Cooperative Trajectory-based Map Construction

Wei Chang, Jie Wu, and Chiu C. Tan
Department of Computer and Information Sciences
Temple University, Philadelphia, PA 19122
Email: {wei.chang, jiewu, cctan}@temple.edu

***Abstract*—Map construction is an integral part of many location-based services. In this paper, we propose a feedback-based heuristic map construction algorithm (FHMCA). This is a lightweight, cooperative, map construction technique which can accurately capture the unique characteristics of each intersection and the length of every road without requiring the users to transmit large amounts of data or use GPS. The proposed algorithm improves the bandwidth and energy efficiency of cooperative map construction as no detailed maps are needed. However, the resulting map is still useful for location-based services. Moreover, our method is applicable when the existing of malicious users, who report wrong data. We validate the effectiveness of our solutions through extensive simulation experiments.**

***Index Terms*—encounter, intersection feature, map construction, subgraph matching, trust.**

I. Introduction

We are entering a mobile computing era where people, services, and locations are connected with each other. Recently, we have witnessed the explosive growth of local-based services (LBSs). There are at least two fundamental issues involved with any LBS: localization of clients, and construction of a location-specified database. However, the solutions to both of these issues require a common problem to be solved first: the construction of the service zone map. Once this map is constructed, clients can be localized on it, and service data can also be associated with it.

Cooperative trajectory mapping is an emerging technique for map construction, which takes advantage of different sensors that are embedded into smartphones to create a map of the whole region by utilizing users' trajectories. This type of map is known as a *trajectory map*. Trajectory maps are being used in various applications, such as traffic monitoring [1], public transportation tracking [2]–[4], and people localization [5]–[7].

Although the GPS-based maps have been widely used, there are still many limitations with them, such as the problem with the unavailability of GPS signals in certain environments, like indoors or in tight urban spaces, high energy consumption [5], [8], a lot of sampling data for constructing a map, and so on. As a result, we generally avoid the use of GPS when building the trajectory map. Instead, the smartphone's sensors, the accelerometer and electronic compass, are used to collect information such as the moving direction and distance between consecutive sampling times [8]. These data are then transmitted to a central depository via 3G connection, which collects and processes the readings from multiple users to arrive at a comprehensive trajectory map.

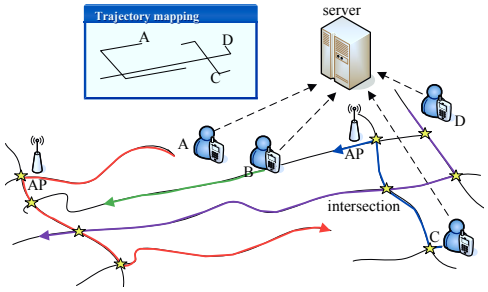


Fig. 1. System model. The system normally contains a central server and several users. Each user periodically reports his trajectory in the form of intersection and displacement. The access point provides land marks for intersection matching.

Prior research [5], [8] focuses on constructing a map by recording the shape of road segments. However, these methods need to transmit a huge amount of information to the server so that the accurate shape of road can be preserved, and they are vulnerable to adversaries who report no-existing trajectories.

In this paper, we consider a new method of map construction that emphasizes the accuracy of *trajectory intersections' connectivity and their distance*, rather than the exact shape of each road segment; we are interested in navigating from one place to the other and the total time of commuting from one place to another. We name the constructed map as an intersection connectivity map. Although the real path may be a curve, depicting the real path as a straight line also allows us to move to the correct destination. Clearly, this problem differs from the traditional localization problem where accurate physical locations are usually required.

In this paper, we propose a new map construction algorithm that is based on encounter information, intersection features, and the server's feedback. Our algorithm can be applied in different types of regions by only using a relatively small amount of data from users, and our method is resistable to adversaries through the use of a trust mechanism, since the reported trajectories can be verified by other users. Moreover, after a slight adjustment, the proposed algorithm can also be used in the localization problem. In short, the contributions of this paper are as follows: 1. To the best of our knowledge, we are the first to construct a road map without using the shapes of the road. 2. We design a self-adjusting system that dynamically changes the sampling policy in different physical conditions. 3. The proposed algorithm requires a much smaller amount of data transmitted from users. 4. We validate the effectiveness of our solutions through extensive simulation experiments.

II. Related work

We can generally categorize the map construction methods into two groups: GPS data-based map construction and non-GPS data-based map construction.

Traditional GPS-data-based map construction methods typically apply one of the following four data update strategies: (1) The periodical update strategy [9], [10] is the simplest location update strategy. All of the mobile users maintain a synchronous clock, and they report their location data at a fixed time interval. The system treats users as stationary between the time intervals. (2) The distance-based update strategy [11], [12] requires the server to update when a user's displacement from the last reported location is greater than a pre-defined distance threshold. The frequency of location updates in this strategy is dependent on the speed of each individual user. (3) The vector-based update strategy uses the velocity vector of a user to make a location prediction. The location update is only sent when the difference between the current location and the predicted location is larger than a pre-defined distance threshold [13]–[15]. (4) The segment-based update strategy [16] limits the amount of location updates by allowing users only to update the location at the end of each segment. The system of using such an update strategy assumes that the users can change their movement direction only at a segment's end node. The idea of our map construction method is similar to this update strategy, but it is different in that we do not use GPS locations.

