Received April 22, 2019, accepted May 16, 2019, date of publication May 24, 2019, date of current version June 5, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2918765*

# Multilevel Object Tracking in Wireless Multimedia Sensor Networks for Surveillance Applications Using Graph-Based Big Data

**CIHAN KÜÇÜKKEÇECI[1], (Member, IEEE), AND ADNAN YAZICI[2], (Senior Member, IEEE)**
[1]Department of Computer Engineering, Middle East Technical University, 06800 Ankara, Turkey
[2]Department of Computer Science, School of Science and Technology, Nazarbayev University, 010000 Astana, Kazakhstan

Corresponding author: Cihan Küçükkeçeci (cihan.kucukkececi@ceng.metu.edu.tr)

**ABSTRACT** Wireless Multimedia Sensor Networks (WMSN), for object tracking, have been used as an emerging technology in different application areas, such as health care, surveillance, and traffic control. In surveillance applications, sensor nodes produce data almost in real-time while tracking the objects in a critical area or monitoring border activities. The generated data is generally treated as big data and stored in NoSQL databases. In this paper, we present a new object tracking approach for surveillance applications developed using a big data model based on graphs and a multilevel fusion. Our approach consists of three main steps: intra-node fusion, inter-node fusion, and object trajectory construction. Intra-node fusion exploits the detection and tracking of objects in each sensor, while inter-node fusion uses spatio-temporal data and neighboring sensors. Then, the fused data of all sensor nodes are combined to construct global trajectories of the detected objects in the monitored area on the WMSN. We implemented a prototype system and evaluated the performance of the proposed approach with both a real dataset and a synthetic dataset. The results of our experiments on the two datasets show that the use of third-level fusion in addition to inter-node and intra-node fusions provides significantly better performance for object tracking in the WMSN applications.

**INDEX TERMS** Big data, graph model, multilevel fusion, object tracking, wireless multimedia sensor networks.

## I. INTRODUCTION

Wireless Multimedia Sensor Network (WMSN) is a very effective technology commonly used to monitor the real world and collect data for analysis. With growing interest in emerging technologies such as the Internet of Things, Industry 4.0 and intelligent solutions, WMSNs have begun to be widely used in various application domains, such as healthcare [1], military [2], defense [3] and industry [4]. Numerous studies on target detection, object tracking, enemy intrusion, environmental detection and border surveillance have been conducted over the last decade. One of the most difficult tasks is tracking objects, be they humans, animals, vehicles or other moving objects, in a guarded area. [5], [6]. Object tracking is crucial for a surveillance system with dozens of cameras recording 7/24 with [7] or a border monitoring system [8], but also various Quad-copters to identify and track objects [9].

In the literature, some of the research studies focus on human tracking using audio and video to fuse multimodal features [10], [11]. Luo *et al.* [12] develop an indoor localization and tracking algorithm using network signals (WSN and WiFi) and predict the trajectory of moving objects. Others focus on animals to analyze their movements [13], [14]. Colceriu *et al.* [15] use mobile sensors worn by people in trains to display traces of railway maps. In general, using GPS data and mobile phone data to track moving objects and analyze performance is a common approach. Yuan *et al.* [16] use taxi trajectories to model the traffic patterns and suggest users the fastest route to their destination. Xu *et al.* [17] propose a method of detecting and tracking several humans based on their heads as rigid parts of the body. Recently, online footprints of users of social media activities are also used to track people [18].

The associate editor coordinating the review of this manuscript and approving it for publication was Takahiro Hara.

In addition, object tracking can also be performed by different types of sensors, from passive infrared motion sensors [19], [20] to cameras equipped with acoustic sensors [21], or RF sensors [22] to acoustic sensors [23], [24]. But generally, the fusion of multiple modalities, such as audio and video, is used to track objects to estimate their position. Zhou and Aggarwal [25] present an object tracking approach using multiple cameras with extracted features including spatial position, shape, and color information. They use the extended Kalman filter to improve the accuracy of their tracking method. The Kalman filter is widely used for multi-target tracking if the number of targets is low [26], [27]. Another research related to the use of multiple cameras is proposed by Kang *et al.* [28] to track the crowded scenes. Their approach is to integrate multiple views into a joint probabilistic data association filter for monitoring overcrowded people.

