2012 32nd IEEE International Conference on Distributed Computing Systems

FindingHuMo: Real-Time Tracking of Motion Trajectories from Anonymous Binary Sensing in Smart Environments

Debraj De† Wen-Zhan Song† Mingsen Xu†
Cheng-Liang Wang‡ Diane Cook⋆ Xiaoming Huo◇

†Sensorweb Research Laboratory, Department of Computer Science, Georgia State University
‡Department of Computer Science, Chongqing University
⋆School of Electrical Engineering and Computer Science, Washington State University
◇School of Industrial and Systems Engineering, Georgia Institute of Technology
†dde1@student.gsu.edu, †wsong@gsu.edu, †mxu4@student.gsu.edu,
‡wangcl@cqu.edu.cn, ⋆djcook@wsu.edu, ◇xiaoming@isye.gatech.edu

***Abstract*—In this paper we have proposed and designed *FindingHuMo* (*Finding Human Motion*), a real-time user tracking system for Smart Environments. *FindingHuMo* can perform device-free tracking of multiple (unknown and variable number of) users in the Hallway Environments, just from non-invasive and anonymous (not user specific) binary motion sensor data stream. The significance of our designed system are as follows: (a) fast tracking of individual targets from binary motion datastream from a static wireless sensor network in the infrastructure. This needs to resolve unreliable node sequences, system noise and path ambiguity; (b) Scaling for multi-user tracking where user motion trajectories may crossover with each other in all possible ways. This needs to resolve path ambiguity to isolate overlapping trajectories; *FindingHumo* applies the following techniques on the collected motion datastream: (i) a proposed motion data driven adaptive order Hidden Markov Model with Viterbi decoding (called *Adaptive-HMM*), and then (ii) an innovative path disambiguation algorithm (called *CPDA*). Using this methodology the system accurately detects and isolates motion trajectories of individual users. The system performance is illustrated with results from real-time system deployment experience in a Smart Environment.**

***Keywords*-Human localization, Tracking, Wireless Sensor Networks, Smart Environments, Hidden Markov Model, binary motion sensor**

I. Introduction

Ubiquitous computing, particularly its adoption in Smart Environments (e.g. [1], [2]), is gaining momentum. Smart Environments are equipped with sensors which keep tracking the movements of users, who can for example be residents in a smart home, or employees in a smart workplace. Modeling the behaviors of users is a key step in developing particular applications in a Smart Environment. Identification and tracking the trajectories of users is the first step towards modeling. In many applications, e.g., in a smart workplace, a smart clinic, or a smart home, users may not want to reveal their identity all the time. In addition, the cost of sensors and communication device may drive designers to choose binary sensors that are relatively cost effective and more likely to be acceptable by general users. The binary sensors (e.g., a binary proximity based sensor, or a motion detector) only generates binary valued times series. This poses a challenge to identify and track user trajectories.

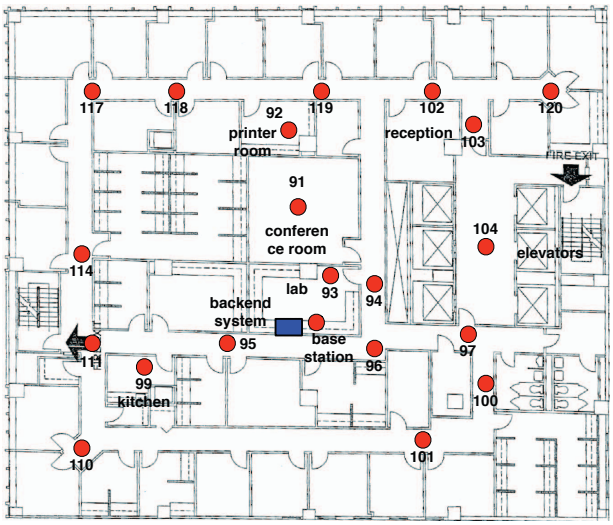


Fig. 1. Motion sensor network deployment in a smart workplace environment. The sensor node position and node ID's are shown.

As discussed, tracking mobile users in Smart Environments has utilization in many effective applications or services, such as: data delivery to mobile users ([13]); mobile and social localization ([7]); smart wireless healthcare ([27]) etc. In addition user tracking has application to study the working culture of a workplace [18]. A smart workplace environment, used in our work for user tracking, is shown in Figure 1. This has been used throughout the paper for illustrating the algorithms and system design.

Our research is partially supported by NSF-CNS-1066391, NSF-CNS-0914371, NSF-CPS-1135814 and NSF-CDI-1125165.

1063-6927/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDCS.2012.76

Precise and dynamic user tracking is a critically important as well as a challenging problem especially in multi-user crowded environment. It is important to note that in user tracking for all these services, exact co-ordinate based localization is not always a requirement ([7]). In addition GPS signals and other co-ordinate based methods are also not always effective in indoor environments. The information necessary in such user tracking is some logical location ([7]), the position of the mobile users with respect to reference nodes or points in the environment. Achieving this becomes hard due to requirements like: (i) ubiquitous applicability (therefore use of limited and cheaper sensing methods), and (ii) user privacies (use of non-invasive sensors), which impose serious practical constraints on using expensive and invasive sensors. It is important to note that the designed system doesn't use any body worn device or phone for the purpose of tracking. This is because user data privacy is a practical concern related to usage of sensor data from wearable devices ([9]). In the existing literature either more sophisticated invasive and expensive sensors are used ([4], [11]), or some theoretical geometric models for sensing and tracking are followed ([5], [26]).

The existing multi-trajectory tracking algorithms mainly deal either with tracking from imperfect sensor measurement data, or with target identity management from overlapping data. Although related, the two problems so far have been studied independently. So in this paper we propose a new approach to deal with them jointly. Our motivation in this paper is solving two main challenges: (i) user specific motion tracking just from anonymous binary motion sensor data (binary motion sensors generate binary 0 or 1 samples, denoting no-motion and motion respectively), (ii) simultaneous tracking of multiple (unknown and variable number of) users in crowded environment where motion trajectories can overlap or crossover in all possible ways. Our designed system *FindingHuMo* (*Finding Hu*man *Mo*tion) does not rely on meticulous calibration, war-driving, GPS localization, or any form of fixed reference frame. The main contributions of our work are as follows: (a) design of a novel approach for scalable and real-time tracking of multiple targets from just anonymous binary motion data. This includes: (i) proposal of a motion activity context driven adaptive order Hidden Markov Model and Viterbi decoding (*Adaptive-HMM* algorithm), and (ii) an innovative path disambiguation algorithm (called *CPDA*). *Adaptive-HMM* removes the system noise and ambiguity in smaller time window using effect of hidden states, while *CPDA* removes path ambiguity in a larger time window by applying constraints and inference on user interaction based graph. (b) Complete system design and performance evaluation in a real-time smart environment (environment is shown in Figure 1).

The rest of the paper is organized as follows. In section II we have discussed the related works in the literature. Next in section III we have described the proposed system *FindingHuMo* in details. In section IV we have done the performance evaluation in real-time system deployed. Finally in section V we conclude this paper.

II. Related Works

Target Tracking: The problem of tracking multiple targets using sensor networks has been explored by prior works in [28], [20], [22], [29] etc. RASS [28] provides a system for transceiver-free user tracking with RF based technology. But it is not suitable when multiple user trajectories overlap. This is because it assumes small enough triangular sized node set deployments to separately detect individual users. The work in [6] uses received signal strength (RSS) measurements for target tracking. Using RSS is unreliable in different physical environments, thus limits it's general applicability.

