Project Report on WLAN RSS-based Indoor Localization and Tracking Algorithms using Field Data

Wenhao Wu
Kun Wang

03/22/2013

Abstract

The proliferation of mobile computing devices and local-area wireless networks has fostered a growing interest in location-aware systems and services, which are enabled by locating and tracking of a mobile user. In this project, we investigated several algorithms for indoor locating and tracking. Firstly, we simulated a radio frequency (RF) based system called RADAR [1], which consists of two basic algorithms for static locating and a Viterbi-like algorithm for tracking. However, there are some shortcomings of the Viterbi-like algorithm in RADAR, so we proposed a more advanced algorithm – HMM-based algorithm. This algorithm makes use of the user's speed information and the floor plan constraint, providing an accurate localization and tracking performance.

1. Introduction

In recent years, wireless indoor locating and tracking received considerable attentions in academia and a bunch of localization algorithms have been proposed based on various measuring principles and systems.

Typically, schemes for indoor locating and tracking can be categorized into three classes [2] – triangulation, scene analysis

and proximity, and the signal metrics used in these schemes could be time of arrival (TOA), received signal strength (RSS) and so forth. Moreover, as for the systems utilized, one approach is to develop new systems and another is to use existing infrastructures, such as wireless local area networks (WLAN).

Generally speaking, there are four types of locating and tracking systems. The first is IR-based. It uses active badge and a lot of sensors that are placed at known locations. While this system provides accurate location information, it has high installation and maintenance cost and it performs poorly when there is direct sunlight. The second is RF-based, which has advantage of being implemented purely in software and being easily deployable over a standard WLAN. The third one is cellular-based. It measures metrics such as signal attenuation, time of arrival, etc. However, it is only promising in outdoor environments because of the multiple reflections in indoor environment, and also it is expensive to implement. Other systems include ultrasound, magnetic field, etc. but they are rarely used.

In this course project, we specifically focus on indoor localization algorithms that are based on the information of RSS in WLAN. Instead of using assumptions only, we adopted empirical data[1]

[1] The dataset is obtained online from Dartmouth Crawdad “Dataset of received signal strength indication

from field test to our simulations. We simulated several algorithms, including nearest neighbor in signal space (NNSS) and its variant NNSS-AVG, which are used for static locating, and a Viterbi-like algorithm, which is used for tracking. Based on these algorithms, we proposed a non-stationary HMM-based algorithm of superior performance. The HMM-based algorithm makes use of the user's speed information, acquired form inertial sensors of mobile devices, by modeling the user's walking pattern as a HMM process. The transition probability depends on the user's speed and the floor plan constraint. We derived a Viterbi-like algorithm to solve this HMM model while keeping the computational complexity rather low. Simulation results demonstrate the effectiveness of this algorithm.

The rest of our report is organized as follows. In Section 2, we describe the RADAR system and HMM-based system, especially the modeling of HMM-based system, along with the corresponding algorithms. Simulation results are shown in Section 3, and we draw the conclusion in Section 4. Moreover, Matlab functions and scripts for our simulation are appended in the end.

2. Algorithm description

2.1 RADAR system

(RSSI) collected from within an indoor office building". (http://crawdad.cs.dartmouth.edu/)

2.1.1 Nearest Neighbor in Signal Space

The RADAR system is built on top of the standard wireless LAN technology. Several access points are located on the floor to cover the area of interest. The mobile user uses his/her mobile device to communicate with these access points.

The fundamental idea behind RADAR is quite intuitive. In an RF network, the received signal is strongest when the receiver is close to the AP and weakest when it is far away, so the received signal strength is a function of the receiver's (mobile user's) location. Consequently, it provides a means for inferring the user's position, namely, the trend in signal strength can be exploited by the system to estimate the mobile's location.

To implement RADAR, it involves two phases. The first phase is offline phase, in which a Radio Map of the building is created. A Radio Map is essentially a database of locations in the building and the measured signal strength of the beacons emanating from the APs as recorded at these locations. Therefore, an typical entry in the Radio Map may look like ($x, y, d, ss_{i\ (i=1...n)}$), where ($x, y$) is the physical coordinates of the location and d is the direction that the mobile user is facing towards, and where ss_i is the signal strength of the beacon signal emanating from the i^{th} AP. For the Radio Map

we used, there are five APs, so the Radio Map contains entries that look like ($x, y, d, ss_1, ss_2, ss_3, ss_4, ss_5$).