Particle filter tracking for wireless sensor networks is presented in [29] and another approach using the Bayesian filter is provided in [30]. Berclaz *et al.* [31] offer the shortest path algorithm for solving the multi-target problem in tracking and Oh *et al.* [32] develop a multi-target hierarchical tracking algorithm based on Markov chain Monte Carlo Data Association (MCM-CDA).

Energy efficiency in object tracking, a major problem in WMSNs, has also been studied by many researchers [33], [34]. We assume that the network and sensor nodes are selected and positioned in the most efficient way, as we focus primarily on the big surveillance data and the tracking of moving objects using this data.

To increase the tracking accuracy, a number of researchers have suggested multilevel fusion [35], [36]. The first detects the object using sensors and generates the local trajectories for a limited area. The latter combines them to construct the final trajectories. Fayyaz [37] provides a comprehensive survey and categorizes tracking based on network architecture, tracking algorithms, sensor types, the number of objects tracked, and wireless communication technologies. In addition, Mazimpaka and Timpf [38] focus on tracking aspects of trajectory extraction methods and applications. They point out that coping with massive big data is a matter of open research for tracking objects. Valsamis *et al.* [39] compare several real-time big data predictive analytics for trajectory prediction in the literature. They use a pre-trained model based on traditional multi-scan machine learning algorithms and focus solely on the trajectory of vessels by analyzing real-time, surveillance and spatio-temporal time series.

In this paper, we present a new object tracking method for surveillance applications developed using a big data model and a multilevel fusion. Our approach is based on three main algorithms: intra-node fusion, inter-node fusion, and object trajectory construction. Intra-node fusion exploits the detection and tracking of objects in each sensor. The output of the intra-node fusion is used with spatio-temporal data as well as neighboring sensors to generate inter-node trajectories. Then, the fused data from all the sensor nodes are combined to construct global trajectories for the detected objects in the monitored area on the WMSN. The proposed method tracks moving objects, such as vehicles, animals and humans, using wireless multimedia sensors for surveillance purposes.

We model the environment and WMSN data using a graph-based model to manage various application domains [40]. We store and manage surveillance sensor data using a NoSQL graph database system that contributes to the scalability of our approach. In addition, an unsupervised machine learning approach is used for tracking objects and there is no need to pre-train with a large dataset. We implemented a prototype system and evaluated the performance of the proposed approach with a real dataset, GeoLife Trajectories [41] and a synthetic dataset generated from the simulator that we developed for this study. Our experimental results on both datasets show that the use of third-level fusion in addition to inter-node and intra-node fusions provide significantly better performance for object tracking. The main difference between our work and previous studies in the existing literature is that they use the track identification created at the first level and continue to use this track identification throughout the process. But here we propose a more realistic approach that uses no pre-identification. Our approach extracts all trajectory information using spatio-temporal data, speed, direction, and low-level features.

The rest of the paper is organized as follows: The following section presents the formulation of the problem. Section III gives an overview of the system architecture and Section IV presents the proposed object tracking approach. Experimental results based on real and simulated synthetic datasets, as well as their evaluation, are presented in Section V. Finally, conclusions are drawn and future work is discussed in Section VI.

## II. PROBLEM FORMULATION

The surveillance area is deployed with a WMSN, which is a wireless distributed sensor network consisting of a set of multimedia sensor nodes. These nodes are connected to each other or connected to the main gateways using a wireless communication protocol. Our goal is to track moving objects using only sensor nodes with a camera and scalar sensors such as seismic, acoustic and PIR sensors.

Suppose that $N$ represents the number of sensor nodes deployed in a grid of the surveillance zone, where $w$ is the width and $h$ is the height of the zone. The grid approach used in this study is also used in [42], [43] because it is commonly applied to monitor the entire area without leaving gaps between the sensor nodes. In addition, we also deploy the sensor nodes randomly to see how the performance of the proposed tracking method is performed.