Binary sensors (e.g. passive infrared motion sensors) have drawn considerable contribution ([26], [23] etc) for tracking applications, because of the properties like simplicity, non-invasive property and minimal communication requirements. However, most of the related works are based on some geometric models that can have limited applicability in real-time systems and varying environments. The existing works on multi-target tracking either use expensive and invasive sensors ([10], [4]) or depend on specific models like geometry of sensing range and noise model ([23], [29]).

Our designed *FindingHuMo* system tries to alleviate the drawbacks in the literature. It provides transceiver-free tracking solution, but also for multiple overlapping users, and don't require specific deployment patterns. In addition, it utilizes *motion activity context* in the system itself to track unknown and variable number of targets just from anonymous binary motion data.

There are existing works on target tracking using movement modeling based filtering and estimation. They mostly use Bayesian networks [17], Particle filters [21] or Kalman filter [12]. But most of those tracking algorithms in sensor networks either use expensive and invasive sensors (e.g. camera system) or depend on assumed movement model, noise model, sensor calibration, war-driving from WiFi/GSM signals (war-driving is the act of locating and possibly exploiting connections to wireless local area networks while in mobility).

In this paper we have used a modified version of Hidden Markov Model (HMM), with in-situ motion activity context. Some existing works on using HMM for multi-target tracking are [3], [4], [16]. But as mentioned earlier, these works use expensive and invasive sensors, while our work uses a proposed activity context aware variable order HMM with simple binary motion sensor.

Smart Environments: There is considerable amount of work done on sensor networks usage in Smart Environments, such as CASAS [1], BScope [15] etc. The related works in the literature mainly deal with analyzed activity patterns, or using activity patterns to adapt sensor network operations ([24], [8]). But the works didn't explore the use of *activity context* for real-time user tracking in Smart Environments. Actually multi-target tracking in Smart Environments with simple non-invasive binary sensor data (and without theoretical geometric models) and multiple overlapping path disambiguation, have been relatively under-explored.

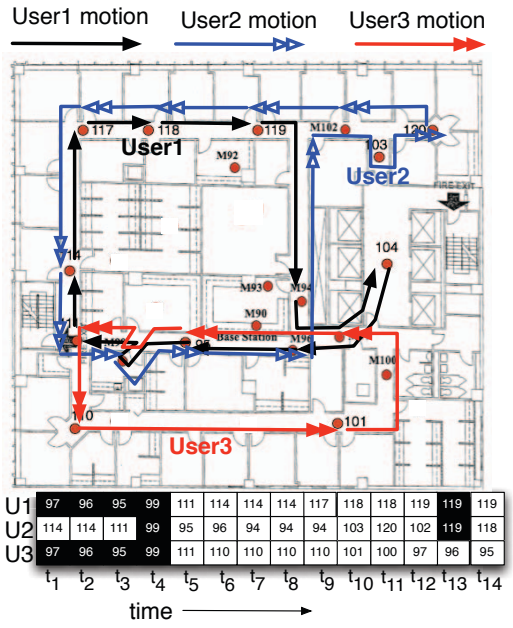


Fig. 2. Multi-user overlapping motion trajectories. The table below the figure shows the node or state sequence of each of the 3 users User1, User2 and User3, with time. The dark blocks in the table indicate motion overlap or crossover among the users.

III. Proposed Real-Time Multi-Target Tracking System

The working methods of the whole system is described step-by-step with the physical layout as shown in Figure 1 and accompanying example scenario in Figure 2. Figure 2 explains instance of overlapping multi-user motion trajectories in a real smart workplace environment (layout in Figure 1).

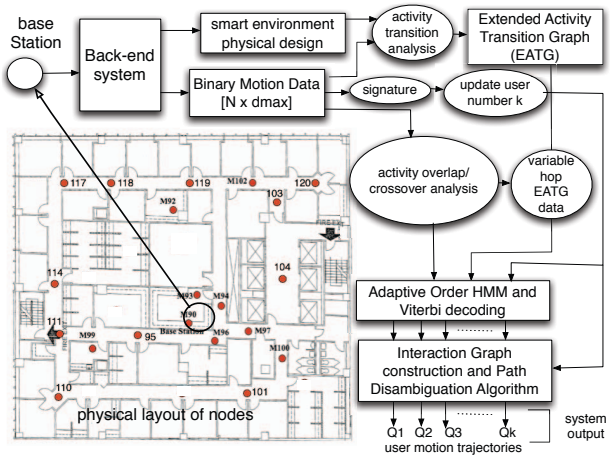


Fig. 3. *FindingHuMo* system: Multi-target tracking from binary motion sensor network.

A. Methodology

System Resources: *FindingHuMo* system consists of: (i) a static wireless sensor network (with binary motion sensors) deployed throughout the physical environment. The binary motion data from each sensor node are collected through multi-hop network into a base station; (ii) a back-end system computing user tracking algorithms on collected binary motion datastream.

TABLE I
List of parameters and their description

M	Set of motion sensor nodes ($N = \|M\|$)
$b_j(t)$	Output of sensor node m_j at timeslot t
K	The number of moving users
$q_i(t)$	Motion status of user i at timeslot t
$(t_s, (t_s + d_{max}T))$	Time window of *Adaptive-HMM* compuation
$(t_s, (t_s + Cd_{max}T))$	Time window of *CPDA* compuattion

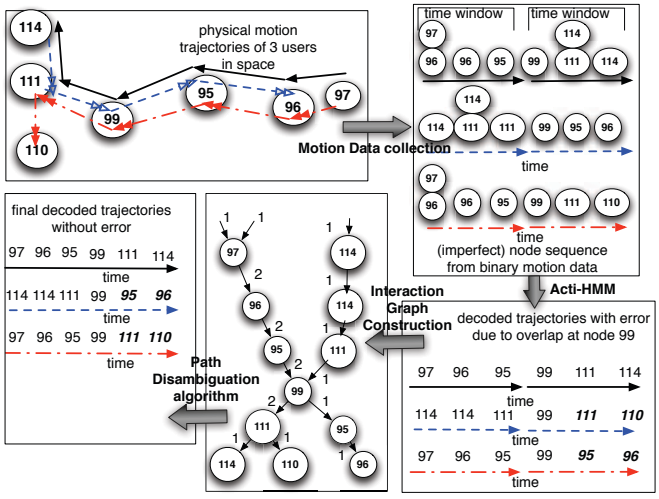


Fig. 4. A working example of *FindingHuMo*.

System Model and Problem Definition: Suppose the set of motion sensor nodes is M (N=$|M|$). Then the motion sensor network collects binary motion data $B(t) = \bigcup_{j=1}^{N} b_j(t)$, where $b_j(t) = 0$ or 1, is the binary motion status detected by node m_j ($\forall\ j \in (1, N)$) at timeslot t. Now each m_j ($\forall\ j \in (1, N)$) denotes the possible motion states of a user in the environment. Therefore for user i ($i \in (1, K)$, K is the number of users), if the motion activity state at time t is $q_i(t)$=m_j, it indicates that the user i is near the location of node m_j at time t. The application output of the system is thus the sequence of states $\{q_i(t_0), q_i(t_0 + T), q_i(t_0 + 2T), ..., q_i(t_0 + d.T), ..., q_i(t_0 + (D-1)T)\}$ $\forall\ i \in (1, K)$, where D can be a time period of user tracking (say 24 hours). The real-time requirement is that each $q_i(t_0 + dT)$ has to be computed in time $(t_0 + dT, t_0 + dT + D'T)$ where $D' << D$. Then the research problem is how to compute motion trajectories (the sequence of states) dynamically for all the users, from anonymous binary motion data stream B which doesn't contain any user specific information but just the collective motion status in the environment.