In paper [8], the authors propose a GPS data-free localization system. They allow users' mobile phones to be embedded with multiple sensors. Each user periodically reports their trajectories (in the form of moving directions and displacements from the last reported position) and encounter information to the server, then the server can build up a map. Other work by [17], [18] also used similar ideas for cooperative trajectory mapping. Our method also uses the multiple sensors embedded in phones. However, the main difference is that prior works focus on using the sensors to record the shape of road segments. Our work focuses on recording the unique features of each intersection.

III. System Model

Our system requires the following components, as shown in Fig. 1: (1) *A central server.* The server collects user data and uses that information to build the map. Except for the collected data, the server does not have any domain knowledge about the map. (2) *Mobile clients.* We assume that every client has a mobile phone that is equipped with three sensors: an accelerometer, an electric compass, and an encounter sensor. The accelerometer is used for counting the displacement between any two consecutive sampling times; the compass is used for measuring the directions of entering and leaving an intersection; the encounter sensor is used to periodically signal a nearby users presence as well as record the presence of other users, which can be accomplished by Bluetooth module. (3) *Land mark (LM).* The LM can be any fixed location reference, such as a WiFi access point. The physical location of the LM is not necessary to be known. The purpose of using LMs is to provide a unique identity for matching.

TABLE I
Table of common notations

G_i	user i reported trajectory (subgraph of G)
C	the constructing map at a server
V_C	the intersection data set at server
E_C	the edge data set at server
w_i	user i's last reported intersection
$Deg(v_i)$	node-degree of v_i
$NVS(v_i)$	location v_i's directly neighboring node set
$NES(v_i)$	location v_i's directly connected edge set
$Len(v_iv_j)$	the length of edge v_iv_j
$Dir(v_i, e/d)$	the entering/departure direction of v_i
$v_i.LM$	the associated LM of v_i
M, N	the number of users and intersections
S	speed of a user

At each intersection, a user will report a displacement from the last intersection to the server. The user can also report the directions of which they were entering or leaving an intersection, and the number of paths connected with an intersection (we also call it as node-degree). However, these are optional: in Section IV, we discuss at what scenarios a user should report what kind of data. We assume that a part of intersections can provide some special LMs: whenever a user passes these intersections, the device will report the LM to the server, as well as other data. We assume that users only move along roads and that the road segments are undirected. Finally, the intersections mentioned in this paper refer to the positions whose node-degree is greater than or equal to 3: a node with degree 2 is just another point on a road segment.

IV. The construction of the map

In this section, we formally introduce how FHMCA performs the map construction. We first formulate the trajectory matching in map construction by using graph theory. Then, we propose three basic conditions. Finally, we present FHMCA.

A. Problem formulation and solution overview

The challenge of cooperative map construction is correctly interpreting every user's data and merging them together. In our problem, we emphasize the connectivity of intersections and the length of road. As a result, the whole map can be represented by a undirected graph $G = (V, E)$, composed of nodes (intersections) $V = \{v_1, v_2, ...v_N\}$ and undirected edges (road segments) $E = \{v_iv_j | v_i, v_j \in V\}$. In this paper we refer to the whole map G as the constructed map, and we refer to the constructing map at the central server as C. As defined, each edge is associated with two and only two nodes, and we call these two nodes as the end nodes of this edge. The listing order of the two end nodes represents the moving direction. For example, edge v_iv_j means a user moved from intersection v_i to v_j. Each intersection may connect with several neighboring intersections by edges, and we use $Deg(v_i)$ to represent the node-degree of v_i. Table I summarizes the notations used.

In the beginning, the server does not have any knowledge about the map. As users continuously report their trajectory

information, the server obtains several partial views of the map G. We use G_i to represent all of the reported data of user i, and $G_i(t)$ stands for the reported data at time t. We formulate the map construction problem as finding a partial one-to-one mapping between nodes/edges in any pair of partial views. We use several types of reported data to map the location data of one user to another, and further merge these trajectories and construct the map. In a nutshell, for a pair of partial view, G_i and G_j, the resulting graph (merged trajectories), $G_i \bigcup G_j$, is the subgraph of G. We use $\int_0^T \bigcup_{i=1}^M G_i(t)\,dt$ to represent the construct map until time T.

***Definition** 1: Suppose that there are M users. The map construction problem can be defined as finding M partial one-to-one mappings such that $\int_0^\infty \bigcup_{i=1}^M G_i(t)\,dt = G$. If we assume that, at time T, the entire road segment has been passed by users at least one time, the map construction problem can also be represented as finding M partial one-to-one mappings such that $\bigcup_{i=1}^M G_i = G$.*

Unlike traditional map construction methods, the users in our problem do not record the shape of the road but instead only report their data at each intersection. Therefore, it is clear that the amount of transmitted data is much less than others. Consider that a LM may be available at some intersections, and that each intersection also involves many features, such as node-degree, path entering and leaving directions, and the length of the path. We can match the reported intersections from different users by using these features. Moreover, the physical encounters of users can also be used in matching since they preserve both temporal and spatial information.