Each sensor node has a fixed position $(x_s, y_s)$ and it is assumed that the sensor nodes are not moving. At some point, an object enters the coverage area of a sensor node and is detected by the sensor node. In each time slot, the sensor node detects the object with different values based on its features and the distance that separates it from the sensor node. We can model the object detected by a sensor node with the following vector:

$$O = [(x, y), t, vx, vy, s, a, p] \tag{1}$$

where

- $(x, y)$ is detected object's location referenced to a sensor node
- $t$ is the discrete time
- $vx$ is the velocity on the X axes of the object in meter/seconds
- $vy$ is the velocity on the Y axes of the object in meter/seconds
- $s$ is the object's seismic vibration
- $a$ is the object's sound effect as an acoustic value in dB
- $p$ is the PIR sensor value which represents motion detected or not

We assume that the state evolves according to the constant velocity model (CV):

$$Ov_t = [x_t, y_t, vx_t, vy_t] \tag{2}$$

$$Ov_{t+1} = F_t.Ov_t \tag{3}$$

where $Ov_t$ is the velocity model and

$$F_t = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where $\Delta t$ is the sampling time interval between two successive measurement times $t$ and $t + 1$.

## A. OBJECT LOCALIZATION

As mentioned above, sensor nodes are assumed to have fixed positions and detect objects through its scalar sensors and cameras. When an object enters the coverage area of a sensor node, the sensor node camera takes a snapshot of the object (*image*) and extracts a silhouette. The location of the object in the field of view of the camera ($L_{fov}$) is calculated by our localization algorithm according to the local coordinate system of the image and other parameters [44].

$$L_{fov} = (x_{fov}, y_{fov}) \tag{5}$$

where $0 \leq x_{fov} \leq image_{width}$ and $0 \leq y_{fov} \leq image_{height}$.

From the local coordinate system of the image to the well-known WGS84 coordinate system (longitude, latitude), object localization is done using our previous work [44]. The translated location information ($L$) is used for trajectory fusions.

## B. SIMILARITY

In classical mechanics, a trajectory is a series of coordinates. Let $T_i$ and $T_j$ be the two trajectories with size K and M as shown below:

$$T_i = ((x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_k, y_k)) \tag{6}$$

$$T_j = ((x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_m, y_m)) \tag{7}$$

Euclidean distance between two coordinates $a$ and $b$ is calculated as below:

$$d_E(a, b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \tag{8}$$

Hausdorff's distance [45] is a measure of similarity, using coordinates in the metric space, of two nonempty trajectories. It measures the relative position of each coordinate in a trajectory with that of another. The Hausdorff distance is defined as

$$H(T_i, T_j) = max(h(T_i, T_j), h(T_j, T_i)) \tag{9}$$

where

$$h(T_i, T_j) = max_{a \in T_i} min_{b \in T_j} d_E(a, b) \tag{10}$$

and $a$ is a coordinate in the trajectory $T_i$ and $b$ is a coordinate in the trajectory $T_j$.

Dynamic Time Warping (DTW) compares the trajectories of different lengths by finding time warping that minimizes the total distance between the corresponding coordinates [46].

To align two trajectories using DTW, we construct an n-by-m matrix where the (x,y) element of the matrix contains the Euclidean distance $d_E(x', y')$ between the two points $x' \in T_i$ and $y' \in T_j$. A warping path $W$, is a contiguous set of matrix elements that define a mapping between $T_i$ and $T_j$.

$$DTW(T_i, T_j) = \min \sqrt{\sum_{n=1}^{k} w_n} \tag{11}$$

$$\gamma(x, y) = d_E(x', y') + \min(\gamma(x-1, y-1), \gamma(x-1, y), \gamma(x, y-1)) \tag{12}$$

where $w_n$ is the $n$th element of $W$ and $\gamma(x, y)$ is the cumulative distance which is calculated very efficiently using dynamic programming to prevent construction of the whole matrix.

## C. PROBLEM DEFINITION

We assume that there are $N$ number of sensors capable of detecting objects at any time and that all measurements are collected at the collector node, which includes a database of NoSQL graphs. Let $M$ be a measurement with the same timestamp $t$ from $n$ sensor nodes that detect the moving object.

$$M_t = \{m_1(t), m_2(t), \ldots, m_n(t)\} \tag{13}$$

where $m_i(t)$ is a measure at the $i$th sensor node. The problem is to extract the state of the object using measurements from time 0 to $t_{max}$ and to specify the global trajectory $T$ using the measurement history to detect anomalies if an object deviates from usual global trajectories.

