Detection and Tracking of Mobile Events with Dynamic Signatures Using Mobile Sensors

Na Yu and Qi Han
Department of Mathematical and Computer Sciences
Colorado School of Mines, Golden, CO 80401
{nyu, qhan}@mines.edu

Abstract—**With the advances in sensing, communication, and computation, there is an increasing need to track mobile events such as air pollutant diffusion, toxic gas leakage, or wildfire spreading using mobile sensors such as robots. Lots of existing work use control theory to plan the path of mobile sensors by assuming that the event evolution is known in advance. This assumption has severely limited the applicability of existing approaches. In this work, we aim to design a detection and tracking algorithm that is capable of identifying multiple events with dynamic event signatures and providing event evolution history that may include event merge, split, create and destroy. Simulation results show that our approach can identify events with low event count difference, high event membership similarity, and accurate event evolution decisions, while using a reasonable number of tracking robots.**

I. Introduction

For environmental monitoring and protection, there is an increasing need to deal with mobile events such as air pollutant diffusion, toxic gas leakage, or wildfire spreading. These events are usually large amorphous phenomena which occupy irregular geographical regions and change the locations and shapes over time. Moreover, several branches of an event may mix when they come close or split into separate parts due to external forces. Mobile sensor networks are very powerful when being used to detect and track these dynamic events for the following reasons. First, mobile sensors such as robots can adapt to the environment of mobile events in the presence of external forces such as winds and obstacles. Second, mobile sensors are able to follow the mobile events as both sensors and actuators. Third, deploying mobile sensor networks with planned paths can decrease the amount of sensors needed than using stationary wireless sensor networks. This work considers event detection and tracking using mobile sensor networks. In particular, we are interested in keeping track of events with dynamic signatures. A signature is the distinct identity of an event that can be used to specify the event from the application level. Events with dynamic signatures are capable of splitting apart or merging together. Our approach also aims to provide the applications with event evolution history that can be used to analyze event patterns and predict future behaviors.

Existing work using mobile sensors for event tracking as detailed in Section II either assume that events never combine into a larger one nor disintegrate into several smaller ones, or assume that event evolution is known in advance so that events can be modeled formally. These assumptions have severely limited the applicability of existing approaches, especially in a general scenario containing multiple dynamic events with different evolving patterns. Although previous work—DRAGON [1] proposes an algorithm that is capable of tracking dynamic events even in the presence of event merges and splits, it only works for stationary wireless sensor networks, and event evolution is not addressed either.

This paper proposes MEMS—a novel pipelined approach for dynamic event detection and tracking. Its noteworthy features are (1) uses detection robots in a distributed way; (2) uses the minimal number of tracking robots as needed; (3) accurately identifies multiple events with dynamic signatures; (4) provides accurate event evolution history including merge, split, create and destroy. We will use mobile sensors and robots interchangeably in the following.

II. Related Work

Event detection and tracking using stationary wireless sensor networks [2]–[6] is a well-studied topic. Using mobile sensor networks, several pieces of work focus on cooperatively improving coverage [7] [8], maintaining connectivity [9]–[11], and dynamically partitioning the mobile sensors for multiple targets [12] [13], however, none of them consider the merging and splitting of the targets. Yet, another body of work focuses on the path planning or control of robots. For instance, [14] models a single cloud using splinegon and plans the paths of the robots [15] according to the modeled cloud dynamics [16], however, it does not consider robot dynamics while generating the paths. The work in [17] and [18] provide control algorithms on robot swarms for detection, tracking, and mapping of chemical clouds. Although [17] [18] do not assume a-prior knowledge of the cloud evolution pattern as [14]–[16], they use very simple search methods that they cannot assure all clouds could be detected.

To the best of our knowledge, the work presented in this paper is the first that can provide event evolution history for dynamic events with possible merges and splits using mobile sensor networks without assuming the models of events in advance.