The second phase is to locate the position of the mobile user in real- time. The mobile measures the signal strength from each AP and then it searches through the Radio Map database to determine the signal strength tuple that best matches the signal strengths it has measured. The system estimates the location associated with the best-matching signal strength tuple to be the location of the mobile user. The basic algorithm for searching is called *nearest neighbor(s) in signal space* (*NNSS*) [3]. The *NNSS* algorithm computes the *Euclidean distance* (in signal space) between each SS tuple in the Radio Map ($ss_1, ss_2, ss_3, ss_4, ss_5$) and the measured *SS* tuple ($ss'_1, ss'_2, ss'_3, ss'_4, ss'_5$). It then picks the SS tuple that minimizes the distance and declares the corresponding physical coordinates as its estimate of the user's location.

2.1.2 *K*-Nearest Neighbors in Signal Space

One variant of the basic NNSS algorithm is *NNSS-AVG.* The intuition here is that in case there is more than one SS tuple in the Radio Map that is "close" to measured SS tuple, there is little reason to pick just the closest one and discard others that are almost as close. So the NNSS-AVG algorithm picks a small number

of closely matching tuples and averages their physical location to obtain an estimate of the user’s location. Often this results in an estimate that is better than any individual tuple.

2.1.3 Viterbi-like Algorithm

The aforementioned two algorithms basically focus on static user localization. The NNSS algorithm and its variant used for this purpose do not consider location information from the past. The idea behind continuous user tracking is precisely to use information from the past to come up with a better guess of a user’s location. The intuition is that since physical constraints preclude a user from "jumping about" over large distances at random, the user’s location at a given time instant is likely to be near that at the previous time instant. So, by tracking the user continuously, we complement signal strength information with the physical contiguity constraint to potentially improve the accuracy of location determination [1].

The Viterbi-like algorithm for continuous user tracking works as follows. Each time a signal strength tuple is obtained by the mobile user, an NNSS Radio map search is done to determine the *k* nearest neighbors in signal space (*k*-NNSS). A history of depth *h* of such *k*-NNSS sets is maintained. The collection of these *h* *k*-NNSS sets can be visualized in figure 1. There are edges only between

vertices contained within consecutive sets. Each edge is assigned a weight to model the likelihood of the user transitioning (in successive time instances) between the locations represented by the two endpoints of the edge. The larger the weight is, the less likely is the transition. A very simple metric is used – the Euclidean distance between the two physical locations – as the weight [1].

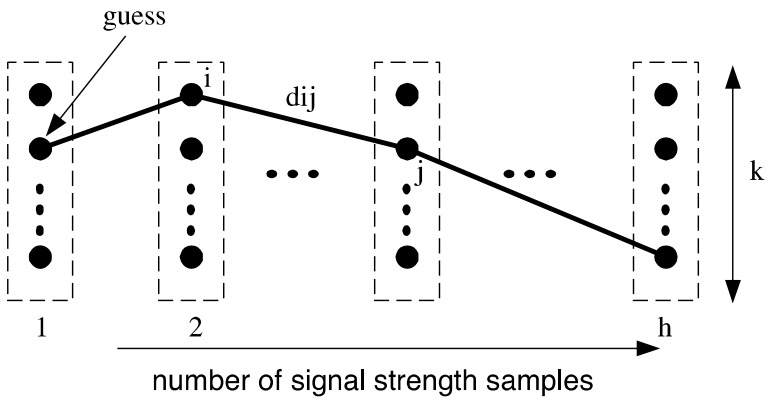

Fig. 1 A depiction of the state maintained by the Viterbi-like algorithm

Each time the history vector is updated with the addition of the most recent *k*-NNSS set and the deletion of the oldest set, the shortest path between the vertices in the oldest and the newest sets is computed. This shortest path can be viewed as representing the "most likely" trajectory of the mobile user. Once the shortest path is determined, we guess the user's location to be the point at the start of the path (Fig.1). This procedure implies that there is a lag of *h* signal strength samples between when a user is at a location and when the system guesses the user's location, whereas

we can neglect this effect as long as the sampling interval is short and the memory length *h* is not too long.

2.2 HMM-based system

In this section we firstly describe the RSSI and user's walking pattern model. Based on these two models, we formulate the localization and tracking problem as a HMM process. Then we proposed our Viterbi-like algorithm to solve this problem

2.2.1 RSSI model

For simplicity, we assume that the RSSI vector at each measurement position follow Gaussian distribution

$$\mathbf{r}_i \sim N\left(\mathbf{m}_i, \mathbf{K}_i\right)$$

where $\mathbf{m}_i$ and $\mathbf{K}_i$ can be computed offline with the dataset.

2.2.2 Walking model with floor plan constraint

Assuming that the user's position is a non-stationary Markov process of which the transition probability depends on the user's instant speed v . The transition probability from position i to position j is

$$p_{ij} \propto \exp\left(-\frac{\left(d_{ij}-vT_S\right)^2}{\left(vT_S+\varepsilon\right)^2}\right)$$

where T_S is the sampling period, ε is a small perturbation coefficient. d_{ij} is the walk distance between position i and position j, which takes into consideration of the floor plan constraint. d_{ij} can be computed offline using the Floyd-Warshall algorithm, v can be estimated on-line with the sensor embedded on mobile devices (such as the inertial sensor which can count the step).