Algorithm 1 Pseudo code for back-end system connected to base station of sensor network

Input: Binary motion data $B(t_s, t_s + C.d_{max}.T) = \bigcup_j b_j(t)$ where $t_s \leq t \leq (t_s + C.d_{max}.T)$ and $j \in M' \subset M$

Output: Decoded state sequence $Q(t_s, t_s + C.d_{max}.T) = \bigcup_i \{q_i(t_s), ..., q_i(t_s + C.d_{max}.T)\}\ \forall\ i \in (1, K)$

1: Q_w = NULL;
2: **for** $t = t_s \rightarrow t_s + C.d_{max}.T$ **do**
3: $Q'(t, t + d_{max}.T) = Adaptive\text{-}Hmm(B(t, t + d_{max}.T))$; (Adaptive-HMM algorithm)
4: Q_w=$Q_w \bigcup Q'(t, t + d_{max}.T)$;
5: $t \leftarrow t + d_{max}.T$;
6: **end for**
7: $Q(t_s, t_s + C.d_{max}.T) = CPDA(Q_w)$; (*CPDA* path disambiguation algorithm)

System Procedure: The operational architecture of proposed *FindingHuMo* is shown in Figure 3. Based on some motion signature activities the system increments/ decrements K, the current number of users. Then based on detection of motion non-overlap/overlap in the binary motion data of time window t_S to $(t_S + d_{max}T)$, the system applies a variable state and variable order modified HMM. This exploits more information available and thus can capture contexts of user activities more accurately. The output is segments of state sequences s_i $(1 \leq i \leq K)$ for K users. This, combined (or can be called *stitched*) with decoded path segments in a larger time window of length $d_w.T$ (where $d_w.T$=$C.d_{max}T$, C is a constant), generates an *Interaction Graph*. A proposed path disambiguation algorithm *CPDA* is processed on that graph, which finally results in disambiguated node or state sequences of individual users. Algorithm 1 provides pseudo code for the back-end system. The description and pseudocode of *Adaptive-HMM* and *CPDA* are later presented in following subsections.

Working Example: Figure 4 illustrates how proposed *FindingHuMo* solves the abovementioned problem. The collected raw motion data contains unreliable node sequence with system noise. This is refined by applying *Adaptive-HMM*. The decoded state sequence may still contain error due to path crossover (e.g. crossover of decoded path for user 2 and user 3 at node 99). This is further corrected by *stitching* the decoded paths and forming an *Interaction Graph*, which is then disambiguated by applying proposed *CPDA* algorithm. This results in final decoded motion trajectories. It is worth mentioning that the position of user is presented in form of sensor nodes' position. Thus the tracking accuracy will be more (w.r.t the actual physical location of user) if the sensor node deployment is more dense. Also the maximum number of users that can be tracked simultaneously, is bounded by the number of sensor nodes deployed.

Next we separately present our proposed algorithms *Adaptive-HMM* and *CPDA*.

Algorithm 2 Pseudo code for *Adaptive-HMM* algorithm *Adaptive-Hmm*()

Input: Binary motion data $B(t, t + d_{max}.T)$

Output: Decoded state sequence $Q(t, t + d_{max}.T)$

1: $EATG(B(t, t + d_{max}.T))$; (explained in subsection III-B1)
2: Update extended activity transition graph G;
3: $\lambda = FormHMM(A, C, d_{max}, \tau, \Pi)$; (Adaptive-HMM model creation, explained in subsection III-B2)
4: $K = UserCount(B(t, t + d_{max}.T))$; (to update the number of current users K, explained in subsection III-B3)
5: $Q(t, t + d_{max}.T) = Viterbi(\lambda, K)$; (explained in subsection III-B4)

B. Adaptive-HMM Algorithm

This subsection describes the *Adaptive-HMM* algorithm (*Adaptive-Hmm*() in main Algorithm 1). The pseudocode for *Adaptive-Hmm*() is shown in Algorithm 2. *Adaptive-HMM*'s operation is motion activity driven to some extent. This is in a sense that, based on the activity amount detected in the motion data segment, it applies different methods to extract the motion trajectories.

*1) **Extended Activity Transition Graph**:* This explains the task $EATG()$ in Algorithm 2. It is important to note that even single user trajectory, or multi-user non-overlapping trajectories cannot be reliably concluded from just the binary motion data. Some knowledge of activity transition relationship among the nodes (or states) is necessary for extracting exact motion trajectories. For example in Figure 1 if both the motion sensors 93 and 94 are triggered, then from transitional relation from last activated state 96 it can be concluded that the motion trajectory was ...96→94... instead of ...96→93... The notion of activity transitional relationship among the nodes or states is presented and used in this work in the form of an Extended Activity Transition Graph or *EATG*. In *EATG* $G = (M, E', E'', A', A'')$, a node $m_j \in M$ represents a sensor node in the environment, weighted edge $e' \in E'$ denotes a pair of sensor nodes that can physically be reached directly from each other, and weighted edge $e'' \in E''$ denotes a pair of sensor nodes that can physically be reached from each other by triggering one more node in between. The weights a' and a'' of edge e' and e'' respectively denote direct 1-hop and indirect 2-hop activity transition between the nodes.

$$a'(m_{j_2}, m_{j_1}) = \frac{\#events\,(m_{j_2}(t) \Rightarrow m_{j_1}(t + T))}{\#events\, m_{j_2}(t)} \quad (1)$$

$$a''(m_{j_2}, m_{j_1}) = \frac{\#events\ (m_{j_2}(t) \Rightarrow m_{j_1}(t + 2T))}{\#events\ m_{j_2}(t)} \quad (2)$$

Example *EATG* belonging the layout in Figure 1 is shown in Figure 5(a). Direct transition probability A' is constructed either directly from the physical layout or is trained from collected binary motion data as illustrated in equation 2. Indirect 2-hop transition probability A'' is trained from binary motion

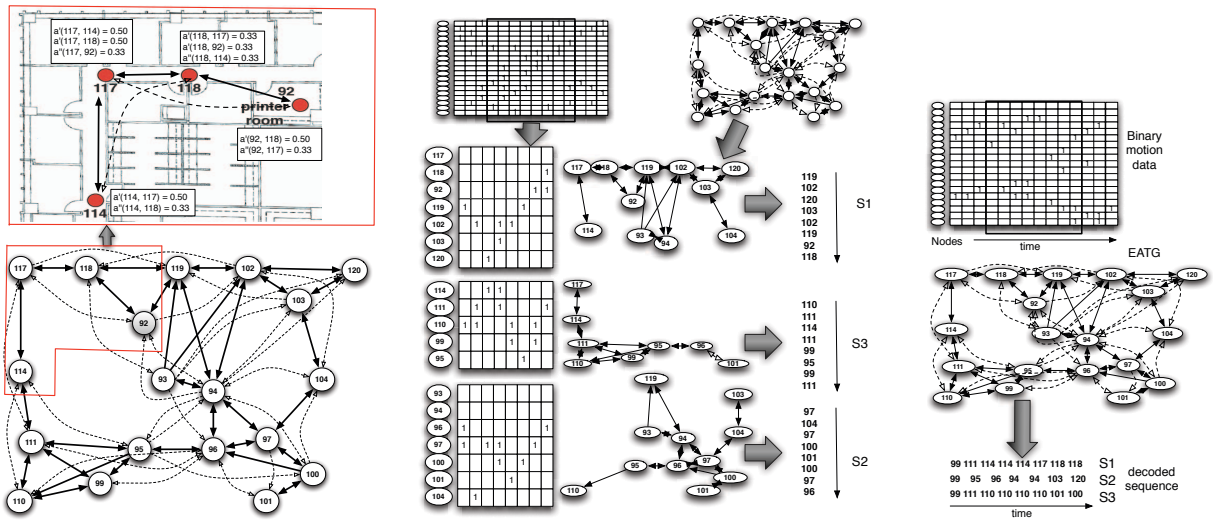


Fig. 5. (a) Extended activity transition graph *EATG* constructed for the smart environment layout shown in Figure 1. Solid lines indicate activity transition between nodes 1-hop away, while dashed lines indicate activity transition between nodes 2-hop away. Nodes 1-hop to each other, can be physically reachable without triggering any other node. (b) *Adaptive-HMM*: Splitting of non-overlapping motion into individual HMM's and then decoding state sequences using first order HMM. (c) *Adaptive-HMM*: Decoding of state sequences for overlapping motion in larger state and second order HMM.

data as in equation 2. The term $\#events\ m_{j_2}(t) \Rightarrow m_{j_1}(t+T)$ denotes the number of events where m_{j_2} is triggered at any time t and then followed by m_{j_1} at time $(t+T)$.