III. MEMS Overview

MEMS is designed to detect and track mobile events using mobile sensors in a large geographical area, where the events are assumed slower than the robots. During initialization, the detection area is decomposed into detection units using the cell decomposition method as widely used in the robot coverage problem [19] with one detection robot in each unit. The size of a detection unit is determined in a way that the robots can directly communicate with robots in adjacent detection units. Each detection unit is further decomposed into a number of sensing cells. The size of a sensing cell is determined by the sensing range of the sensors used to track the events. Similarly, each sensing cell is decomposed into a number of tracking cells. The size of a tracking cell is a determining factor for tracking precision. Then, the time period for one round is calculated based on the maximum event size, maximum event speed, and maximum robot speed. For instance, Fig. 1 shows 2 detection units (outlined using solid bold lines), each has 16 sensing cells inside (divided by solid lines), and each sensing cell has 25 tracking cells like the top right sensing cell.

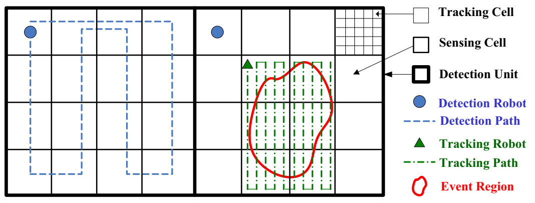


Fig. 1. Sample cell decompositions

After initialization, MEMS works in two phases: detection and tracking. To speed up the process, it adopts a pipelined approach where detection and tracking execute in parallel with detection robots and tracking robots, respectively (Fig. 2). MEMS periodically initiates a new round of detection, whose results will be used for tracking in the next round (Fig. 3).

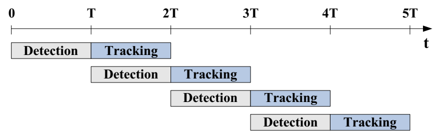


Fig. 2. Event detection and tracking pipeline

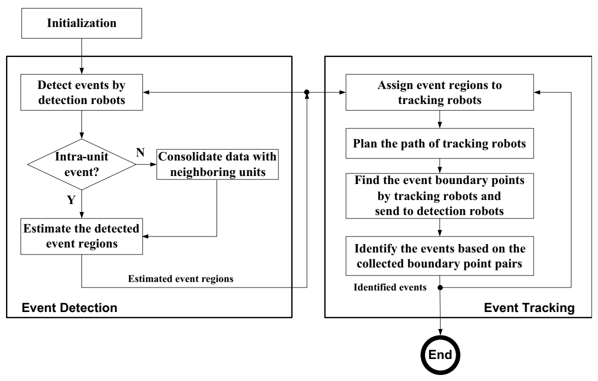

Initialization
Detect events by detection robots
Intra-unit event?
N
Y
Consolidate data with neighboring units
Estimate the detected event regions
Estimated event regions
Event Detection
Assign event regions to tracking robots
Plan the path of tracking robots
Find the event boundary points by tracking robots and send to detection robots
Identify the events based on the collected boundary point pairs
Identified events
Event Tracking
End

Fig. 3. Detailed event detection and tracking in one round

In the **detection phase**, each detection robot follows a certain path in its detection unit as shown in Fig. 1 to check any new event regions (i.e., consecutive sensing cells that are detected with events by the corresponding sensors of the robot). MEMS represents a sensing cell using 1 when the event is present in that sensing cell and using 0 otherwise. This way, the event presence information can be represented using a bit vector. If an event region is only inside one detection unit (i.e., intra-unit event), then the corresponding detection robot has the complete information of the event region. Otherwise, if the event region spans several detection units (i.e., inter-unit event), the corresponding detection robots in those units need to consolidate their information about the event region and designate one detection robot to hold the complete information. MEMS accomplishes this by gathering the information from all relevant detection units in a clock-wise fashion to the designated detection robot.

In the **tracking phase**, detection robots assign consecutive sensing cells in each event region to several tracking robots, where the number of tracking robots is determined by the event region size and the maximum robot speed. Further, detection robots plan the tracking path in order to cover all the tracking cells in the consecutive sensing cells assigned to the tracking robots. Then, the tracking robots sense the events along their tracking paths, and find event entry and exit boundary points and send the points in pairs to the detection robots in charge. An $O(nlog(n))$ plane sweep algorithm is applied to the boundary point pairs to separate the individual events in each event region. Boundary point pairs of the identified events along with event properties such as ECoM and ESoR as detailed in section IV can be further used to map the boundary of the events using the technique such as the Voronoi graph based contour map generation [20].