In our solution, the amount of reported data is based on the server's feedback: constructing maps in ambiguous regions requires more information, and therefore the server should inform the corresponding users of the needed extra knowledge. In general, our algorithm consists of two phases: **Phase I:** in the beginning, users only provide basic information to the server, and the server makes its best effort to construct the map. After awhile, the reported data converges to a temporary map with several ambiguous regions where more information is needed. The server then goes into phase two. **Phase II:** whenever a user enters into an ambiguous region, the server will send a special message, requiring extra data from the user; when the user leaves the region, the server will also request that user to stop sending the extra data. By these two phases, the server can gradually gain enough knowledge for cooperatively constructing an intersection connectivity map.

In the following, we first introduce three basic conditions and their corresponding algorithms. After that, the general condition is introduced, followed by the presentation of our dynamically adjusted algorithm, FHMCA.

B. Basic conditions in map construction

During cooperative map construction, there are two issues that need to be resolved: (1) for any two users' reported data, we need to determine the intersection set of them, $G_i \bigcap G_j$, such that G_i and G_j can be correctly connected; (2) for a piece of new information, we need to determine whether the incoming data has already been preserved by the currently under-construction map or not, which means we need to determine whether $G_i(t) \subseteq C$. If we can find at least one combination of unique characteristics of any intersection, the above two problems will be overcome. In this part, we introduce three basic conditions which can provide such a unique feature combinations.

1) The first basic condition (uniqueness-based basic condition): if the node degree of each intersection or the length of each road segment is unique, the map can be constructed. However, this condition is exclusively determined by realistic road connectivity, and therefore the application scenarios are very limited. The corresponding algorithm for the uniqueness-based basic condition is given by Algorithm 1.

Algorithm 1 Uniqueness-based Map Construction Algorithm (UMCA)

for Each incoming data $G_i(t) = \{v_i, v_k v_i\}$ **do**
  if $Deg(v_i) \notin \{Deg(v_j)\}, v_j \in NVS(w_i)$ **then**
    Add v_i into the constructing map C
    Connect previous location w_i with v_i in C
  else
    if $Len(v_k v_i) \notin \{Len(v_j v_i)\}, v_j v_i \in NES(w_i)$ **then**
      Connect w_i with v_i in C
  Record current location of user i by $w_i = v_i$

2) The second basic condition (LM-based basic condition): every intersection has a unique LM: the trajectory of a user can be regarded as a sequence of LMs. Since each road can be represented by two consecutive intersections, if the reported LMs of two users contain two consecutively the same LMs, it means that these two users have passed the same road in the past; if a common LM is found in two users' reported data, it means that these two users spatially passed the same intersection. Based on these facts, the server can connect all of users' reported trajectories by matching LMs. As the reported data also contains the length of displacements between intersections, the server eventually can construct an intersection connectivity map. The corresponding algorithm for the LM-based basic condition is given by Algorithm 2.

Algorithm 2 LM-based Map Construction Algorithm (LMCA)

for Each incoming data $G_i(t) = \{v_i, v_k v_i\}$ **do**
  if $v_i.LM \notin \{v_j.LM\}, v_j \in V_C$ **then**
    Add v_i and $v_k v_i$ to C
    Connect v_i with w_i in C
  else
    if $v_k \notin NVS(w_i) || Len(v_k v_i) \notin \{Len(v_j v_i)\}$ where $v_j v_i \in NES(w_i)$ **then**
      Connect v_i with v_k in C
  Record current location of user i by $w_i = v_i$

However, the cost of providing an LM at each intersection may be too high, and the success of map construction exclusively depends on those local infrastructures which can offer LMs. We need an alternative way for supplying the unique information of intersections, and it should be independent from the local infrastructures.

3) The third basic condition (direction-based basic condition): an alternative option for map construction is to use the directions of entering into and leaving from an intersection. As there are no absolute location markers at all of the intersections, we need to consider two cases: (i) matching the trajectories of any two users who move along the same path; (ii) connecting the trajectories of any two users whose trajectories share the same intersection.

If at least one intersection from a user's trajectory can be correctly matched with others' data, the whole trajectory can be uniquely constructed by using the directions. Moreover, any user who walks along the same paths can also be uniquely identified and matched with each other. Suppose that two users, A and B, respectively pass the same intersection, v_a, which has a LM. The road segment walked by A is represented as $v_a v_b$, which means A moved from intersection v_a to the successive intersection v_b. The road segment $v_a v_c$, passed by B, means that B passed a road segment from intersection v_a to v_c. We assume that intersections b and c do not have LMs. If the departure directions of A and B are the same $A.Dir(v_a, d) = B.Dir(v_a, d)$, we can infer that $v_b = v_c$.

