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
E-mail: {*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 system. The method achieves low-cost estimation of trajectories and provides accurate solution (the average estimation error was less than 40% of the 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 about 3m.

1. Introduction

Trajectory information has been recognized significant for provisioning pervasive services. For example, personal mobile assistants may be more intelligent so that they can anticipate walking paths of the users and suggest shops or restaurants on those anticipated paths if they know the trajectories of past walkers. GPS receivers are useful for collecting trajectories, but 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. Existing positioning techniques using ad-hoc communication facility may be used. However, most of them assume multi-hop ad-hoc wireless networks which are sometimes partitioned and instable because of mobility of nodes.

In this paper, we propose an off-line algorithm called TRACKIE to estimate trajectories of mobile nodes using encounter information, which is a record of encounter of two mobile nodes or encounter of a mobile node and a landmark. We do not assume multi-hop ad-hoc networks nor GPS. Instead we use more opportunistic communication style as studied more recently, hence we only require short-range wireless communication devices and a small amount of memory to accumulate encounter information. In addition, we are able to determine trajectories of mobile nodes more accurately in an "off-line" manner. We have evaluated the proposed method using network simulator MobiREAL [1, 8]. From the experimental results, we have shown that our two-phase strategy, which consists of a greedy position determination phase and a global refinement phase, 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. Also, we have evaluated TRACKIE using MICAz Mote and have shown that estimation error was about 2m where the wireless range was about 3m.

2. Related work and contribution

Most 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 [2, 10, 11], and (2) *range-free methods* that only use wireless connectivity information [4, 6, 7, 9, 12, 14, 15]. The range-based methods may be able to achieve higher accuracy than range-free methods, but they require higher costs for measurement equipment. On the contrary, range-free methods are cost-efficient, but less accuracy is demerit. Centroid [4] and MCL [7] directly receive location information from landmarks and estimate positions. Amorphous [9] and HCRL [15] receive location information from landmarks in multi-hop manners and estimate positions. In Sextant [6] and UPL [14], each node uses the estimated positions of its neighbor nodes to estimate its own position. Some methods like MDS-MAP [12] calculate relative positions based on connectivity information, and then decide absolute positions by positions of landmarks. TRACKIE is different from the ex-

0-7695-3113-X/08 $25.00 © 2008 IEEE
DOI 10.1109/PERCOM.2008.112

isting range-free methods since it computes trajectories off-line. Taking the merit of off-line computation, TRACKIE allows us to estimate trajectories more accurately using both spatial and temporal relationship than existing positioning methods that use only spatial relationship.

Also localizing and tracking methods for mobile nodes in sensor networks have been investigated so far [3, 13, 16]. These methods assume that sensors are deployed in high density and these sensors send sensed events to base stations. Then the objects and their movements are detected by fusing the collected information. These methods assume centralized off-line computation, but situations are very different because they use information from many stationary sensors. On the contrary, TRACKIE uses only a few landmarks and the encounter information accumulated by mobile nodes.

3. Problem definition and algorithm overview

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 mobile nodes and landmarks are capable to communicate with each other via 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 broadcasts *hello packets* at a regular interval to its neighbors. Every time node i receives a hello packet 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 packet and ID_i (ID_j) is the ID of node i (node j). Here, we call this tuple *encounter record*. When a mobile node 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 the information about the positions of the 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.

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 moving between two landmarks. An intuitive and simple solution is to give all the encounter records to constrain the distance between nodes (*i.e.* the distance must not be greater than the communication 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, it is not easy to find a solution which reproduces natural trajectories of pedestrians since there may be many feasible solutions that satisfy the constraints.

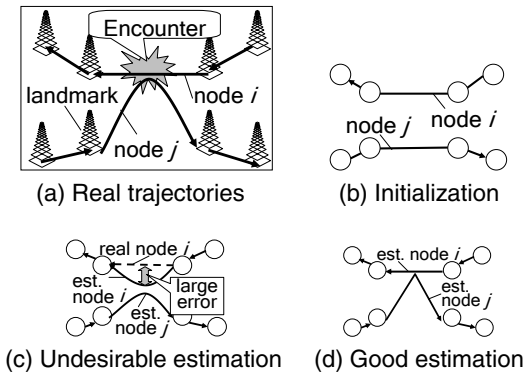


Figure 1. Estimation of trajectories.