2.2.3 Viterbi-like algorithm

Combining the RSSI model and the walking model, the localization problem can be formulated as a non-stationary Hidden Markov process. Denote the hidden position $\mathbf{i}_t = \left(i(t), i(t-1), ..., i(1)\right)^T$, the observed RSSI $\mathbf{R}_t = \left(\mathbf{r}(t), \mathbf{r}(t-1), ..., \mathbf{r}(1)\right)^T$. the maximum a priori estimation of $\mathbf{i}_t$ at time t is

$$\hat{\mathbf{i}}_t = \arg\max_{\mathbf{i}_t}\left(f\left(\mathbf{R}_t, \mathbf{i}_t\right)\right) = \arg\max_{\mathbf{i}_t}\left(\ln\left(f\left(\mathbf{R}_t, \mathbf{i}_t\right)\right)\right)$$

where $f(\mathbf{R}_t, \mathbf{i}_t)$ is the joint pdf of observing $\mathbf{R}_t$ and hidden position $\mathbf{i}_t$. Given the above RSSI model and walking model, we have

$$\begin{aligned}\ln\left(f\left(\mathbf{R}_t, \mathbf{i}_t\right)\right) = & \sum_{u=1}^{t-1} p_{i(u)i(u+1)} \\ & - \frac{1}{2}\sum_{u=1}^{t}\left(\left(\mathbf{r}(u) - \mathbf{m}_{i(u)}\right)^T \mathbf{K}_{i(u)}^{-1}\left(\mathbf{r}(u) - \mathbf{m}_{i(u)}\right) + \ln\left(\det\left(\mathbf{K}_{i(u)}\right)\right)\right) \\ & + C(t)\end{aligned}$$

where $C(t)$ is a constant number irrelevant to the observation or hidden position. Therefore

$$\ln\left(f\left(\mathbf{R}_t,\mathbf{i}_t\right)\right)=\ln\left(f\left(\mathbf{R}_{t-1},\mathbf{i}_{t-1}\right)\right)+p_{i(t-1)i(t)}-\frac{1}{2}\left[\left(\mathbf{r}(t)-\mathbf{m}_{i(t)}\right)^T\mathbf{K}_{i(t)}^{-1}\left(\mathbf{r}(t)-\mathbf{m}_{i(t)}\right)-\ln\left(\det\left(\mathbf{K}_{i(t)}\right)\right)\right]+C$$

where C is a constant. Then it is straightforward to apply the Viterbi algorithm to solving this problem in real time.

To alleviate the computation complexity of the Viterbi algorithm which is proportional to the square of the number of possible positions, we first find k most possible positions based on the distance between the mean RSSI and the observed RSSI, and then use the Viterbi-like algorithm to solve this reduced problem.

3. Simulation results

3.1 Data import and preprocessing

Our simulation is based on the RSSI data measured in real building [4], The RSSI data was measured at 179 locations. At each location, RSSI is measured with an omni-directional antenna at four directions (up, down, left and right). On each direction, the measurement takes place during the transmitting of around 500 packets (some measurements are lost apparently). The total number of measured RSSI is 343458.

The first step of our simulation is data importing and preprocessing. This step includes

- RSSI data file import; Estimation of the RSSI mean $\mathbf{m}_i$ and covariance $\mathbf{K}_i$ of the at each position;

- Floor plan import (as the location of the walls); Computing the walk distance and the shortest path between each pair of positions with Floyd-Warshall algorithm.