***Theorem** 1: suppose that two users, A and B, have the same LM respectively at the i^{th} record of A and at the j^{th} record of B. If the next k departure (or entering) directions of them are also the same, then the two users walk along the same paths in this period of time.*

Proof: we use $v_{A(p)}$ to represent the location of the p^{th} intersection reported by A. Since users report the entering (and leaving) direction at every intersection, the corresponding intersections of two consecutively reported directions are uniquely connected by one and only one path. We use $A(i).LM$ and $B(j).LM$ to represent the beginning LMs of A and B. We also have $A(i).LM = B(j).LM$ and $A.Dir(v_{A(i)}, d) = B.Dir(v_{B(j)}, d)$. Therefore, $v_{A(i+1)} = v_{B(j+1)}$. Assume that $v_{A(i+p)} = v_{B(j+p)}, (p \in N, 0 \leq p < k)$. Based on the fact that we can determine one and only one radial from the same source at one direction, and the fact that every intersection on the paths of A and B has been reported, we can get $A.Dir(v_{A(i+p)}, d) = B.Dir(v_{B(j+p)}, d)$. Hence, $v_{A(i+p+1)} = v_{B(j+p+1)}$. ■

Based on Theorem 1, the intersections on the paths passed by different users at different time can be matched if they have at least one joint intersection and consecutively equal turning directions. However, we also need to match the intersections that are passed by different users along different paths. In the traditional map construction problem, finding the union of several trajectories is trivial once some common vertices or common edges are obtained. However, in our problem, matching intersections from different data sets is still hard, even if we can obtain partially matching relations of the sets. The main reason for such a difference is that preserving the shapes of the trajectories in traditional methods provides a strong spatial restriction. In our system, without recording the shapes of trajectories, the central server may construct more than one isomorphous map, which all satisfy the known partial-matching relations. We use encounter information to solve the problem.

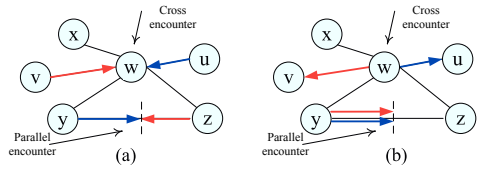


Fig. 2. Parallel and cross encounters.

There are two types of encounters. When two users are walking along the same path and encounter with each other, we call it as a *parallel encounter*, as shown in Fig. 2; if two users are walking along different paths and encounter with each other at an intersection, we name this condition as *cross encounter*, as shown in Fig. 2. Different types of encounters hold different information, as a result they should be treated separately during intersection matching. Axioms 1 and 2 give the policies for using these two types of encounters.

***Axiom** 1: if two users, A and B, have a physical parallel encounter, edge matching can be applied: the road segment, where the encounter happened, should match each other; the end points of the paths should match each other.*

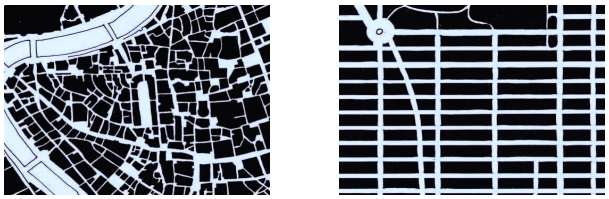
***Axiom** 2: suppose that two users, A and B, have a physical cross encounter at an intersection. Node matching can be applied: the intersections, where the encounter happened, should match with each other. Moreover, the instant speed before and after the encounter also informs the exclusive-matching relation: a different entering/leaving direction of instant speed implies that the last/next reported intersections of A and B should not be merged together.*

Algorithm 3 Direction-based Map Construction Algorithm (DMCA)

```
for Each incoming data G_i(t) = {v_i, v_k v_i} do
    if i.LM ≠ ∅ then
        if i.LM ∉ {v_j.LM}, v_j ∈ NVS(w_i) then
            Add v_i, v_k v_i to C, connect v_i with w_i
        else
            if i.Dir(w_i, d) ∉ {Dir(w_i, d)} || i.Dir(v_i, e) ∉
            {Dir(v_i, e)}, v_i ∈ C then
                Connect v_i with w_i in C
    else
        if i.Dir(w_i, d) ∉ {Dir(w_i, d)} then
            Add v_i, v_k v_i to C, connect v_i with w_i in C
    Record current location of user i by w_i = v_i
for Each incoming encounter records do
    if The corresponding locations of i and j in C do not follow
    Axioms 1 and 2 then
        Merge the two intersections together
```

Based on Theorem 1, and Axioms 1 and 2, we find the third basic condition for map construction, which is more realistic. Assume that in a close region there are N intersections, and only n ($n \leq N$) of them can provide an LM. Users are required to report the displacement from the last intersection, both entering and leaving directions at each intersection, as well as encounter information and LMs (if they have any).

The corresponding map construction algorithm of basic condition three works as follows. At the server, the incoming



(a) complex structure (b) simple structure

Fig. 3. Various structures of street networks.

data will be matched with the constructing map by using LMs, directions, and physical encounters. The server first matches the reported trajectories with the constructing map. If the server cannot match the incoming intersection with the existing ones, a new intersection will be added in the constructing map. Then, the server will use encounter information to adjust the constructing map by merging reduplicate intersections: if the virtual identities of the encounter-related intersections are different, merge the virtual identities into a unique one. We name the corresponding algorithm, under the third basic condition, as the direction-based Map Construction Algorithm (DMCA). The details of DMCA are given by Algorithm 3.

C. Feedback-based heuristic map construction algorithm

The complexity of road structures can vary greatly. Fig. 3 gives us two examples. The road structure of Fig. 3(a) and the shape of each road segment is complex, but such structure complexity also provides the uniqueness of the length of the road segment; Fig. 3(b) only has a very simple structure, however, the intersections or road segments are very common as well as similar to each other, which makes trajectory matching much harder than in Fig. 3(a).