TRACKIE first identifies such trajectories that can be estimated easily with high fidelity, and estimates them. Then, it uses the estimated trajectories as "quasi landmarks" to localize the others. In this paper, we consider that mobile nodes with faster speeds between two landmarks move more straightly, and thus their trajectories are easier to estimate. 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.

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

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, we can consider the constraint that the positions of their foot-

prints at time t must be within the radio range R (called *encounter constraint* hereafter). Here, to satisfy this constraint, 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 (c)). Thus we first identify the trajectory which is a straight line with high possibility, and use it to modify the other trajectories. Also, in order to avoid big modification of trajectories that may lead to unnatural trajectories, we iteratively localize the footprints. For example, in Fig. 1 (d), we can see that node i's trajectory is determined first, and node j's trajectory is stretched by the encounter constraint with node i.

Finally, given the modified trajectories, we should modify them such that they never traverse obstacles like buildings and walls and never experience drastic change of speeds and directions. Since such modification needs to check all the footprints, we apply a global optimization technique based on Simulated Annealing (SA) in the third phase and call it *SA-based modification phase*.

4. Algorithm description

In the algorithm, a "state" is associated with each footprint. 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.

4.1. 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 trajectory". So, an initial trajectory is regarded as a set of straight lines between landmarks. For readability, we use an example shown in Fig. 2 (a) in the following sections. Fig. 2 (b) shows the initial trajectory where $p_{i,0}$, $p_{i,40}$, $p_{j,0}$ and $p_{j,40}$ are *anchored*.

4.2. 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

Iterative Modification Phase

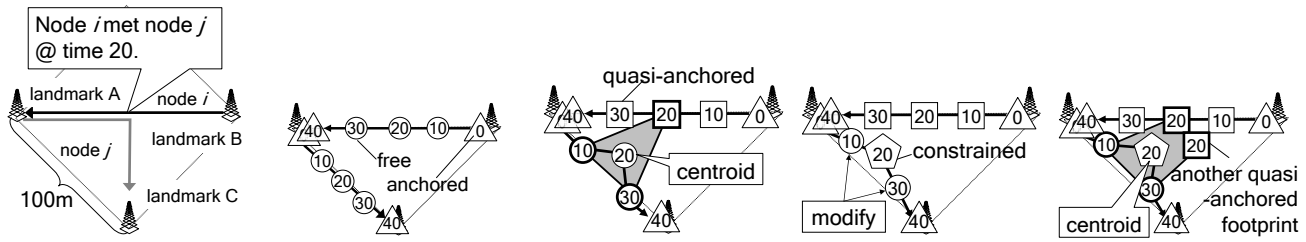
```
1  initialize NQ as set of all the free footprints
2  do{
3  //determine footprints to quasi-anchored
4     cand ← fastest_footprints(NQ)
5     for (each p_{i,t} ∈cand) {
6        p_{i,t}.state ← "quasi-anchored"
7        NQ ← NQ − {p_{i,t}}
8     }
9  //localize non-quasi-anchored footprints
10    for (each p_{i,t} ∈ NQ)
11       for (each p_{j,t} ∈encounter(p_{i,t})){
12          if(p_{j,t}.state = "quasi-anchored")
13             EQ ← EQ ∪ {p_{j,t}}
14          p_{i,t}.pos = centroid(p_{i,t−1}, p_{i,t+1},EQ)
15       }
16    for (each constrained footprint p_{i,t})
17       modifyFreeFootprint(p_{i,t}, nextConstrained(p_{i,t}))
18 }while (NQ ≠ φ)
```

Figure 3. Algorithm description.

a *quasi-anchored* footprint is treated as an *anchored* footprint. The iterative modification phase modifies the initial trajectory considering the encounter constraints between mobile nodes. This is done by gradually increasing *quasi-anchored* footprints and iteratively localizing the other footprints using the encounter constraints with both the *anchored* footprints and the *quasi-anchored* footprints. This terminates when all the footprints except the *anchored* footprints become *quasi-anchored*. Now, we define *DCQ (Determining Candidate to be Quasi-anchored) policy* that determines the ordering to make footprints *quasi-anchored*, and *LNQ (Localizing Non-Quasi-anchored footprints) policy* that localizes the positions of *non-quasi-anchored* footprints. In Fig. 3, we give the formal description of this iterative modification phase. The lines from 3 to 8 correspond to DCQ policy and the lines from 9 to 17 correspond to LNQ policy. Two policies are explained below.