IV. MEMS Details

We first present the two key steps of MEMS: event region estimation and event identification, then present the unique feature of MEMS: event evolution.

A. Event Region Estimation

This step estimates the event regions (Algorithm 1) using the event presence information gathered by detection robots. For instance, a detection unit with 4x4 sensing cells and a bit vector $v = \{0011001100001111\}$ will be grouped into two disconnected event regions, i.e., $c[0][2], c[0][3], c[1][2], c[1][3]$ and $c[3][0], c[3][1], c[3][2], c[3][3]$. We assume that there is a server located in the reference detection unit with the ID (0,0). Then, the inter-unit event information is consolidated to the detection robot in the detection unit with the lowest ID which is closest to the sever, incurring the least communication overhead if the event information (i.e., the bit vector) needs to be sent to the sever.

To determine whether two events are overlapped or whether an event is inside another one (note that these events are not necessarily identified at the same time), two key concepts are used: event center of mass, and event size of radius.

Event Center of Mass (ECoM): for event e_i, assume the boundary points set identified at time t is $P_{e_i} = \{p_1, p_2, ..., p_n\}$. For each point p_j, the sensor reading is $s_{p_j,t}$, and the location is l_{p_j}. Then, the ECoM for event e_i at time t is shown in Eqn(1).

$$M_{e_i,t} = \frac{\sum_{\forall p_j \in P_{e_i}} s_{p_j,t} \times l_{p_j}}{\sum_{\forall p_j \in P_{e_i}} s_{p_j,t}} \quad (1)$$

Event Size of Radius (ESoR): the ESoR for an event is defined as the average distance from the boundary points to the ECoM of the event, as shown in Eqn(2).

$$R_{e_i,t} = \frac{\sum_{\forall p_j \in P_{e_i}} |l_{p_j} - M_{e_i,t}|}{n} \quad (2)$$

If $|M_{e_1,t_1} - M_{e_2,t_2}| < R_{e_1,t_1} + R_{e_2,t_2}$, then event e_1 at time t_1 and event e_2 at time t_2 are overlapped. Further, if $|M_{e_1,t_1} - M_{e_2,t_2}| < R_{e_1,t_1} - R_{e_2,t_2}$, then event e_2 at time t_2 is inside event e_1 at time t_1. ECoM and ESoR can be applied to event regions as well, where the boundary points are approximately the centers of the outmost sensing cells.

Algorithm 1 : Event Region Estimation

```
Input: sensing cells of detected events
Output: estimated event regions with ECoMs/ESoRs
 1: for all sensing cells detected with events do
 2:    Group consecutive sensing cells for each disconnected
       event regions;
 3: end for
 4: for all disconnected event regions do
 5:    if inter-unit event region then
 6:       Aggregate information of the consecutive sensing
          cells clockwise to the detection robot in the detec-
          tion unit with the lowest ID among all the units it
          occupies;
 7:       if the detection robot is different from last round
          then
 8:          Mark it as "event migrate";
 9:          Update the records of this event on the detection
             robots involved in current round and last round;
10:       end if
11:    end if
12:    Estimate ECoM and ESoR for the event region;
13: end for
14: for all estimated event regions do
15:    Compare ECoM and ESoR with last round;
16:    if no overlap and no migration then
17:       Mark it as "event create"
18:       Create new event record;
19:    end if
20: end for
21: Save all the estimated event regions in current round;
```

At the end of this step, an event region may contain multiple events which will be further identified according to the tracking precision in event identification.

B. Event Identification

Based on the estimated event regions, the detection robots can assign the event regions to a number of tracking robots and also plan the tracking paths as illustrated in section III. Then, the tracking robots for each event region report the boundary point pairs of events along their tracking paths to the detection robot who assigns the event region, and each detection robot further uses these boundary point pairs to identify individual events as shown in Algorithm 2. The plane sweep algorithm used to solve the closest pair of points problem [21] is applied here. By sweeping the line with an interval of the tracking cell size, it can group the consecutive boundary point pairs for individual events [22].