Considering that the structure complexity of a map is a regional feature, we can regard the whole map as the composition and combination of several sub-maps whose structures are simpler and coherent. We name these sub-maps as *cells*. We use C to represent the cells, $\bigcup_{i=1}^{k} C_i = G$.

Typical map construction methods do not have an adjustment policy for different regional structure complexities. Hence, their location update strategy at the user side is unchangeable. For guaranteeing the quality of the service, most of them apply a uniform strategy that is suitable for a region with a complex structure. The obvious weakness of using a uniform update strategy is the superfluous information. If the central server can inform users of how to change their update strategy at different regions, the cost of map construction can be definitely reduced. However, in our problem, the server does not have such a general knowledge about the map.

The amount of information used in three basic conditions is different: direction-based map construction requires the largest amount of information; the length of edges and LMs are used in all of the three conditions. Moreover, the construction time under these three conditions is also different. The LM-based map construction method requires the least time, while the direction-based method costs the most. We use the different converging time of the three basic conditions to create a new map construction method that can appropriately use data.

Considering that it is not necessary to match every user's reported data in the map construction phase, we can just let users simply report the node degree of intersections and the length of paths in the beginning, as well as the LM if it is available. Gradually, the cells in basic conditions one and two can be built. The server can notify the users who are (not) in the constructed cells to begin (stop) reporting the entering and leaving directions at intersections. Hence, the cells can be gradually connected with each other. Since the notification works as feedback, which reveals the difficulty of construction in users' nearby regions, we name the above map construction process as a feedback-based heuristic map construction algorithm (FHMCA), which is shown by Algorithm. 4.

Algorithm 4 Feedback-based heuristic Map Construction Algorithm (FHMCA)

```
Setup timer, system initialization
while Timer is valid do
    for Each incoming data G_i(t) do
        if i.LM ≠ ∅ then
            Apply LMCA on the constructing map C
        else
            Apply UMCA on C
while Time out do
    for Each incoming data G_i(t) do
        if Dir(v_i, e) ≠ ∅ or Dir(v_i, d) ≠ ∅ then
            Apply DMCA on C
        else
            Localize user's position (w_i = v_i) on C
        if One of the neighbors of current intersection is not fully
        constructed then
            Inform user to begin to report directions
        else
            Inform user to stop reporting directions
```

V. Localization on the constructed map

The constructed map of FHMCA can be used in localization. Here, we introduce a new localization method by using a FHMCA constructed map, which can efficiently localize users and costs less amounts of data transmissions. This localization method shares the same idea as FHMCA: if users are located at unique regions, the server will inform them, and the users only need to report simple version of the data; as in common region, the server will require the clients to submit extra data. In general, the proposed localization method uses the encounter, intersection features, and graph matching. In our localization method, a user's position can be represented as being at an intersection or at the segment between two intersections.

A. Intersection-based localization

After the intersections connectivity map is constructed, the server can localize the users by associating them with the last passed intersection. If some applications need the users' real-time positions, the server can estimate them by estimating the departure from the last reported intersection, and the server only needs to capture two events: users' average moving speeds, and last departed intersection. The average speed can be obtained easily since the server already has both the length of road segments and the time-stamps of the reported data.

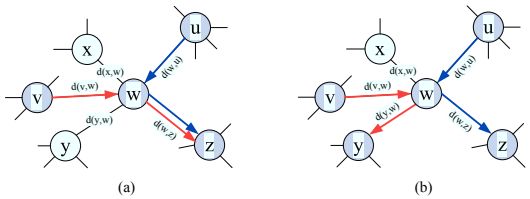


Fig. 4. 2-hop information for searching initial positions.

The idea of intersection map-based localization is that localization at a unique intersection requires less information than that at general intersection. As the server already known the uniqueness information of every intersection, we just let the server look ahead at the constructed map to check the uniqueness of nearby intersections, and then inform users when and where to report what kind of information.

For the applications, which do not require a user's real-time location, the localization can be achieved as follows. After receiving data, the server first updates the user's location. Then, the server checks the uniqueness of the current intersection's neighbors. If these neighbors can be distinguished by the node-degrees, the length of the displacement, or the LMs, the server informs the user to report regular data; however, if at least one pair of neighbors are similar, the server has to require the user to report leaving directions. As for the applications required the real-time locations, the users need to report leaving directions at each intersection. The server can estimate the positions of users by using the average speed.

B. Fast position initialization

In our proposed localization method, we do not apply any location-determined device, which can provide exact locations. As a result, there may be a long time gap from the moment a new user joined the system to the moment of being localized; if the surrounding region of users' entering positions is unique, the users can be quickly localized. For example, if all of the intersections near the entering positions have LMs, or if the lengths of paths are all different in this region, the time gap will be short. However, if the surrounding regions are very common, the server will not be able to find out the initial locations, even if the users report turning directions.

The encounter information can be used to solve the time gap problem involved with position initialization: if new users come across some others users, who have already been localized, the new users can be localized by using the locations of the old users. However, if both of them have not been localized, initial localization is still a problem.

One straightforward solution is to use graph matching on a single user's trajectory. If the user's trajectory in a period of time can be uniquely identified, the initial location will be found. However, the length of the time gap from using this method is determined by many factors, such as the speed of the user or the uniqueness of the region.