Here, we address the DCQ policy. We let $\overrightarrow{v_{i,t}}$ denote the node i's estimated velocity vector at time t in the initial trajectory, 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 trajectory. After obtaining the initial trajectory, we compute $|\overrightarrow{v_{i,t}}|$ for each pair of $i \in N$ and $t \in [0..T]$. In DCQ policy, we choose the trajectory which is the straightest of all the trajectories between landmarks. In the case there is not so much of a difference among the velocity of the nodes, it corresponds to selection of the footprints of the nodes moving fastest between two landmarks in the initial trajectory. So, using magnitude of these velocity vectors, 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 $p_{i,t+1}$,...,$p_{i,t+l-1}$ are not less than any of the others (they must be equal due to the property of the initial trajectory).

Here, we address the LNQ policy. We introduce an addi-



(a) Real trajectories (b) Initial trajectory (c) Modification of initial trajectory

Figure 2. Exemplification of iterative modification phase.

tional state, *constrained*, for this localization process. Once a footprint is modified with encounter constraints with *anchored* or *quasi-anchored* footprints, it is set to be *constrained*. In LNQ policy, we identify each footprint $p_{i,t}$ that is either *free* or *constrained*, and has encounter constraints with some *anchored* or *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 *anchored* or *quasi-anchored* footprint $p_{j,t}$ where node i and j encountered at time t. By using the positions of the predecessor and successor footprints, we avoid big change of positions by one attempt of localization. Thirdly, we set the state of $p_{i,t}$ to be *constrained* if it is *free*. After these steps, for *free* footprints $p_{i,t-k+1}, ..., p_{i,t-1}$ 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 $p_{i,t+k'}$.

We exemplify by an example in Fig. 2 (c). First, we choose the sequence $p_{i,10}$, $p_{i,20}$, $p_{i,30}$, and set their states to be *quasi-anchored*. Second, 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. Then, 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 $p_{a,20}$. 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 iterative modification phase terminates.

4.3. SA-based modification phase

In general, Simulated Annealing 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 acceptance ratio $P(rand(0,1) < e^{-\Delta Cost/T})$, where $\Delta Cost$ 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 refine the trajectories modified by the iterative modification phase using the SA technique in Ref. [5]. 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;

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

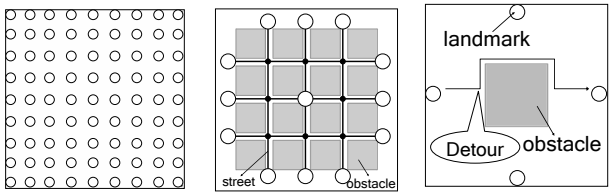
where c_u is a constant number and we set $c_u = 2$ empirically. As the cost function, we consider the following three functions $Cost_a$, $Cost_b$ and $Cost_c$, and we execute SA optimization for each function.

Minimizing encounter constraint errors: If two nodes i and j encountered at time t, the distance between the two footprints $|\overrightarrow{p_{i,t}p_{j,t}}|$ must be less than or equal to the radio range R. It is natural to minimize the violation of encounter constraints for better solutions. Thus we define $Cost_a = \sum_i \sum_j \sum_t e_{i,j,t}$, where $e_{i,j,t}$ is *encounter constraint errors* defined by the formula (1). If we can reduce the value of $Cost_a$, we can mitigate inconsistency with encounter constraints.

$$e_{i,j,t} = \begin{cases} 0, & |\overrightarrow{p_{i,t}p_{j,t}}| \leq R \\ |\overrightarrow{p_{i,t}p_{j,t}}| - R, & otherwise \end{cases} \quad (1)$$

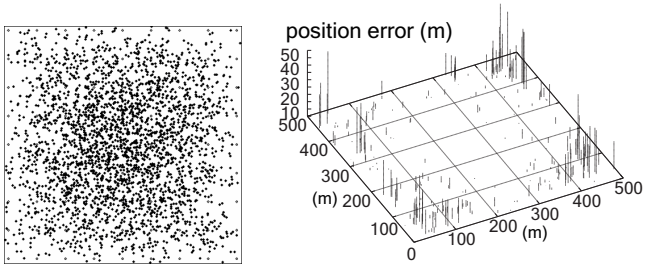
Minimizing obstacle constraint errors: When we can obtain obstacle information, we define *obstacle constraint error* $o_{i,t}$ as the distance from the borderline of an obstacle to footprint $p_{i,t}$ if $p_{i,t}$ is located inside the obstacle. Then we define $Cost_b = \sum_i \sum_t o_{i,t}$.