*2) **Adaptive Order HMM Modeling**:* This explains the task $FormHMM()$ in Algorithm 2. The system model is designed as a modified Hidden Markov Model (HMM) with a discrete time stochastic process. The modified model is named *Adaptive Order HMM* or *Adaptive-HMM*, and it's working is shown in Figure 6(a). The *Adaptive-HMM* model is $\lambda = (A, C, d_{max}, \tau, \Pi)$ and the set of states is M'.

(a) **States**: M'=$\{m_j\}$ where $m_j \in M$. So M' contains only part of the states in M. *Adaptive-HMM* chooses only the subset of states that are active and the neighbor (1-hop or 2-hop in $EATG$) states. This reduces the computational complexity without compromising the accuracy (theorem 1). In the HMM time window t to $(t + d_{max}.T)$, if the system detects say x non-overlapping motion (activated nodes at each slot of T are 1-hop away) it creates or forks out x HMM computations with each state set M' containing activated nodes of corresponding motion sequence and their 1-hop nodes (example in Figure 5(b)). But if it detects overlapping motion (sequence of motion activated nodes overlap in at least one slot of T) it creates a single HMM computation with state set M' containing activated nodes of motion sequences and upto their 2-hop nodes (example in Figure 5(c)). *Example*: In the example in Figure 5(b), the state sequence 110→111→114→111→99→95 →99→111 generate non overlapping state sequences with other activated states due to other users. Thus the state set of individual HMM computation contains only activated nodes (110, 111, 114, 99, 95) and their 1-hop nodes (117,96,101). But when that user's motion states are overlapped with others (motion shown in Figure 2) then the HMM computation contains all the active states and upto their 2-hop neighbors (Figure 5(c)).

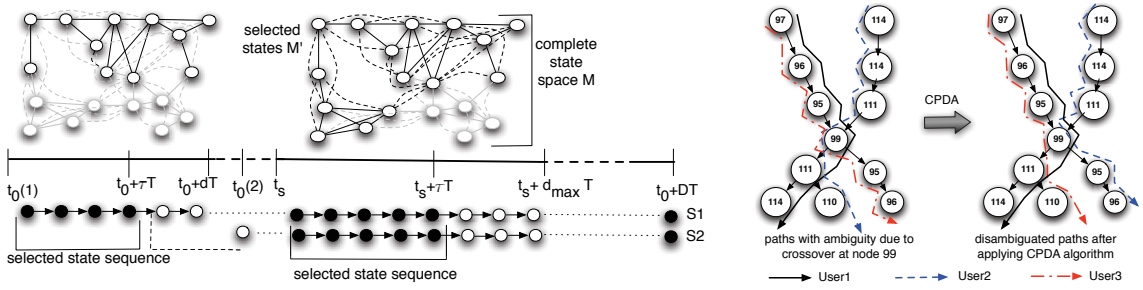
Sub-state selection in *Adaptive-HMM*: The sub-state selection in *Adaptive-HMM* doesn't affect the optimality of HMM model and Viterbi computation in our application scenario (due to activity transitional relationship among nodes upto 2 hop away). The proof could not be shown due to space constraints.

Theorem 1: In *Adaptive-HMM* the reduced state set M' results in the same optimal state sequence as that with complete state set M.

(b) **State Transition Probability**: In HMM computation for non-overlapping motion, $A = \{a(j_2, j_1)\}$, where $a(j_2, j_1) = P[q(t) = m_{j_1} | q(t-1) = m_{j_2}] = a'(m_{j_2}, m_{j_1})$, and in HMM computation for overlapping motion $A = \{a(j_3, j_2, j_1)\}$, where $a(j_3, j_2, j_1) = P[q(t) = m_{j_1} | q(t-1) = m_{j_2} AND q(t-2) = m_{j_3}] = a'(m_{j_2}, m_{j_1}).a''(m_{j_3}, m_{j_1})$; $a(j_3, j_2, j_1)$ is the state transition probability from motion activated node m_{j_3} at $(t-2)$ and m_{j_2} at $(t-1)$ to node m_{j_1} at time t ($q(t)$ denotes the current motion activated node at time t). Equivalently $a(j_2, j_1)$ denotes state transition from $(t-1)$ to t. Here *Adaptive-HMM* is motion activity driven. If there is motion overlap within time window, A includes transition from states at $(t-2)$ and $(t-1)$. But for no overlap A includes transition only from state at $(t-1)$.

(c) **Emission Probability Distribution**: For non-overlapping motion $C = \{c_j(p)\}$ and for overlapping motion $C = \{c_{j_2 j_1}(p)\}$. $c_{j_2 j_1}(p)$=$P[o_p | q(t) = m_{j_1} AND q(t-1) = m_{j_2}]$ is the probability that the system outcome at time t is $o_p \subseteq M'$ given node m_{j_1} is activated at current t and node m_{j_2} was activated at $(t-1)$. Equivalent meaning stands for $c_j(p)$.

(d) **Time Window and Threshold**: d_{max} is the time duration of the applied HMM. So the HMM time length or time window is from t_s to $(t_s + d_{max}.T)$. The HMM is computed for the time



(a) (b)

Fig. 6. (a) Activity context driven selection of state set and state transitions in *Adaptive-HMM*. The state sequence is saved only till $t_s + \tau.T$, and the next HMM window computation starts at $t_s + (\tau + 1).T$ instead of $t_s + (d_{max} + 1).T$. For single activated state the state set is smaller (activated nodes and their 1-hop neighbors) and uses transition only from $(t-1)$. But for multiple simultaneous activated states the state set is larger (activated nodes and upto their 2-hop neighbors) and uses transitions from time $(t-2)$ and $(t-1)$. (b)Illustrative example of proposed *CPDA* algorithm.

window t_s to $(t_s + d_{max}.T)$, but the resulting state sequence are saved only from t_s upto an instant $(t_s + \tau.T)$. τ indicates a threshold point for accepting the resulting state sequence.

(f) **Initial State Distribution**: $\Pi = \{\pi_j\}$ $(m_j \in M')$, where $\pi_j = P[q(t_s) = m_j]$ is the probability that at starting time t of HMM time window the activated node is m_j.

*3) **User Count in HMM time window**:* This explains the task $UserCount()$ in Algorithm 2. It updates the number of users K in HMM time window $(t, t + d_{max}.T)$ using following: $K = \max(K_{pre} + Sig_{in} - Sig_{out}, K_{now})$. K_{pre} is the number K from the previous HMM window $(t - d_{max}.T, t)$, Sig_{in} is the number of *user entry signatures* (e.g. node sequence $104 \rightarrow 103$), Sig_{out} is the number of *user exit signatures* (e.g. node sequence $97 \rightarrow 104$), K_{now} is the maximum number of triggered nodes (that are at least 2 hops away in $EATG$) in any unit slot of T in the window.