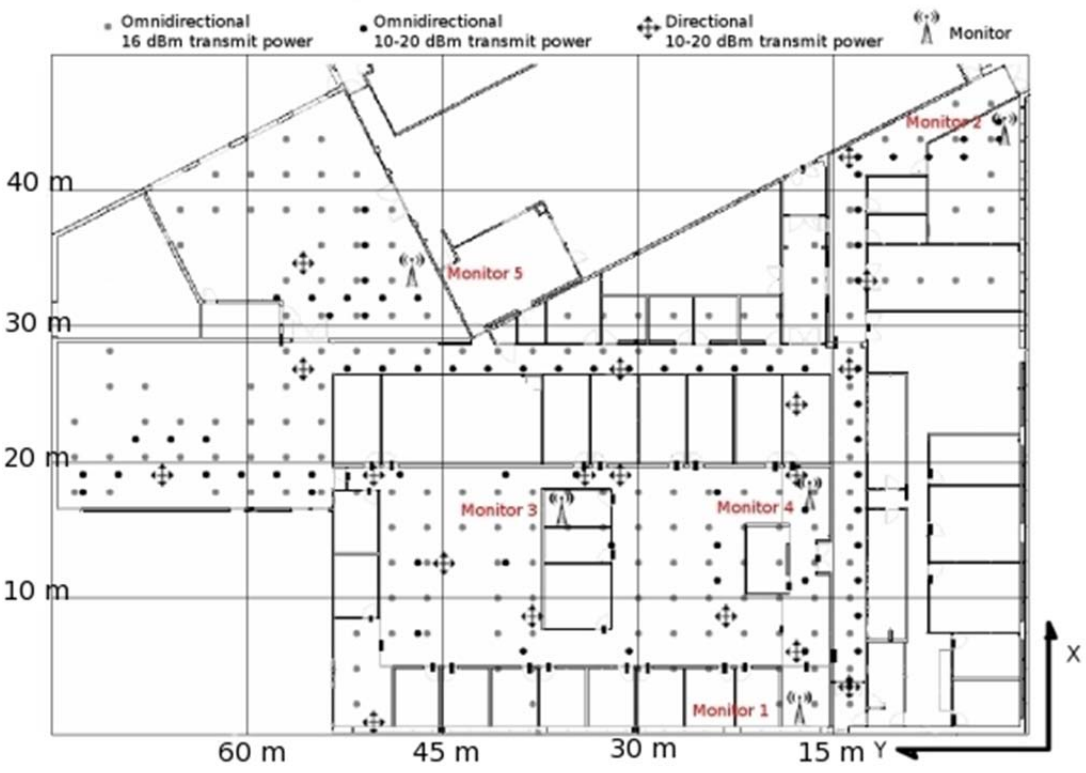

Fig. 2 Original floor plan

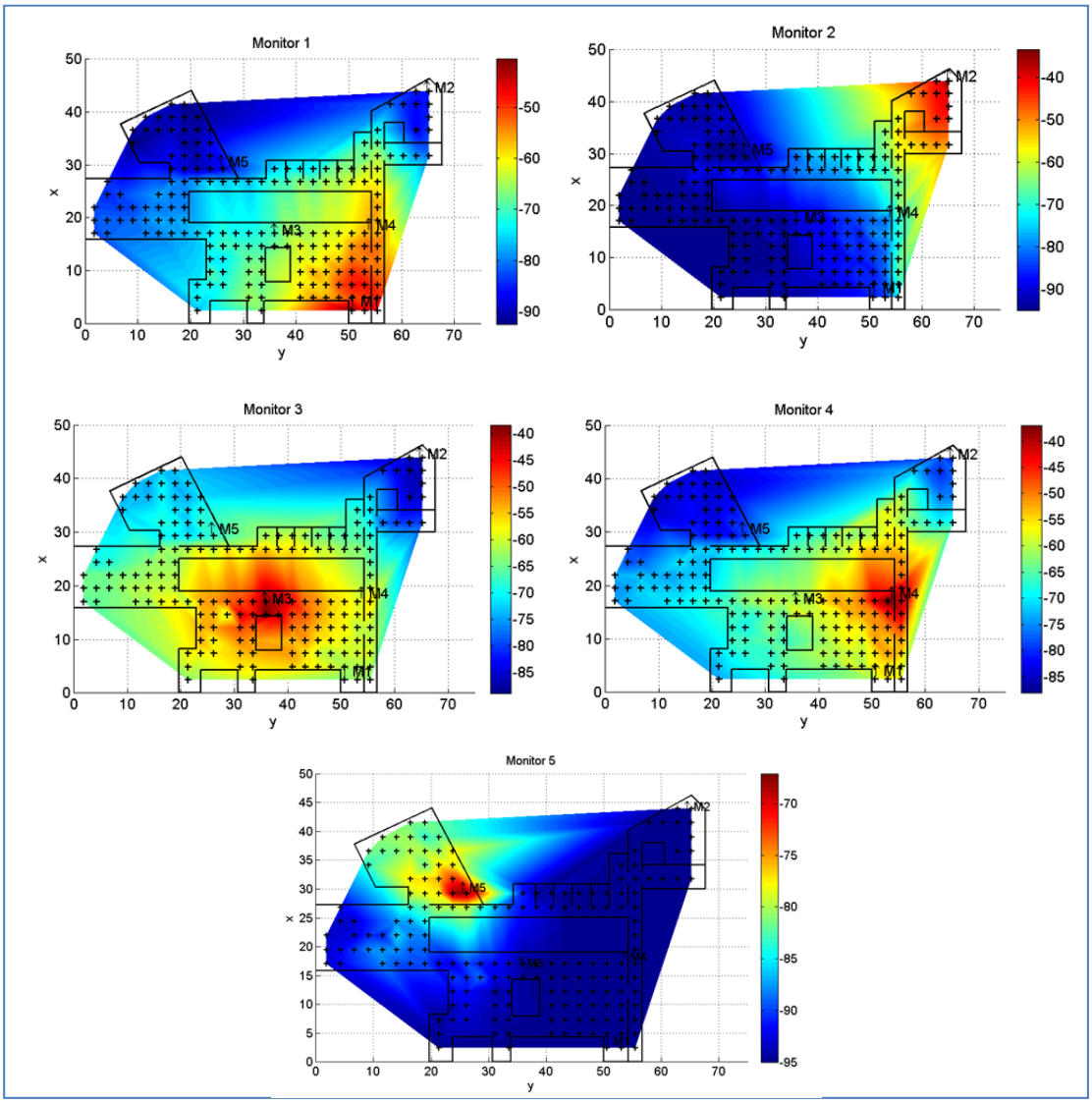

Fig. 3 Mean RSSI measured from the 5 monitors

3.2 Generate simulation input data

In order to run our algorithm on the dataset, we need to generate simulation input data based on the dataset. This consists of two parts:

- Generating user movement path. In our simulation, we specify the user's route on this floor plan by specifying a series of turning points (on measurement positions), then

the total route is determined as the concatenation of the shortest path between each two neighboring turning points. The user's instant speed information is defined in a speed profile variable. Then we sample along this route with period T_S to get the user's actual location and speed at each sampling point.