Minimizing speed and direction fluctuation: We would like to derive trajectories such that speeds and moving directions of nodes do not change suddenly. We define $\overline{V_i}$ as node i's average speed, which is defined as



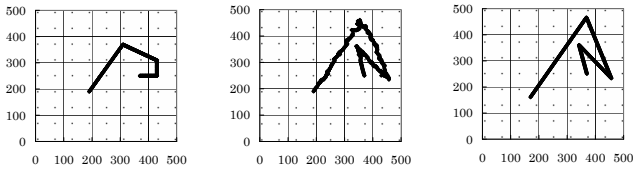
(a) Free space (b) Manhattan (c) Void

Figure 4. Maps of simulated areas (circles represent landmarks).



(a) Node distribution (b) Positioning errors

Figure 5. A snapshot in zigzag scenario.



(a) initial (b) estimated (c) real

Figure 6. Estimation in zigzag scenario.

$\overline{V_i} = \sum_t |\overrightarrow{v_{i,t}}|/T$. We also define $s_{i,t} = |\overrightarrow{v_{i,t}}| - \overline{V_i}$. Here, $\sum_t s_{i,t}$ denotes the sum of speed fluctuation of node i. We also define $a_{i,t}$ in the formula (2). Then, $\sum_t a_{i,t}$ denotes the sum of moving direction fluctuation 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 and moving direction fluctuation.

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

We first use $Cost_a$ in order to derive the trajectories less inconsistency with encounter constraints. Then, using $Cost_b$ and $Cost_c$, we obtain more natural trajectories.

5. Simulation

5.1. Performance evaluation of TRACKIE

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 RWP (Random WayPoint) model in the free space map of 500m $\times$ 500m as shown in Fig. 4 (a) to see the reproducibility of zigzag trajectories. The *street walking scenario* assumes the RSD (Random Street Decision) model in the Manhattan map of 500m $\times$ 500m where 6 streets of 20m width are deployed as shown in Fig. 4 (b). In the RSD model, each node moves along a street, and randomly chooses a street except the backward direction at each intersection. Finally, the *detour scenario* assumes four types of 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). We have evaluated estimation of trajectories in the environment where there is no straight trajectory between landmarks.

We have conducted simulations using the network simulator MobiREAL [1, 8] We have used the LOS (Line Of Sight) radio propagation model where two nodes can communicate with each other if they are visible from each other and the distance between them is less than R_{max}. We assume that R is equal to R_{max} and set both R_{max} and R to 10m. The speeds 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. All the nodes sent a hello packet at every second.

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

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

$$AAE = \frac{\sum_i \sum_t \arccos\left(\frac{\overrightarrow{avgv_{i,t}} \cdot \overrightarrow{\hat{avgv}_{i,t}}}{|\overrightarrow{avgv_{i,t}}||\overrightarrow{\hat{avgv}_{i,t}}|}\right)}{T \cdot N}$$

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

In Table 1, we have shown APEs and AAEs 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, APE (of the whole area) in the zigzag scenario was more than 5m,

Table 1. APEs / AAEs (bold font indicates the best throughout scenario).

	zigzag (m) / (rad)		street walking	detour
	Whole area	Central area	(m) / (rad)	(m) / (rad)
(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

which was rather larger than those in the other two scenarios. On the other hand, APE of the central area (300m $\times$ 300m area) 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 shown by Fig. 5 and it is hard for the nodes near the boundary to encounter the other nodes. If nodes had enough chance to encounter each other, TRACKIE could accurately estimate the trajectories that are irregular (and thus complex). In fact, the trajectories in the central area can be estimated accurately like Fig. 6

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. In contrast, in the detour scenario, we can see that APE after the iterative modification phase was still large, 7.69m, because no node is able to move straightly between two landmarks in this scenario. However, the SA-based modification phase improves APE to 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 accuracy could not be improved without the iterative modification (case (iii')). Thus this indicates that both modification phases help each other to accomplish high accuracy.