In order to quickly find the initial position, we propose the second solution. We notice that when two users encounter each other at an intersection, the central server can obtain partial 2-hop information of this intersection. Fig. 4 is an example. When users A and B meet each other at an intersection, the server can know the node degree of parts of its neighbor intersections and the distance to them. Moreover, if both of the users also report the entering and leaving directions at each intersection, the server can obtain more information, which can increase the chance of having unique features: the server may apply some graph matching algorithm on this partial subgraph and the constructed map. If the server exclusively finds one matching region from the constructed map, then both of the users are localized. Otherwise, the server will continue to add the two users' reported trajectories to the subgraph until finding only one matching result or one of them can be localized, such as encountering an old user or passing an intersection with an LM. Compared to the first solution, which only uses the data from single user, the growing speed of the subgraph in solution two is much faster; as a result, the time gap of initialization will become shorter.

VI. Data verification

There are two key features of cooperative trajectory mapping: (1) when users encounter with each other, they independently report encountering to the server; (2) any path can be passed by any user. The intuition behind our data verification is to make use of these features to detect adversaries[1].

Consider that an adversary that is physically at intersection α (loc_α) but reports that he is at loc_β. Since the adversary is not at loc_β, he cannot determine whether another user is at loc_β. The probability of guessing correctly is very small; we do not consider it. If an honest user is at loc_β, the central server can detect an inconsistency: the absent of an encounter, which ought to happen from the reported trajectories. The honest user hence acts like a *witness* that can be used to identify an adversary. We assign an honest degree to each user, if the sever find an inconsistency, both involved users will get a penalty. Over time, an adversary will be associated with more penalties, and the server will not use any data from the low-honest users. Another inconsistency can be found if lots of high-honest users can not pass a path but some less trustable users can. It is very possible that the less trust users lie.

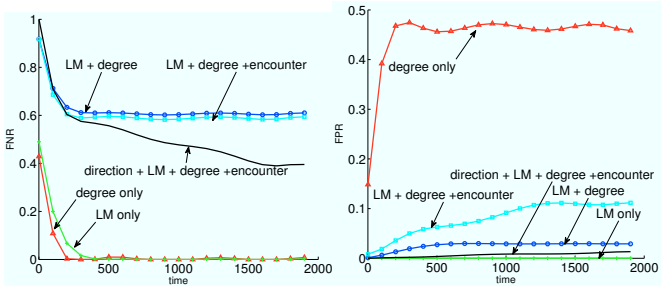
VII. Performance Analysis and Evaluation

In this section, we first describe our simulation setup and evaluation metrics. Then, the simulation results and analysis are presented.

A. Experiment setup and evaluation metric

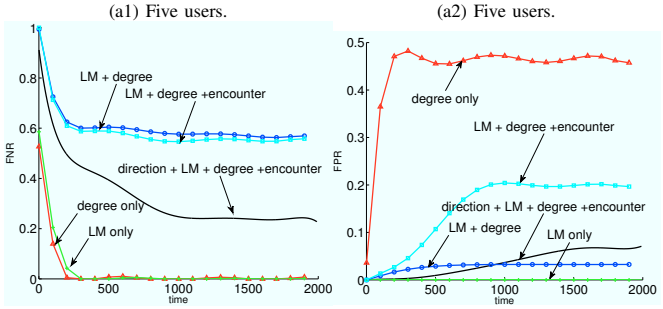
We use Matlab to perform our simulations. The simulation setup is as follows: we first synthetically generate a grid map and set the distance between neighborhood paths equally. Then, we randomly generate the initial positions of users and the speed, which varies from 1 to 10 distance units per time unit. The speed follows a uniform distribution. Then, we generate the trajectories of users. Whenever a user passes an intersection, the server may get all of or parts of the

[1]We assume that the number of honest users are greater than adversaries.



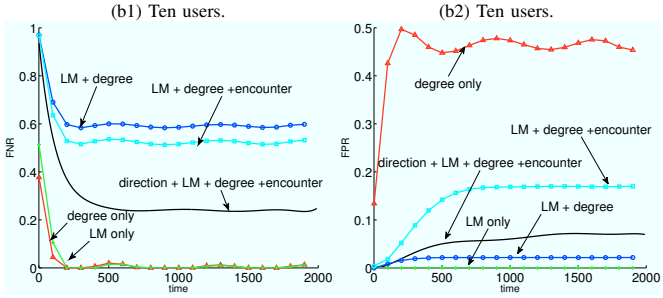
(a1) Five users.

(a2) Five users.



(b1) Ten users.

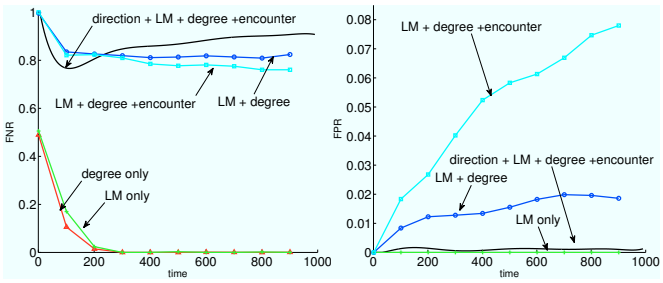
(b2) Ten users.