- Generating RSSI data along the path. For each sampling point on the user's path, we first find the 4 measurement positions with the shortest walk distance from this point. For each of the 4 measurement positions, we randomly select a RSSI vector from the dataset. We then compute the weighted average of the 4 selected RSSI vectors based on their walk distance from the sampling point as the RSSI observed.

3.3 Results of the RADAR algorithm

Firstly we demonstrate the localization results of both NNSS and k-NNSS version of the RADAR algorithm. As expected, the performance of k-NNSS is better than NNSS. Then the tracking performance of the k-NNSS Viterbi-like RADAR algorithm is illustrated in Fig. 5. We can see that RADAR algorithm provide

rather accurate estimation when the user is walking at a constant speed.

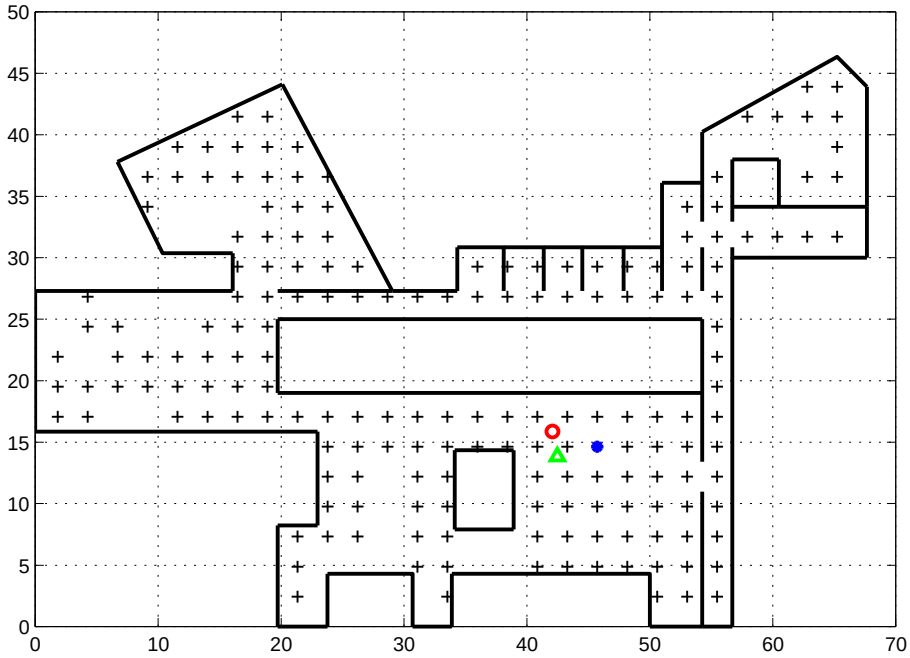

Fig. 4 NNSS and *k*-NNSS (red: actual location; blue: NNSS; green: *k*-NNSS)