Algorithm 2 : Event Identification

```
Input: event boundary point pairs
Output: identified events with signatures/ECoMs/ESoRs
 1: for all event boundary point pairs do
 2:    Use the plane sweep algorithm to group the boundary
       point pairs within consecutive tracking cells for each
       individual event;
 3: end for
 4: for all separated groups of boundary point pairs do
 5:    Calculate ECoM and ESoR;
 6:    Compare ECoM and ESoR with last round;
 7:    if separated events in last round overlap in current round
       then
 8:       Mark it as "event merge";
 9:       Combine event signatures to the earliest one;
10:    end if
11: end for
12: for all identified events of last round do
13:    Compare ECoM and ESoR with current round;
14:    if ECoMs of multiple events in current round are inside
       a single event in last round then
15:       Mark it as "event split";
16:       Assign new event signatures to the splitted parts;
17:    end if
18:    if ECoM disappears in current round and it is not
       marked as migrate then
19:       Mark it as "event destroy";
20:       Terminate the event evolution records;
21:    end if
22: end for
23: Save all the identified events in current round;
```

C. Event Evolution

Event signature is a unique feature of MEMS, which is provided with a label consisting of round number (R#), detection robot ID (D#), and the group ID of the corresponding tracking robots (T#). Event evolution contains a series of records of the dynamic event signatures and the event merge, split, create, destroy actions in each round. The event evolution history is maintained on the detection robots in a distributed manner that each detection robot records the identified events under its management. Once a detection robot takes over an event that migrates from its previous detection robot in charge, it will request the last event signature of this event from the previous detection robot and send back the updated event signature with a status of migrating. Based on the event evolution history held by all the detection robots, we can construct an event evolution tree to answer event queries. For example, if the application is interested in the entire history of event $R3D1T2$, then the query results might look like $R3D1T2 \leftarrow R2D1T2 \leftarrow R1D1T2 + R1D1T3$. We can further map the detection robot ID and tracking ID to geographical locations, which in conjunction with contour maps will provide a clear idea of how the event has changed over time.

V. Performance Evaluation

We implement a middleware layer simulator focusing on the event detection and tracking to evaluate MEMS.

The performance metrics we consider are:

- Event Count Difference: the event count difference between the tracking results and the ground truth is used to show whether MEMS successfully detects all of the distinct events in the detection area.
- Event Membership Similarity: this metric is similar to the Jaccard Coefficient in data mining and is used to show whether each individual event is delineated correctly.
- Number of Tracking Robots per Event: the average number of tracking robots used to track an event during the event evolution is used to show whether MEMS uses a reasonable low number of tracking robots.
- Action Count: the number of rounds in 100 rounds that MEMS identifies event merge/split/create/destroy is compared with the actual actions to demonstrate the detection and tracking accuracy at a high level. In conjunction with the evaluation of event count difference and membership similarity, it can evaluate the event evolution accuracy.

We model events as follows: the number of events equals the initial event count and are generated at the beginning of the simulation, each with a random event size no larger than the maximum event size and a random event speed no larger than the maximum event speed. During the simulation, the events move individually with varying direction and speed no larger than the maximum speed in the detection area until merges or splits happen. Once a merge happens, the events merged into one event will have the same movement pattern. Once a split happens, the events will have individual movement patterns. Also, there are certain chances of event creation and event destroy in each round. We present several sets of simulation results to show the performance of MEMS. For all simulations, the maximum robot speed is set to 10m/s. As discussed in section II, there are no competing algorithms that can be used to compare the performance with MEMS, we only discuss the performance of MEMS itself in this paper.

Fig. 4 demonstrates the impact of the detection area size when the maximum event size is set to 2units and the initial event count is set to 20. In general, MEMS provides very high accurate event tracking (event count difference less than 3 out of 20, event membership similarity higher than 0.85) with modest overhead (number of tracking robots per event less than 5) when the maximum event speed is 1m/s.

Fig. 5 demonstrates the impact of the maximum event size when the detection area size is set to 10units*10units and the initial event count is set to 20. Similarly, MEMS provides very high accurate event tracking (event count difference less than 4 out of 20, event membership similarity higher than 0.92) with modest overhead (number of tracking robots per event less than 6) when the maximum event speed is 1m/s.

Fig. 6 demonstrates the impact of the initial event count when the detection area size is set to 10units*10units and the maximum event size is set to 10units. MEMS also provides very high accurate event tracking (event count difference less than 5 out of 50, event membership similarity higher than 0.9) with modest overhead (up to 200 tracking robots for 50 events) when the maximum event speed is 1m/s.