5.2. Comparison with existing methods

We have compared the performance of TRACKIE with MCL [7] and Amorphous [9] under different parameter settings, although this may not be fair comparison because MCL and Amorphous are on-line (real-time) and distributed localization techniques. MCL stands for the Monte Carlo Localization and is a range-free localization method where each node manages its area of presence and refines it whenever it encounters a landmark. Amorphous is also a range-free method 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 used the zigzag scenario in Section 5.1 where the speeds of nodes followed the normal distribution with mean 2.0m/s and variance 0.1. Also we have set both R_{max} and R to 10m, and have set the ranges of parameters as shown in Table 2. In each experiment, we have varied one of these parameter values and let the others remain the default values emphasized by a bold font. We have measured the APE (average position error) in the central area.

Number of nodes: Table 2 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 each node can estimate the distance from landmarks more accurately with 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.

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 Table 2. 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. In the other two methods, only the receiver can take this benefit. As a result, TRACKIE is robust to longer intervals.

Number of landmarks: Table 2 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.

6. Experiment using Micaz Mote

We have conducted the experiment to evaluate TRACKIE using collected encounter records in real

Table 2. APE of TRACKIE, MCL and Amorphous under different parameter settings.

Parameter		TRACKIE	MCL	Amorphous
# of Nodes	**3,000**	4.04	56.4	65.2
	4,000	3.97	53.3	50.7
	5,000	3.80	53.4	42.6
Interval of Hello Packets	**1(s)**	4.04	56.4	65.2
	10	4.64	93.6	72.6
	20	6.33	114.8	86.0
# of Landmarks	57	5.14	95.5	65.7
	69	4.70	79.9	63.2
	81	4.04	56.4	65.2

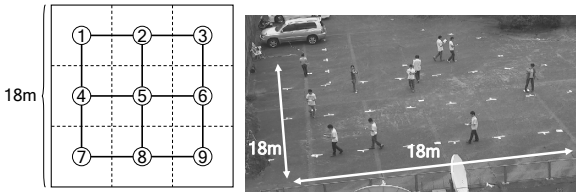
Table 3. Packet loss rate.

Source node ID	ID_1	ID_2	ID_3	ID_4	ID_5
Packet loss rate	0	0.48	0.70	0.77	0.97

environment with Micaz Motes. As preliminaries, we have measured radio range of Motes in the open air with radio power -15dBm. We used 6 Motes deploying (ID_0,...,ID_5) on the ground along a straight line at 1 meter interval in this order. Then, we have held ID_0 at 1 meter height from the ground and measured the number n of received hello packets at ID_0 in the situation the nodes ID_i ($i = 1..5$) were transmitting hello packets every 1 sec. Table 3 shows packet loss rates defined as $(T - n)/T$. From the result, we assume the wireless range of Motes R is 3m.

Now, we explain how to evaluate TRACKIE in real environment. In this experiment, students carried Motes transmitting hello packets every 1 sec. 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 APE (average position error). Fig. 7 shows the area map and a photo from the experiment. The area was $18m \times 18m$ free space with 9 intersections. There are 4 landmarks placed at the intersections 1, 3, 7 and 9. 10 students carried Motes and moved following the RSD model.

The last column of Table 4 shows APEs measured in this experiment. We can see that APE of TRACKIE was better than the initial trajectory, but it is still $1.06R$. This is because Motes sometimes are supposed to receive hello packets from the others which were more than 3m away because of phasing caused by mobility of nodes or reflection from the ground. To see this influence, we show the normalized distribution of transmitted distances of hello packets in Fig. 8. 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



(a) Experiment map (b) Scene of experiment

Figure 7. Experiment environment.

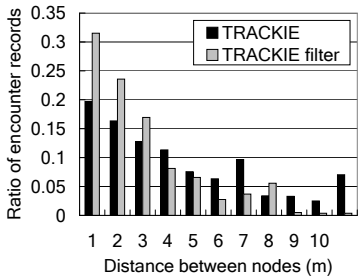


Figure 8. Distribution of transmitted distances of hello packets.