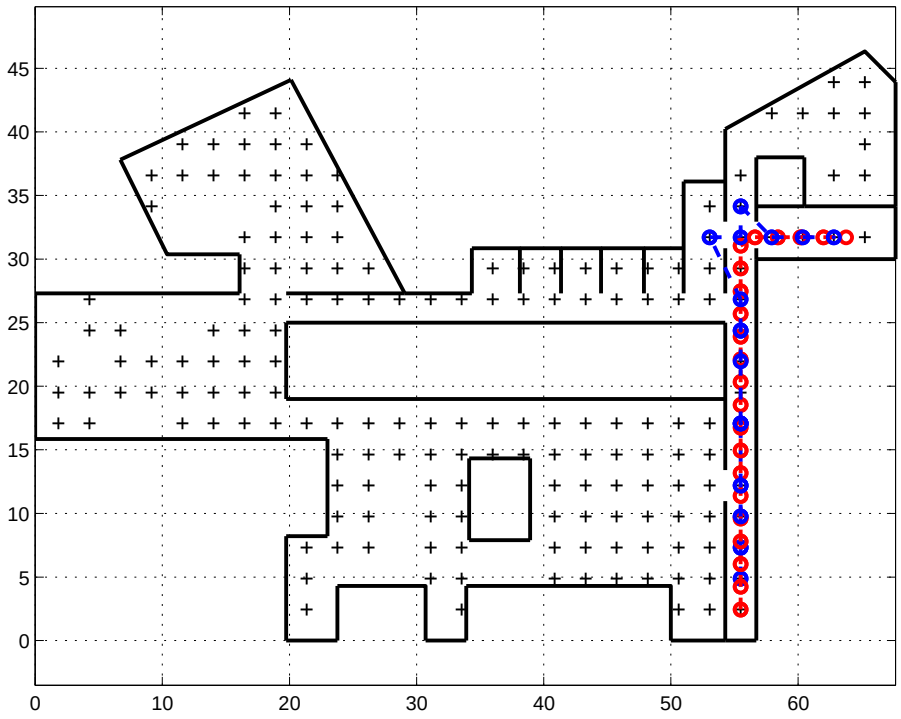

Fig. 5 Viterbi-like algorithm for continuous tracking

3.4 Results of the HMM-based algorithm

As we can see from Fig. 6, our HMM-based algorithm provides rather good localization and tracking result even though the user is walking in varying speed. One interesting phenomenon that, similar to the Viterbi-like algorithm for RADAR system, the localization is best in the corridor but not as good in the middle of the room. This is partly due to the fact that the RSSI characteristics are less distinguishable within one room than between different rooms, which we will demonstrate with spectral graph partitioning (SGP) technique [5] in the next section.

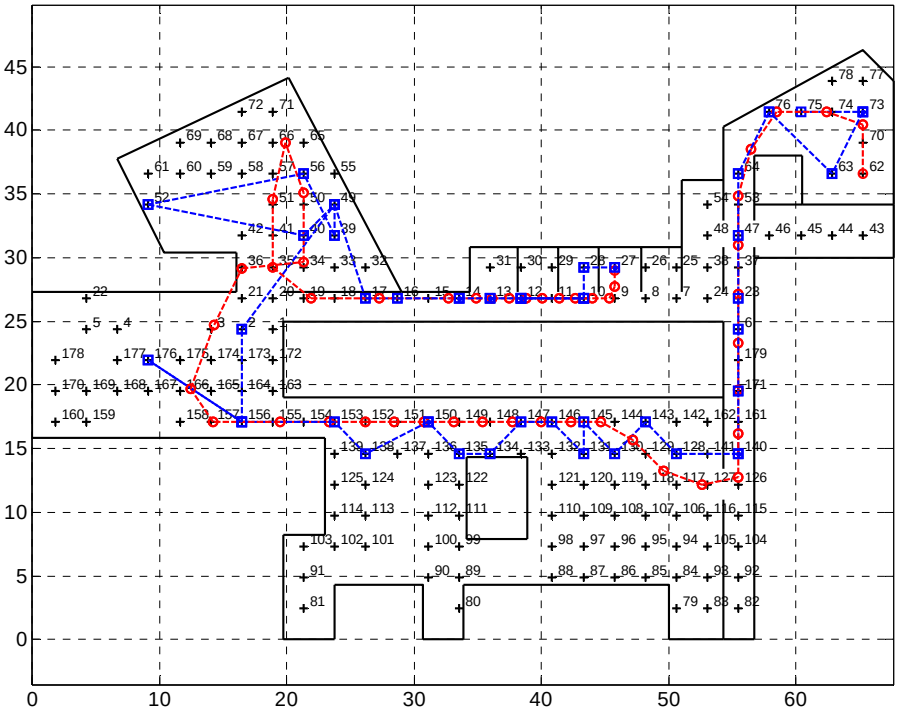

Fig. 6 HMM-based Viterbi-like algorithm (red: actual path, blue: estimated path)

3.5 Spectral graph partitioning based on RSSI and spatial proximity

We define the weighted adjacency matrix to indicate the RSSI proximity and spatial proximity between two measurement points:

$$w_{ij} = \begin{cases} \exp\left(-\frac{\left\| \mathbf{m}_i - \mathbf{m}_j \right\|^2}{\hat{\sigma}_m^2} - \frac{d_{ij}^2}{\hat{\sigma}_x^2} \right) & d_{ij} < d_0 \\ 0 & otherwise \end{cases}$$

where $\hat{\sigma}_m^2$ and $\hat{\sigma}_x^2$ are the trace of the empirical variance matrix of the mean RSSI (over all measurement positions) and that of the location of the measurement positions respectively. The SGP technique will cluster measurement positions with similar mean RSSI and close to each other. The result is shown in Fig. 7 and Fig. 8 As we can see that the floor plan is clustered into “rooms” which resembles the real room a lot. This verifies our argument that RSSI characteristics are more indistinguishable within a single room.