(c1) Fifteen users.

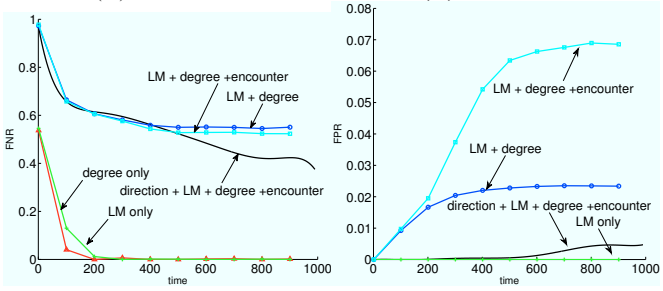
(c2) Fifteen users.

Fig. 5. The number of users vs. false negative rate and false positive rate.



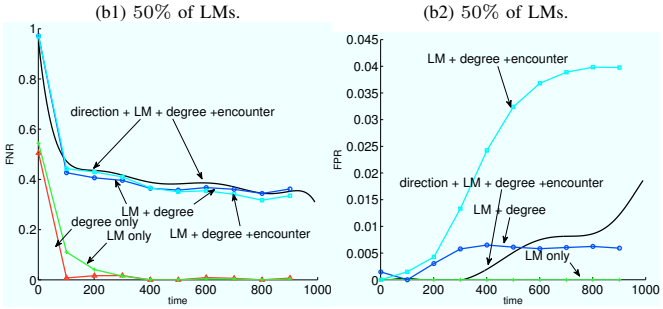
(a1) 25% of LMs.

(a2) 25% of LMs.



(b1) 50% of LMs.

(b2) 50% of LMs.



(c1) 75% of LMs.

(c2) 75% of LMs.

Fig. 6. The number of LMs vs. false negative rate and false positive rate.

follows: node degree, displacement from the last reported location, encounter, instant moving direction, LMs, entering or departure direction from an intersection. Exactly which data is used is determined by the simulation algorithm.

In this simulation, we do not consider the effects of noise. Normally, we use the 10×10 or 5×5 grid maps considering the computing time. For the consideration of generality, each data point in our graphs is the average result.

The metrics we applied are False Positive Rate (FPR) and False Negative Rate (FNR): $FPR = \frac{\text{FP}}{\text{FP+TN}}$, $FNR = \frac{\text{FN}}{\text{TP+FN}}$ where FP is the number of false positive, FN means false negative, TP stands for true positive, and TN represents true negative. We use FPR and FNR to represent the false connectivity between intersections. Consider that users may not report LMs at some intersections. During the map construction, the central server needs to provide unique identities for them. When calculating the error, to each identity in the constructed map, we first find its corresponding identity in the real intersection map, then compute the connectivity error.

B. Simulation results

In simulation, we compare five different algorithms:

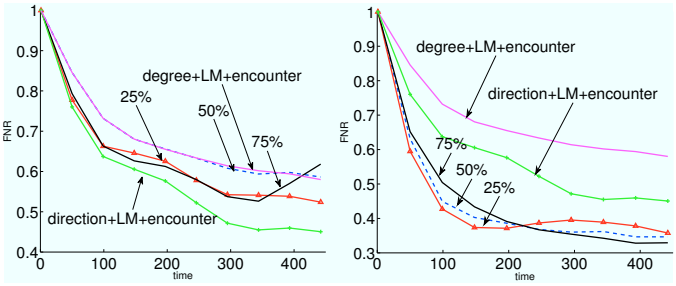
LM-based algorithm. We assume that all of the intersections can provide unique LMs, and therefore, the accuracy of this algorithm is the highest. We use the result of the LM-based algorithm as a standard.

Node degree only algorithm. Normally, if the node degree or the length of the road segment is unique, the server can correctly localize the users and build the map. However, since the grid map is the most difficult condition for map construction, and most of the intersections share the same node degree and the length of road segment, the node degree only algorithm does not suit our simulated map well.

Algorithm by applying LM and node degree. We assume that some intersections can provide LMs to users, while others cannot. Consider the accuracy of using LMs. The algorithm gives a higher priority to LM information.

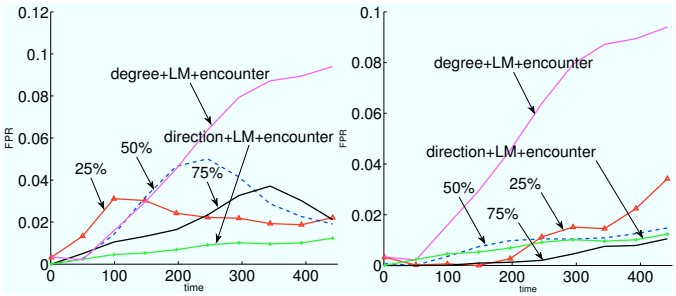
Algorithm by applying LM, node degree, and encounter. This algorithm tries to apply encounter information to improve the performance of the LM and node degree-based algorithm. We use the encounter record to find out whether two different intersections in the constructing map should be merged, or whether one intersection should be split into two.

