, and J. Reich, "Collaborative signal and information processing: an information directed approach," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1199–1209, 2003.

[18] C. Taylor, A. Rahimi, J. Bachrach, and H. Shrobe, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *Proc. of IPSN 2006*, 2006, pp. 27–33.