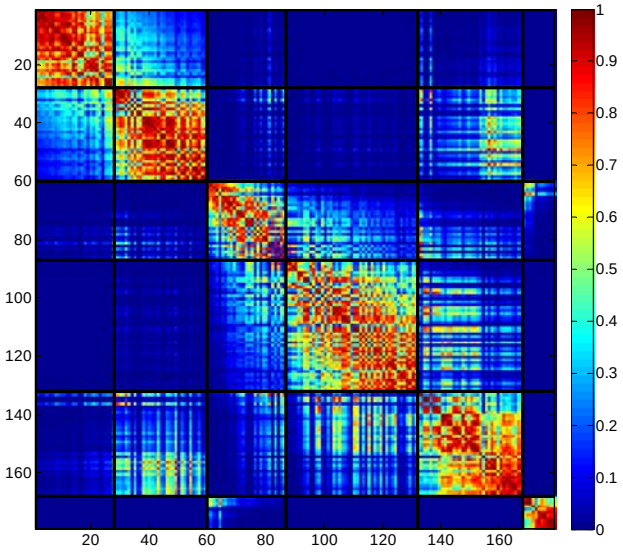

Fig. 7 Sorted weighted adjacency matrix (2-way SGP)

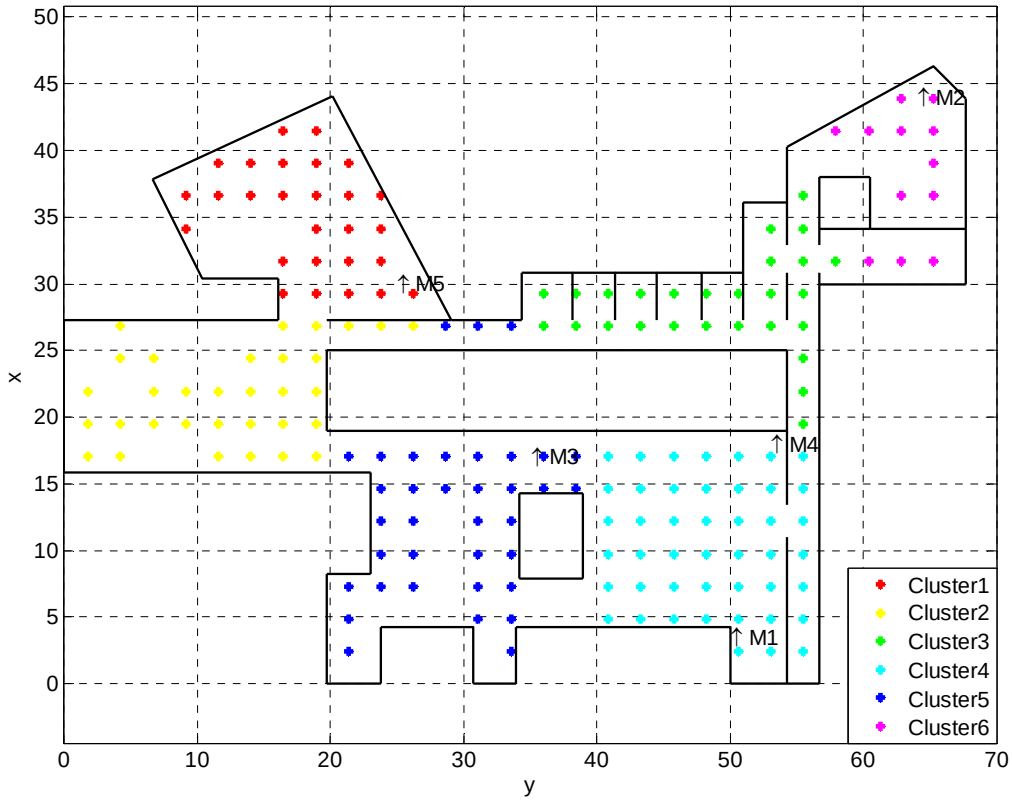

Fig. 8 Partitioned “rooms” (2-way SGP)

4. Conclusion

In this project, we proposed a novel HMM-based localization and tracking algorithm which makes use of more information than the popular RADAR algorithm. Simulation results demonstrate that with floor plan constraints and user’s walk speed information, our algorithm provides good estimation of the user’s position and path. In our future work we plan to make use of information from other sensors such as the compass to further enhance the performance. We are also considering applying the crowdsourcing technique [6] to get rid of the tedious off-line measurement phase.