Direction-based algorithm. The direction-based algorithm applies direction, LM, node degree, and encounters. This algorithm also uses encounter information to merge and split intersections in a constructing map. However, the converging time of this algorithm is long since the algorithm requires users



(a) Node degree followed by direction. (b) Direction followed by node degree.

Fig. 7. The impact on FNR when using different data sequences.



(a) Node degree followed by direction. (b) Direction followed by node degree.

Fig. 8. The impact on FPR when using different data sequences.

to encounter with others at each intersection at least one time. Consider that our FHMCA takes more time to converge than the direction-based algorithm and that the accuracy of both the FHMCA and the direction-based algorithm are similar, we do not draw the FHMCA curves. But, the comparison of these two algorithms is presented later.

The first tested factor is the number of users. We test FPR and FNR when the number of users are 5, 10, and 15. The observation time is set as 2000 time units, and the encounter' sensor range is 1. We randomly select 50% of the intersections to have LMs. The initial positions of users are randomly deployed. Fig. 5 shows our simulation results.

The second tested factor is the density of LMs. We test FPR and FNR when the density of LMs are 25%, 50%, and 75%. The observation time is set as 1000 time units, and the encounter' sensor range is 1. The number of users is 5, and their initial positions are also randomly deployed. Fig. 6 shows our simulation results. From the figures we can find that the number of LMs has a huge impact on both FPR and FNR. Moreover, the converge speed is faster in high LMs-density.

Figs. 7 and 8 show the relationship between the sequence of applying varying data and the accuracy of the constructed map. In Figs. 7(a) and 8(a), we let the server only use the node degree information at the very beginning, and then, direction information is applied; however, in Figs. 7(b) and 8(b), we let the server first use the direction information followed by using node degree information. From the simulation results, we can see that directly using direction information at the beginning can cause a lower FPR while using direction information after others can result in a lower FNR.

Our last tested factor is the time for transforming the data types. From Figs. 7 and 8, we advert that the time of transforming data types affects more the method that used the node degree first. Moreover, at the beginning of changing the data types, the accuracy may fluctuate for a short time, and then, the curve begins to converge again.

VIII. Conclusion

In this paper, we consider the problem of map construction in cooperative trajectory mapping. We propose a server feedback-based map construction algorithm, which can gradually construct a map without using GPS data. Considering that there are multiple kinds of sensor data available for the server at different times and that different combinations of them may have different results, we use several algorithms to maximally use that data. Extensive simulations and comparisons are made. Our future work intends to test our algorithm with real maps. In a real-world map, the features of each intersection may be more unique than our simulation grid maps.

Acknowledgment

This research was supported in part by NSF grants ECCS 1128209, CNS 1065444, CCF 1028167, CNS 0948184, and CCF 0830289.

References

[1] P. Mohan, V. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *ACM SenSys*, 2008.

[2] A. Repenning and A. Ioannidou, "Mobility agents: guiding and tracking public transportation users," in *ACM AVI*, 2006.

[3] J. Froehlich, T. Dillahunt, P. Klasnja, J. Mankoff, S. Consolvo, B. Harrison, and J. Landay, "UbiGreen: investigating a mobile tool for tracking and supporting green transportation habits," in *ACM CHI*, 2009.

[4] S. Carmien, M. Dawe, G. Fischer, A. Gorman, A. Kintsch, J. Sullivan, and F. James, "Socio-technical environments supporting people with cognitive disabilities using public transportation," *ACM TOCHI*, 2005.

[5] I. Constandache, R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *IEEE INFOCOM*, 2010.

[6] P. Bahl and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *IEEE INFOCOM*, 2000.

[7] Y. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm, "Accuracy characterization for metropolitan-scale Wi-Fi localization," in *ACM MobiSys*, 2005.

[8] I. Constandache, X. Bao, M. Azizyan, and R. Choudhury, "Did you see Bob?: human localization using mobile phones," in *ACM MobiCom*, 2010.

[9] A. Bar-Noy, I. Kessler, and M. Sidi, "Mobile users: To update or not to update?" *Wireless Networks*, 1995.

[10] P. Pesti, L. Liu, B. Bamba, A. Iyenga, and M. Weber, "RoadTrack: scaling location updates for mobile clients on road networks with query awareness," *VLDB Endowment*, 2010.

[11] S.K. Sen, A. Bhattacharya, and S. Das, "A selective location update strategy for PCS users," *Wireless Networks*, 1999.

[12] V. Wong, and V. Leung, "An adaptive distance-based location update algorithm for next-generation PCS networks," *IEEE JSAC*, 2001.

[13] A. Civilis, C.S. Jensen, and S. Pakalnis, "Techniques for efficient road-network-based tracking of moving objects," *IEEE TKDE*, 2005.

[14] O. Wolfson and H. Yin, "Accuracy and resource consumption in tracking and location prediction," *SSTD*, 2003.

[15] J. Yick, B. Mukherjee, and D. Ghosal, "Analysis of a prediction-based mobility adaptive tracking algorithm," in *Broadnets*, 2005.

[16] A. Civilis, C.S. Jensen, J. Nenortait, and S. Pakalnis, "Efficient tracking of moving objects with precision guarantees," *Mobiquitous*, 2004.

[17] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *ACM SenSys*, 2009.

[18] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson, "Cooperative transit tracking using smart-phones," in *ACM SenSys*, 2010.