The results on the impact of the maximum event speed are omitted due to space constraints. In general, event speed significantly affects MEMS performance. Further, we have observed that MEMS detects the similar number of event evolution actions (i.e., merge, split, create, and destroy) as the ground truth by examining the records of event evolution history during the simulations [22].

In summary, MEMS provides accurate event detection and tracking when the number of events in the area is small or the event speed is slow comparing to the robot speed, and it also provides accurate event evolution history.

VI. Conclusion

In this paper, we have presented MEMS, a novel pipelined approach for detecting and tracking events with dynamic event signatures. MEMS has been shown to be capable of identifying multiple events with low event count difference and high event membership similarity, while using a reasonable number of tracking robots. Further, it has also been verified to provide accurate event evolution history including event merge, split, create and destroy. Overall, MEMS is promising for phenomena monitoring applications using robotic sensor networks. Our future work will provide robust network layer support for robots communication in the presence of packet loss, and will also design the tracking paths adaptive to the event evolution while minimizing robots movement delays.

Acknowledgement

This project is supported in part by NSF grant CNS-0915574.

References

[1] N. Hubbell and Q. Han, "Detection and tracking of dynamic amorphous events in wireless sensor networks," in *SECON*, 2011.

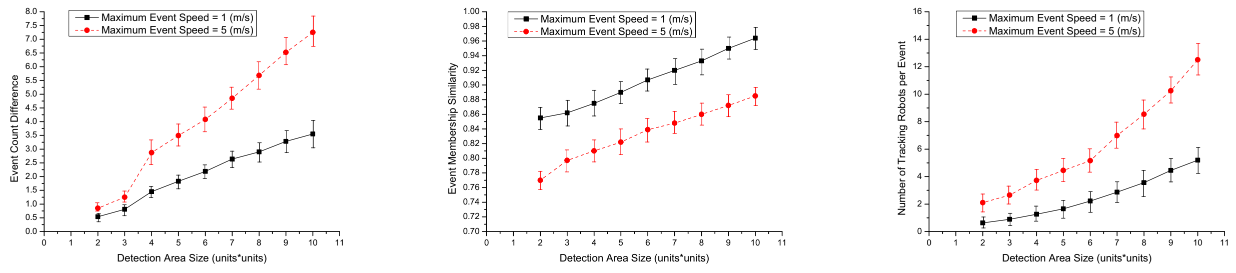
[2] R. Sarkar and J. Gao, "Differential forms for target tracking and aggregate queries in distributed networks," in *MOBICOM*, 2010.

[3] D. Zhang, Y. Liu, and L. M. Ni, "Rass: A real-time, accurate and scalable system for tracking transceiver-free objects," in *PERCOM*, 2011.

[4] X. Jiang, M. Li, Y. Yao, and L. J. Guibas, "Overcomplete radon bases for target property management in sensor networks," in *IPSN*, 2011.

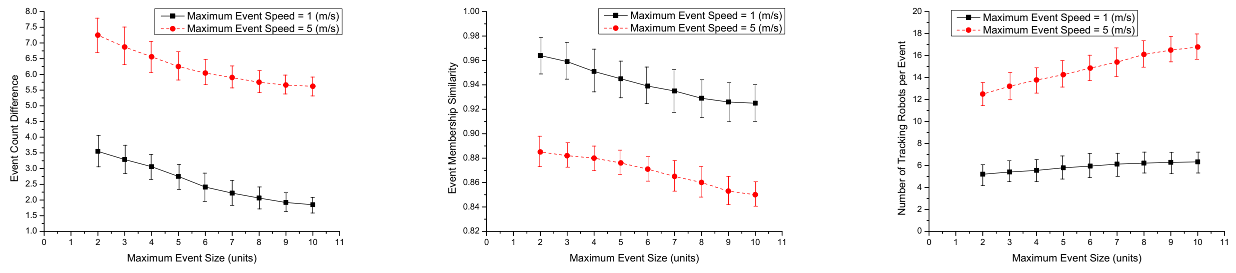
[5] X. Chen, A. Edelstein, Y. Li, and M. Coates, "Sequential monte carlo for simultaneous passive device-free tracking and sensor localization using received signal strength measurements," in *IPSN*, 2011.