*4) **Viterbi Computation**:* The procedure is explained in the task $Viterbi()$ in Algorithm 3. Given the values of M', A, C, d_{max}, τ and Π, the HMM generates system observation sequence $O = O(t)\ O(t+T)\ O(t+2T)\ ...O(t+dT)..\ O(t+d_{max}.T)$ = say $o^f(t, T, d_{max})$ (where each $O(t + dT) \subseteq M'$). So the problem is given such system observation sequence O, the model λ and states M', how to choose the corresponding state sequence $Q_i = q_i(t)\ q_i(t + T)..q_i(t + dT)..q_i(t + \tau.T) =$ (say) $q_i^f(t, T, \tau)$ for each user i between time t and $(t + \tau.T)$. Finding the optimal state sequence with respect to the Maximum a posteriori (MAP) criterion is efficiently done with the Viterbi algorithm and tracing back through a matrix of back-pointers, starting from the end of the sequence. Standard Viterbi decoding algorithm is modified for: multiple observation, multiple sequence decoding, and fitting for activity awareness. For non-overlapping motion, viterbi algorithm is computed on first order HMM [20] (task $Viterbi^1()$) where transitions from time $(t-1)$ to t are considered. For overlapping motion, viterbi algorithm is computed on second order HMM [25] (task $Viterbi^2()$) where transitions from time $(t-2)$ and $(t-1)$ to t are considered. The pseudocodes for $Viterbi^1()$ and $Viterbi^2()$ are shown in Algorithm 4 and Algorithm 5 respectively. It's worth mentioning that second order HMM captures a more amount of the activity contextual information than the first order HMM.

Algorithm 3 $Viterbi(\lambda, K)$: Viterbi decoding in *Adaptive-HMM*

Input: HMM model λ, user number K
Output: Decoded sequence $Q(t, t + d_{max}.T)$

1: **if** No motion overlap detected among trajectories **then**
2: for each trajectory: (i) Update λ by keeping only triggered nodes and their neighbors in $EATG$; (ii) $Q(t, t + d_{max}.T) = Viterbi^1(\lambda)$; (1-state Viterbi decoding)
3: **else**
4: (i) Update λ by keeping only triggered nodes and their neighbors in $EATG$; (ii) $Q(t, t + d_{max}.T) = Viterbi^2(\lambda, K)$; (2-state Viterbi decoding)
5: **end if**

Real-Time applicability of Adaptive-HMM: There were some constraints to directly using standard HMM model and Viterbi algorithm to our real-time application scenario. Regarding length of time window (say W) the standard Viterbi algorithm requires $O(W)$ operations. But the standard algorithm is not applicable in the case of a streamed input (with potentially no ending in sequence) and requirement of output within bounded delay. Regarding size of state space (say S), the standard Viterbi algorithm requires $O(S^2)$ operations, and still even on average $O(S\sqrt{S})$ operations by a modified version of Viterbi [19]. Thus for real-time applicability we have designed a model that is activity context aware with: (a) bounded length of time window (t_s to $(t_s + d_{max}.T)$), and (b) varying size of system state M' that is the set of motion activated nodes (in the time window) and their 1-hop or 2-hop neighbor nodes in $EATG$ (explained earlier). Therefore depending on the amount of activity in sensed binary motion data *Adaptive-HMM* dynamically selects the state space and HMM order.

C. Path Disambiguation Algorithm CPDA

The output state sequences from *Adaptive-HMM* in each time window (of length $d_{max}.T$) is partially disambiguated

Algorithm 4 Viterbi algorithm $Viterbi^1(\lambda)$ on HMM for *non−overlapping* motion

Input: HMM λ=$(A, C, d_{max}, \tau, \Pi)$; state set M'; observation sequence O=$O(t_s)O(t_s+T)..O(t_s+dT)..O(t_s+d_{max}.T)$; A and C contain state information from $(t-1)$ to t; M' contains activated nodes and their 1-hop nodes in $EATG$.
Output: Optimal activity state sequence Q=$q(t_s)q(t_s+T)..q_i(t_s+dT)..q_i(t_s+\tau.T)$ ($\tau \leq d_{max}$).
Variables: $\delta_d(j)$= $\max_{q(t_s),q(t_s+T),..,q(t_s+(d-1)T)} P[q(t_s), q(t_s+T), .., q(t_s+dT) = j, O(t_s), .., O(t_s+dT)|\lambda]$
$\psi_d(j)$= $\text{argmax}_{q(t_s),q(t_s+T),..,q(t_s+(d-1)T)} P[q(t_s), q(t_s+T), .., q(t_s+dT) = j, O(t_s), .., O(t_s+dT)|\lambda]$

1: Initialization:

$$\delta_0(j) = \pi_j.c_j(O(t_s)) \ (m_j \in M') \quad and \quad \psi_0(j) = 0$$

2: Recursive step:

$$\delta_d(j) = \max_{m_{j'} \in M'} [\delta_{d-1}(j')a(j', j)].c_j(O(t_s + dT))$$

$$\psi_d(j) = \underset{m_{j'} \in M'}{argmax}[\delta_{d-1}(j')a(j', j)]$$

where ($1 \leq d \leq d_{max}$ and $m_j \in M'$)

3: Termination:

$$P^* = \max_{m_j \in M'} [\delta_{d_{max}}(j)] \quad u^*_{d_{max}} = \underset{m_j \in M'}{argmax}[\delta_{d_{max}}(j)]$$

4: State sequence backtracking:

$$u^*_d = \psi_{d+1}(u^*_{d+1}), \quad d = (d_{max} - 1), (d_{max} - 2), ..., 0$$

5: Output: $\{q(t_s)$=u^*_0, $q(t_s + T)$=u^*_1, .., $q_i(t_s + dT)$=$u^*_d..q_i(t_s + \tau.T)$=$u^*_\tau\}$

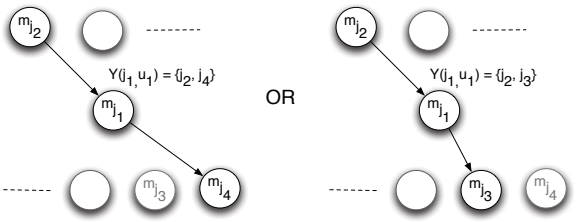


Fig. 7. Explanation of *CPDA*. The *non-feature* node m_{j_1} has property $Y(j_1, u_1)$ (u_1 is the identifier of one of the users passing through m_{j_1}). $Y(j_1, u_1)$=$\{j_2, j_4\}$ is the scenario on left-hand side, and $Y(j_1, u_1)$=$\{j_2, j_3\}$ is the scenario on right-hand side.

from the effect of path overlap or crossover. But it can't always remove longer term path ambiguity that spreads beyond the *Adaptive-HMM* time window. To alleviate this, *FindingHuMo* applies a proposed *Crossover Path Disambiguation Algorithm* or *CPDA* to the joint *Adaptive-HMM* output of last C number of time windows. It works as follows:

1) Combining *Adaptive-HMM* output of last C HMM time windows, it creates a directed graph $G_I = (V_I, E_I, P)$ (called *Interaction Graph*) that contains nodes V_I and edges E_I belonging to the output state sequences. P (=$\{P(j)\}$ $j \in V_I$) is set of node property, explained in next step.
2) In the constructed *Interaction Graph*, each node j is given a state variable $P(j)$ containing information about how the user paths from the incoming nodes are distributed into the outgoing nodes. $P(j) = \{Y(j, u)\}$ for every user u passing through node j. Nodes with deterministic and fixed value of $Y(j, u)$ (like nodes containing exactly one path, entry/exit nodes) are called *feature* nodes. The other intermediate nodes with possibly different values of $Y(j, u)$ are called *non-feature* nodes. Say on node m_{j_1}, one of the incoming node to node m_{j_1} is m_{j_2}, and one of the outgoing nodes from m_{j_1} is m_{j_3}. Then for user path u_i, if $Y(j_1, u_i) = \{j_2, j_3\}$, then it indicates that path of u_i goes from m_{j_2} through m_{j_1} to m_{j_3}.
3) Now based on the constraint on path distributions (imposed by G_I) and the defined state values $Y(j, u)$ of *feature nodes*, the system applies Bayesian Network Inference on G_I to calculate most desirable state values $Y(j, u)$ of non-feature nodes. The conditional probability table value is selected as follows: $P(Y(j_1, u_i) = \{j_2, j_3\}|j_2) = a''(m_{j_2}, m_{j_3})$ (thus utilizing $EATG$ graph).
4) Finally, if for some *non-feature node* the $Y(j, u)$ contradicts with the state sequence computed by *Adaptive-HMM*, path segments in the state sequences are switched to follow $Y(j, u)$.

Algorithm 5 Viterbi algorithm $Viterbi^2(\lambda, K)$ on HMM for *overlapping* motion

Input: HMM λ=$(A, C, d_{max}, \tau, \Pi)$; state set M'; observation sequence O=$O(t_s)O(t_s+T)..O(t_s+dT)..O(t_s+d_{max}.T)$; A and C contain state information from $(t-2)$ and $(t-1)$ to t; M' contains activated nodes, their 1-hop and 2-hop nodes in $EATG$.
Output: Optimal activity state sequence Q=$q(t_s)q(t_s+T)..q_i(t_s+dT)..q_i(t_s+\tau.T)$ for each $k \in K$ ($\tau \leq d_{max}$).
Variables: $\delta_d(j', j, k)$= $\max_{q(t_s),q(t_s+T),..,q(t_s+(d-2)T)} P[q(t_s), q(t_s+T), .., q(t_s+(d-1)T) = j', q(t_s+dT) = j, O(t_s), .., O(t_s+dT)|\lambda]$ for each user $k \in K$
$\psi_d(j', j, k)$= $\text{argmax}_{q(t_s),q(t_s+T),..,q(t_s+(d-2)T)} P[q(t_s), q(t_s+T), .., q(t_s+(d-1)T) = j', q(t_s+dT) = j, O(t_s), .., O(t_s+dT)|\lambda]$ for each user $k \in K$

1: Initialization (for each user $k \in K$):

$$\delta_0(j', j, k) = \pi_j.c_{j'j}(O(t_s)) \ and \ \psi_0(j', j, k) = 0 \ (m_j, m_{j'} \in M')$$

2: Recursive step (for each user $k \in K$):

$$\delta_d(j', j, k) = \max_{m_{j''} \in M'} [\delta_{d-1}(j'', j', k)a(j'', j', j)].c_{j'j}(O(t_s + dT))$$

$$\psi_d(j', j, k) = \underset{m_{j''} \in M'}{argmax}[\delta_{d-1}(j'', j', k)a(j'', j', j)]$$

where ($1 \leq d \leq d_{max}$ and $m_j, m_{j'}, m_{j''} \in M'$)

3: Termination (for each user $k \in K$):

$$P^*(k) = \max_{m_{j'}, m_j \in M'} \delta_{d_{max}}(j', j, k)$$

$$u^*_{d_{max}}(k) = \underset{m_{j'}, m_j \in M'}{arg_j max}[\delta_{d_{max}}(j', j, k)] \quad u^*_{d_{max}-1}(k) = \underset{m_{j'}, m_j \in M'}{arg_{j'} max}[\delta_{d_{max}}(j', j, k)]$$

4: State sequence backtracking:

$$u^*_d(k) = \psi_{d+1}(u^*_{d+1}, u^*_{d+2}, k) \ d = (d_{max} - 2), (d_{max} - 3), ..., 0$$

5: Output (for each user $k \in K$): $\{q(t_s)$=$u^*_0(k)$, $q(t_s + T)$=$u^*_1(k)$, .., $q_i(t_s + dT)$=$u^*_d(k)..q_i(t_s + \tau.T)$=$u^*_\tau(k)\}$

Now we explain the proposed *CPDA* algorithm step-by-step through a working example shown in Figure 6(b).

1) After computing $Adaptive - HMM$ for the last 2 time

windows (thus here C=2) the combined output state sequences are: (i) 97 → 96 → 95 → 99 → 111 → 114, (ii) 114 → 114 → 111 → 99 → **111** → **110**, and (iii) 97 → 96 → 95 → 99 → **95** → **96**. These form the *Interaction Graph* $G_I = (V_I, E_I, P)$ containing the nodes and the corresponding edges, as shown in Figure 6(b).

2) Nodes 99 and 111 here are the *non-feature* nodes. Node 99 has 2 incoming nodes (95 and 111) and 2 outgoing nodes (111 and 95). Similarly node 111 has 1 incoming node (99) and 2 outgoing nodes (114 and 110). The for example for the user (say u_1) who was moving in the path 114 → 111 → 99... can cause for node 99: $Y(99, u_1) = \{111, 111\}$ or $Y(99, u_1) = \{111, 95\}$.
3) Now after running the Bayesian Network Inference, for example for node 99 and for user u_1, $P(Y(99, u_1) = \{111, 95|111\}) > P(Y(99, u_1) = \{111, 111|111\})$.
4) Thus the path (...111 → 99 → 95...) suggested by $Y(99, u_1) = \{111, 95|111\}$ contradicts the part of $Adaptive - HMM$ output path ...111 → 99 → 111.... Therefore the path 114 → 114 → 111 → 99 → **111** → **110** is corrected to 114 → 114 → 111 → 99 → **95** → **96**. This triggers the path 97 → 96 → 95 → 99 → **95** → **96** corrected to 97 → 96 → 95 → 99 → **111** → **110**. This is because the path constraint has to be satisfied for each node that number of incoming user paths is equal to the number of outgoing user paths. In this way the path disambiguation is performed in $CPDA$.

Therefore Bayesian inference in $CPDA$ helps eliminate some path ambiguity. This finishes the description of the proposed system $FindingHumo$.

IV. Performance Evaluation

In this section we first explain our experimental setup in a real smart environment, followed by system performance analysis of multi-user tracking experiments.