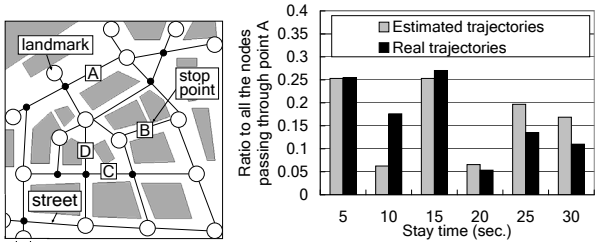
the propriety of encounter records. It accepts an encounter record (t, ID_i, ID_j) if (t, ID_j, ID_i) exists, in order to exclude encounter records created by irregular radio ranges. Hereafter, we call the TRACKIE algorithm with this filter *TRACKIE*$_{filter}$. In Fig. 8, we can see that TRACKIE$_{filter}$ can eliminate influence of irregular radio propagation. Also, from Table 4, APE of TRACKIE$_{filter}$ is 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 LOS model, the ideal radio propagation model. In the second column of Table 4, the "LOS" subcolumn shows APE of TRACKIE and TRACKIE$_{filter}$ with $R = 10$m in this simulation experiments. We can see considerable difference compared with the ones in the real environment.

Considering this fact, we introduce the more real radio propagation model, RIM (Radio Irregularity Model) proposed in Ref. [17]. RIM considers irregular radio pattern by varying the communication distance for each angle θ (θ = 0..360°) between R_{max} and R_{min} according to DOI, which denotes irregularity of the radio pattern. In this case, we set $R_{max} = 8$m, $R_{min} = 7$m, and $DOI = 0.1$. The subcolumn "RIM" of Table 4 shows the result. The difference from the real environment became smaller, and in particular the difference in TRACKIE$_{filter}$ was less than 0.2m. From the above results, we can say that the position errors of TRACKIE$_{filter}$ were small enough and the validity of the experiment in real environment was proved by simulations with the realistic radio propagation model.

Table 4. APEs of simulation and real env.

	Simulation (m)		Real env.
	LOS	RIM	(m)
Initial trajectory	2.88	3.94	3.51
TRACKIE	1.79	2.57	3.19
TRACKIE$_{filter}$	1.79	2.09	2.24



(a) City area (b) Stay time at point A

Figure 9. Application example.

7. Application

We give an application example and show the simulation result of TRACKIE$_{filter}$ under RIM. Trajectories of pedestrians in city regions can be used to analyze their behavior. We used the real map of $500m \times 500m$ region in front of the Osaka train station like Fig. 9 (a) with 17 landmarks. We put 3,000 nodes and let 60 % of them move along the RSD mobility and let 40 % move along *customer mobility* where each node visits 4 points (point A..D) randomly 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 RIM with $R_{min} = 7.5$m, $R_{max} = 12.5$m and $DOI = 0.1$ was used. We set R to 10m. APE of TRACKIE$_{filter}$ was 4.05m. This result indicates that we could estimate the trajectories in city area with enough accuracy. Also, we have computed the stay time from real trajectories and estimated trajectories and show the distribution of stay times at point A in Fig. 9 (b). Such information is very useful to learn trend of pedestrian's behavior.

8. CONCLUSION

In this paper, we have proposed TRACKIE, an algorithm to estimate the trajectories of mobile nodes. This is a low-cost, simple and accurate method. Also we do not require many landmarks nor well-connected ad-hoc networks (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] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proc. of INFOCOM 2000*, pages 775–784, 2000.

[3] R. R. Brooks, P. Ramanathan, and A. M. Sayeed. Distributed target classification and tracking in sensor networks. *Proc. IEEE*, 91(8):1163–1171, 2003.

[4] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, 2000.

[5] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Trans. Mathematical Software*, 13(3):262–280, 1987.

[6] 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*, pages 205–216, 2005.

[7] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proc. of MobiCom 2004*, pages 45–57, 2004.

[8] 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 MSWiM 2005*, pages 151–158, 2005.

[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*, pages 333–348, 2003.

[10] D. Niculescu and B. Nath. Vor base stations for indoor 802.11 positioning. In *Proc. of MobiCom 2004*, pages 58–69, 2004.

[11] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *Proc. of MobiCom 2000*, pages 32–43, 2000.

[12] Y. Shang, W. Ruml, and Y. Zhang. Localization from mere connectivity. In *Proc. of MobiHoc 2003*, pages 201–212, 2003.

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

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

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

[16] F. Zhao, J. Liu, J. Liu, L. Guidas, and J. Reich. Collaborative signal and information processing: an information directed approach. *Proc. IEEE*, 91(8):1199–1209, 2003.

[17] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Models and solutions for radio irregularity in wireless sensor networks. *ACM Trans. Sensor Networks*, 2(2):221–262, 2006.