[6] A. Liu, J. Bunn, and K. Chandy, "Sensor networks for the detection and tracking of radiation and other threats in cities," in *IPSN*, 2011.



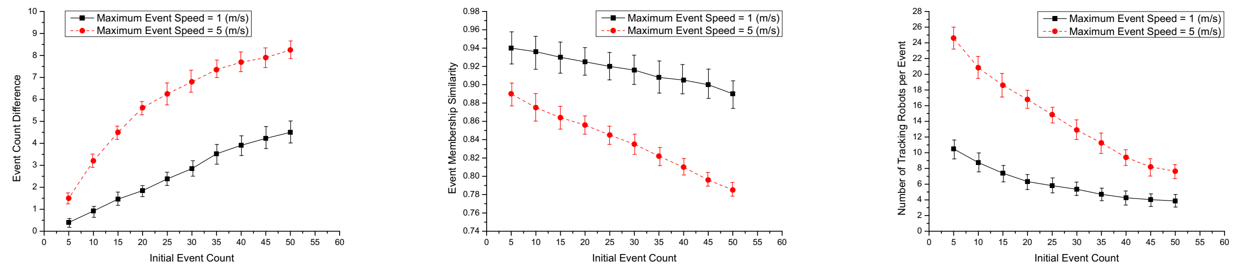
(a) (b) (c)

Fig. 4. Impact of Detection Area Size



(a) (b) (c)

Fig. 5. Impact of Maximum Event Size



(a) (b) (c)

Fig. 6. Impact of Initial Event Count

[7] E. Yanmaz and H. Guclu, "Stationary and mobile target detection using mobile wireless sensor networks," in *INFOCOM Workshop on Computer Communications*, 2010.

[8] O. Kosut, A. Turovsky, J. Sun, M. Ezovski, and L. Tong, "Integrated mobile and static sensing for target tracking," in *MILCOM*, 2007.

[9] H. M. La and W. Sheng, "Adaptive flocking control for dynamic target tracking in mobile sensor networks," in *IROS*, 2009.

[10] Y. Zou and K. Chakrabarty, "Distributed mobility management for target tracking in mobile sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, 2007.

[11] N. Kumaresh, "An empirical approach - distributed mobility management for target tracking in manets," *International Journal of Computer Applications*, vol. 2, no. 6, 2010.

[12] H. M. La and W. Sheng, "Multi-target tracking and observing in mobile sensor networks," in *TACS*, 2009.

[13] S. Kamath, E. Meisner, and V. Isler, "Triangulation based multi target tracking with mobile sensor networks," in *IEEE International Conference on Robotics and Automation*, 2007.

[14] A. Sinha, A. Tsourdos, and B. White, "Monitoring the dispersion of a contaminant cloud in an urban region by a swarm of uav sensors," in *IFAC Workshop on Networked Robotics*, 2009.

[15] S. Subchan, B. A. White, A. Tsourdos, M. Shanmugavel, and R. Zbikowski, "Dubins path planning of multiple uavs for tracking contaminant cloud," in *the 17th World Congress*, 2008.

[16] B. A. White, A. Tsourdos, I. Ashokaraj, S. Subchan, and R. Zbikowski, "Contaminant cloud boundary monitoring using network of uav sensors," *IEEE Sensors Journal*, vol. 8, no. 10, 2008.

[17] M. A. Kovacina, D. palmer, G. Yang, and R. Vaidyanathan, "Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles," in *IROS*, 2002.

[18] M. Scheutz, P. Schermerhorn, and P. Bauer, "The utility of heterogeneous swarms of simple uavs with limited sensory capacity in detection and tracking tasks," in *IEEE Swarm Intelligence Symposium*, 2005.

[19] H. Choset, "Coverage for robotics-a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, 2001.

[20] M. Li and Y. Liu, "Iso-map: Energy-efficient contour mapping in wireless sensor networks," *IEEE Transactions on Knowledge and data Engineering*, vol. 22, no. 5, 2010.

[21] wikipedia, "Closest pair of points problem," http://en.wikipedia.org/wiki/Closest_pair_of_points_problem.

[22] N. Yu and Q. Han, "Technical report of mems," http://thor.mines.edu/~nyu/file/MEMS.pdf.