References

[1] P. Bahl, V.N. Padmanabhan, A. Balachandran, Enhancement to the RADAR user location and tracking system, Microsoft Research Technical Report MSR-TR-2000-12, February 2000.

[2] H. Liu, H. Darabi, P. Banerjee and J. Liu, “Survey of Wireless Indoor Positioning Techniques and Systems”, *IEEE Trans. on Systems, Man, and Cybernetcs, Part C: Applications and Reviews*, vol. 37, pp. 1067-1080, Nov. 2007.

[3] Bahl, P.; Padmanabhan, V.N.; , "RADAR: an in-building RF-based user location and tracking system ," INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE , vol.2, no., pp.775-784 vol.2, 2000.

[4] CRAWDAD metadata: cu/rssi (v. 2009-05-28) : http://www.crawdad.org/meta.php?name=cu/rssi

[5] Shi, J.; Malik, J., "Normalized cuts and image segmentation," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.22, no.8, pp.888,905, Aug 2000

[6] Chenshu Wu; Zheng Yang; Yunhao Liu; Wei Xi; , "WILL: Wireless indoor localization without site survey," INFOCOM, 2012 Proceedings IEEE , vol., no., pp.64-72, 25-30 March 2012.

Appendix: Content of the Code Files and Resource Files

1. Data input / preprocessing

1.1 Matlab script file

- ‘FloorPlanInput.m’: Input manually specified floor plan.
- ‘RssiDataFileInputt.m’: Read the text file containing RSSI dataset from Crawdad.
- ‘PreProcess.m’: Compute RSSI statistics and the floor plan graph characteristics.

1.2 Matlab function file

- ‘GetWalkDistance.m’: Floyd-Warshall algorithm. Find the walk distance between any 2 measurement positions.
- ‘ReconPath.m’: Reconstruct the shortest path between any 2 measurement positions with the next matrix.
- ‘IsCross.m’: Test whether 2 pieces of lines cross each other.

2. Simulation data generation

2.1 Matlab function file

- ‘ModifyRoute.m’: Modify a user defined route to make sure that any 2 neighboring turning points are not separated by walls.

- ‘GenSpeedProfile.m’: Given the user’s speed range and number of speed changes, randomly generate a user’s speed profile along the specified route.
- ‘GenRouteSample.m’: Get the user’s real time location at each sampling instance along this road.
- ‘GenRssiSample.m’: Generate a set of RSSI at each sampling point along the route.
- ‘FindPosNeighbor.m’: For the input location, Find n nearest measurement positions that have line of sight from this location.
- ‘GenRssi.m’: Randomly select n measurements from the dataset for the n positions and compute the weighted average.

3. Localization algorithm

3.1 Matlab script file

- ‘demo_NNSS.m’: Demonstrate the localization result of the NNSS RADAR and k-NNSS RADAR.
- ‘demo_viterbi_tracking.m’: Demonstrate the tracking performance of the RADAR algorithm.
- ‘HMMbasedAlg.m’: Simulate the real time action of the

HMM-based algorithm.

3.2 Matlab function file

- ‘k_NNSS.m’: k_NNSS algorithm, interpolation using k neighbors.
- ‘viterbi_like.m’: Viterbi-like algorithm with k possible states and depth h.
- ‘GetkNear.m’: Coarsely get k nearest neighbors in signal strength space as the possible current states.
- ‘GetDeltaLLR.m’: Compute the log-likelihood part of the cost function increment.
- ‘GetPosCurrent.m’: In HMM-based algorithm, update the most likely path so far based on non-stationary transition probability and the observed RSSI.

4. Spectral graph partitioning

4.1 Matlab script file

- ‘SGP.m’: Compute the weighted adjacency matrix. Demonstrate spectral graph partitioning based on RSSI proximity and spacial proximity..

4.2 Matlab function file

- ‘BiPartition.m’: 2-way spectral graph partitioning algorithm.

5. GUI demo

- ‘Localization.m’: Demonstrate the real time action of the HMM-based localization and tracking algorithm.
- ‘Localization.fig’: GUI layout.

6. Others

- ‘PlotFloorPlan.m’: Plot the floor plan and the measurement positions.

7. Data file

- ‘omni_16dbm.txt’: The RSSI dataset text file from Crawdad.
- ‘omni_16dbm’: The RSSI dataset saved as Matlab resource file, generated by ‘RssiDataFileInputt.m’.
- ‘Data.mat’: Global data file, contains struct variable data whose member variables include all intermediate variable needed by all scripts and functions.