A. System setup

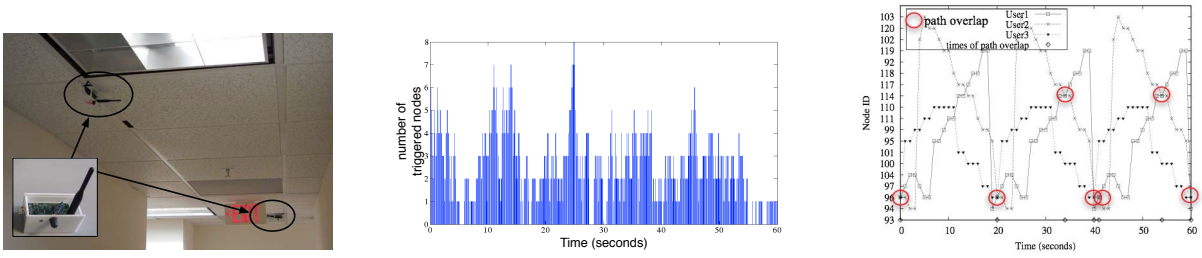
A network of 20 $TelosW$ [14] static wireless sensor nodes (Figure 8(a)) are deployed throughout the 30 meter x 30 meter floor (Figure 1) workplace environment (in workplace of Department of Computer Science, Georgia State University). As earlier shown in the physical layout in Figure 1, the sensor nodes are deployed mainly in the hallways, key positions (e.g. entry points: nodes 103, 104; exit points: nodes 97, 104; positions with high motion activities in workday: nodes 118, 117, 102,114, 100, 101), some rooms (e.g. printer room: node 92, kitchen: node 99, busy lab: node 93). These nodes are fixed on the ceiling and each of them are equipped with Panasonic AMN-31111 PIR (passive infrared) motion sensor. The $TelosW$ sensor nodes have sensor wake-on capability [14] to rationalize MCU (the processing unit) usage. The detected motion data (sampled at 10 Hz when event triggered by motion) are collected by the base station through multi-hop communication (formed a 5-hop network) and stored in a back-end database. The choice for system parameters are as follows. Length of unit timeslot T has been chosen as 1 second, M contains 20 motion sensor nodes, $d_{max}.T$ is 5 seconds. Based on system testing and evaluation, there were no false positive or false negative observed from the motion sensor. However, the tracking challenges come from factors like time synchronization, loss of transmitted sensor data, non-uniform node distribution and large number of overlapping users.

B. Multi-user Tracking Experiment

Experiment setup: In the performance evaluation experiment, three users (say $U1$, $U2$ and $U3$) are made to repeatedly move through the space in overlapping paths (as shown in Figure 2 and Figure 8(c)). The ground truth of user position was recorded by the moving user, to compare later with computed user trajectories. The ground truth has been compared to the following: (i) offline 1-HMM (first order HMM computed offline on full time data), (ii) offline full state 2-HMM (second order HMM computed offline on full time data), (iii) online full state 1-HMM (first order HMM computed online on time window of data), (iv) online full state 2-HMM (second order HMM computed online on time window of data), and (v) $FindingHumo$ (this uses second order HMM computed online on time window of data with activated subset of states, and then applying path disambiguation algorithm). It is important to mention that no current method was found in the existing literature, that fits this application scenario and requirements (binary sensor data, no geometric model etc.). $FindingHumo$ is compared here with different possible configurations of HMM computation, showing the utilities of: online time window based HMM, partial state HMM and path disambiguation algorithm.

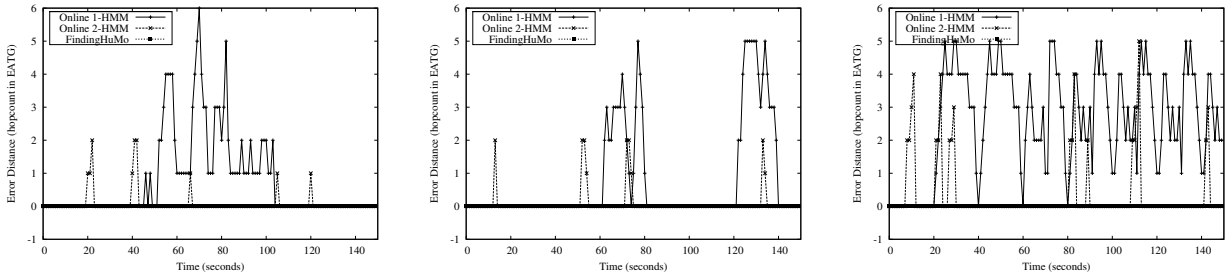
Performance evaluation of multi-user experiment: The 3 user overlapping path experiment was conducted for about 150 seconds. Figure 8(b) shows the number of motion triggered nodes through 60 seconds run of the experiment process. This indicates that a lot of time more or less than 3 (user number) nodes are triggered at once. This is because of real scenarios: the different and non-uniform walking speeds of users, time synchronization issue (thus unreliable state sequences), the non-uniformity of node deployments etc. This creates the challenge for tracking algorithms to decode 3 best states indicating the state of each user. The tracking goal is to locate users with key logical points, instead of exact co-ordinate. Therefore the tracking error is measured with hopcount in the $EATG$ graph. Say the ground truth is state m_1 and detected state is m_2, then the tracking error is the shortest hopcount between nodes m_1 and m_2 in $EATG$. This is valid because adjacent nodes in $EATG$ indicate direct physical reachability. If the error had to be measured in terms of distance, then the error distance could be measured as the summation of all hop distances. This doesn't change the general applicability of our designed system.

Figures 9(a), 9(b) and 9(c) show the performance of $FindingHuMo$ for tracking user1, user2 and user3 respectively. There are some key observations made from the comparative performance analysis. (i) Full-state HMM compu-



(a) (b) (c)

Fig. 8. (a) Testbed deployment of PIR motion sensor nodes in a smart workplace environment. (b) Number of motion triggered nodes during 3 user experiment. (c) Ground truth motion trajectories of 3 users during experiment, with the path overlap/crossover shown.



(a) (b) (c)

Fig. 9. (a) Tracking error (hopcount in *EATG*, between ground truth node and detected node) measured for user1. (b) Tracking error measured for user2. (c) Tracking error measured for user3.

tation time overhead is much higher than partial-state computation. (ii) Online 1-HMM and online 2-HMM (use same time windows as that of *FindingHumo*) performed same (same tracking error) as their corresponding offline version. Therefore it is validated that here time window based online computation in *FindingHumo* does not affect the optimality of computed path. (iii) As shown in Figures 9(a), 9(b) and 9(c), 2-HMM has much lesser tracking errors than 1-HMM. This is the advantage of using second order HMM for overlapping paths. (iv) Finally, as in Figures 9(a), 9(b) and 9(c), *FindingHuMo* showed no error in computed path. The errors that occurred in online 2-HMM (mostly 1 or 2 hop error distances) were locally corrected by path disambiguation algorithm in *FindingHuMo*. The average tracking error (in terms of tracking distance in hopcount) per unit timeslot T is as follows: (a) 1-HMM: 0.75 for user1, 0.78 for user2, 2.74 for user3; (b) 2-HMM: 0.08 for user1, 0.1 for user2, 0.35 for user3; (c) *FindingHuMo*: 0.00 for all users. Tracking User3 had more errors for just HMM based computation, because it had more crossover with other users. But overall *FindingHuMo* system corrects this error for all users, by added inference in *CPDA*.

In addition to overlapping path experiment, also test was conducted with non-overlapping paths of those 3 users. The system performance indicated same trend as for the overlapping scenario. Therefore overall, the experimental results show that *FindingHumo* performs much better than other comparative configurations in computing accurate multi-user motion trajectories.

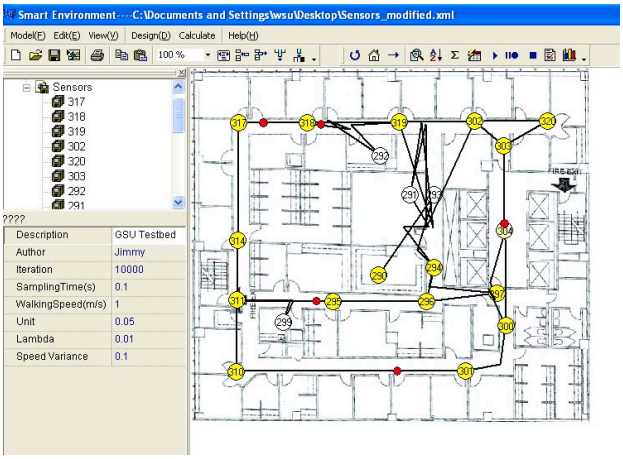


Fig. 10. The Smart Environment Simulator used for validating tracking performance of *FindingHuMo*.

C. Smart Environment Simulator and Real-time Graphical User Interface Design

We have designed a Smart Environment Simulator (shown in Figure 10) that can inject moving users in a virtual Smart Environment and generates corresponding binary motion sensor data stream. The output of the simulator and the ground truth are used to analyze and validate the tracking performance of *FindingHuMo* system. The simulator is available publicly for download at the

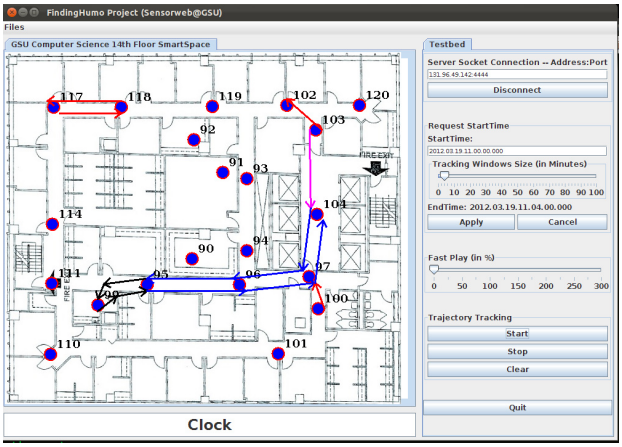


Fig. 11. The designed Graphical User Interface (GUI) for visualizing user movements in *FindingHuMo*.

following link: http://sensorweb.cs.gsu.edu/sites/default/files/research/projects/ActiSen/smartenvironment-simulator.zip.

We also have designed a live Graphical User Interface (GUI) (shown in Figure 11) that shows the real-time motion paths of all the users in the smart workplace environment. It can also replay the motion trajectories of users in the Smart Environment for some history of time window. These can be utilized in future work for analyzing the user motion patterns or behaviors in a workplace.

V. Discussion and Conclusion

This work opens a relatively under-explored area where real-time multi-target tracking is done without any geometric model and with simple non-invasive sensors. In conclusion, this paper presents a novel design for real-time and scalable tracking of multiple (unknown and variable number of) targets or human users in any crowded Smart Environments. The performance improvement is demonstrated with results from experiments in real-time system in a smart workplace environment.

References

[1] CASAS Smart Home Project. http://ailab.wsu.edu/casas/.

[2] MavHome. http://ailab.uta.edu/mavhome/.

[3] H. Ardö, K. Åström, and R. Berthilsson. Online Viterbi Optimisation for Simple Event Detection in Video. In *ICVSS*, 2007.

[4] H. Ardö, K. Åström, and R. Berthilsson. Real Time Viterbi Optimization of Hidden Markov Models for Multi Target Tracking. In *WMVC*, 2007.

[5] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a Moving Object with a Binary Sensor Network. In *SenSys*, 2003.

[6] X. Chen, A. Edelstein, Y. Li, M. Coates, M. Rabbat, and A. Men. Sequential Monte Carlo for Simultaneous Passive Device-Free Tracking and Sensor Localization Using Received Signal Strength Measurements. In *International Conference on Information Processing in Sensor Networks (IPSN'11)*, Chicago, USA, 2011.

[7] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury. Did You See Bob?: Human Localization using Mobile Phones. In *MobiCom*, 2010.

[8] D. De, S. Tang, W.-Z. Song, D. Cook, and S. K. Das. ActiSen: Activity-aware Sensor Network in Smart Environments. *Journal of Pervasive and Mobile Computing (PMC)*, Jan. 2012.

[9] N.-W. Gong. Configurable Dynamic Privacy for Pervasive Sensor Networks. Master's thesis, Massachusetts Institute of Technology, 2009.

[10] M. Han, W. Xu, H. Tao, and Y. Gong. An Algorithm for Multiple Object Trajectory Tracking. In *Conference on Computer Vision and Pattern Recognition (CVPR'04)*, Washington DC, USA, June 2004.

[11] D. Jung, T. Teixeira, and A. Savvides. Toward Cooperative Localization of Wearable Sensors using Accelerometer and Camera. In *INFOCOM*, 2010.

[12] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[13] H. Lee, M. Wicke, B. Kusy, O. Gnawali, and L. Guibas. Data Stashing: Energy-Efficient Information Delivery to Mobile Sinks through Trajectory Prediction. In *IPSN*, 2010.

[14] G. Lu, D. De, M. Xu, W.-Z. Song, and B. Shirazi. TelosW: Enabling Ultra-Low Power Wake-On Sensor Network. In *Seventh International Conference on Networked Sensing Systems (INSS'10)*, Kassel, Germany, June 2010.

[15] D. Lymberopoulos, T. Teixeira, and A. Savvides. The BehaviorScope Framework for Enabling Ambient Assisted Living. *International Journal on Personal and Ubiquitous Computing*, 14(6), 2010.

[16] X. Ma, D. Schonfeld, and A. Khokhar. Distributed multi-dimensional hidden Markov model: theory and application in multiple-object trajectory classification and recognition. In *Conference of Multimedia Content Access: Algorithms and Systems II (SPIE'08)*, San Jose, CA, USA, Jan. 2008.

[17] P. Nillius, J. Sullivan, and S. Carlsson. Multi-Target Tracking - Linking Identities using Bayesian Network Inference. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2187–2194, 2006.

[18] D. O. Olguín, B. N. Waber, T. Kim, A. Mohan, K. Ara, and A. Pentland. Sensible Organizations: Technology and Methodology for Automatically Measuring Organizational Behavior. *IEEE Transaction on Systems, Man, and Cybernetics*, 39, Feb. 2009.

[19] S. Patel. A lower-complexity Viterbi algorithm. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'95)*, Detroit, USA, 1995.

[20] L. R. Rabinder. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb. 1989.

[21] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. 2004.

[22] J. Shin, L. J. Guibas, and F. Zhao. A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In *Proceedings of the 2nd international conference on Information processing in sensor networks (IPSN'03)*, Palo Alto, California, USA, Apr. 2003.

[23] J. Singh, R. Kumar, U. Madhow, S. Suri, and R. Cagley. Tracking Multiple Targets Using Binary Proximity Sensors. In *Proceedings of the 6th international conference on Information processing in sensor networks (IPSN'07)*, Cambridge, Massachusetts, Apr. 2007.

[24] S. Tang, D. De, W.-Z. Song, D. Cook, and S. Das. ActSee: Activity-Aware Radio Duty-Cycling for Sensor Networks in Smart Environments. In *Eighth IEEE International Conference on Networked Sensing Systems (IEEE INSS)*, 2011.

[25] S. M. Thede and M. P. Harper. A second-order Hidden Markov Model for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics (ACL'99)*, Stroudsburg, PA, USA, 1999.

[26] Z. Wang, E. Bulut, and B. K. Szymanski. Distributed Target Tracking with Directional Binary Sensor Networks. In *GLOBECOM*, 2009.

[27] A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *Special Issue IEEE Networks*, 22(4):26–33, July 2008.

[28] D. Zhang, Y. Liu, and L. Ni. RASS: A Real-time, Accurate and Scalable System for Tracking Transceiver-free Objects. In *Ninth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'11)*, Seattle, USA, 2011.

[29] Z. Zhong, T. Zhu, D. Wang, and T. He. Tracking with Unreliable Node Sequences. In *Proceedings of the 28th Conference on Computer Communications (INFOCOM'09)*, pages 1215–1223, Reo de Janeiro, Brazil, Apr. 2009.