## III. SYSTEM ARCHITECTURE

To manage the big data and high throughput of many wireless multimedia sensors, we design a scalable system architecture supported by NoSQL databases and messaging queues. Fig.1 shows a visualization of our multilayer system architecture composed of four layers; Detection Layer, Message Layer,

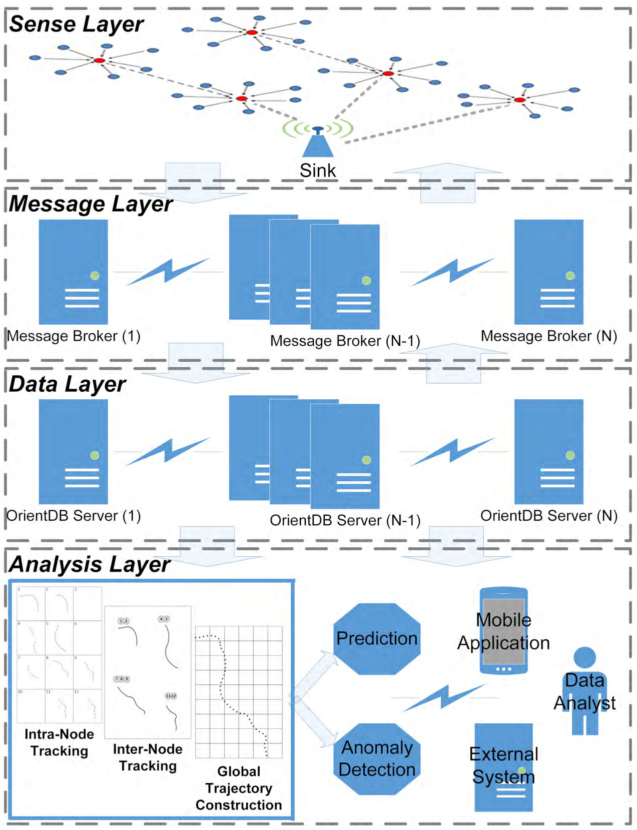


**FIGURE 1.** System architecture.

Data Layer, and Analysis Layer. Each layer is responsible for different parts of the entire process of analyzing big data in the proposed architecture.

The detection layer is composed of sensor nodes, gateways, and a sink. The raw data generator, producing synthetic data with position, PIR, seismic and acoustic information, simulates the sensor nodes activated by a moving object. Raw data is created for each sensor node with the sensor data calculated as a function of the distance to the position of the moving object. The detected data is collected at the sink using the message brokers that are positioned in the message layer. In order to cope with bottlenecks due to the high throughput of streaming sensor data, we position message queues between each process. The message data is orchestrated and sent to the data layer for retention by the message brokers.

In the data layer, NoSQL databases built using the OrientDB graph database system are clustered to process big graph data. The graph structure better supports connectivity analysis. Because object tracking is completely tied to the connection between each piece of data that is detected, graph models work better by analyzing the relationship between different sensor nodes or by identifying anomalies in a trajectory. Because the graph-based big data model allows you to query based on node attributes and relationships between them, a sophisticated prediction model can be developed and integrated into different domain applications.

In order to choose the most appropriate graph-based database system for our WMSN application, we use and compare the two well-known graph database systems, Neo4j and OrientDB [40]. OrientDB seems to be more convenient and efficient than Neo4j for our specific application. Because the streaming data is distributed by the message layer, the data layer must be aligned with the clustered big data architecture.

Finally, in the analysis layer, the data stored in the graph database is used for data analytics by the data scientist, the external system, or the mobile application for end-user reports and live portals. The object tracking algorithms we propose are executed in the analysis layer.

## IV. OBJECT TRACKING

Our object tracking approach is developed specifically for the surveillance requirements, but we believe that it can also be easily adapted to other application areas. An object's tracking is mainly focused on detecting an object when it enters the surveillance zone, from the moment it enters the zone until it leaves it. The tracking begins when the object is detected by a sensor node using its physical sensors. According to the algorithm implemented on the sensor nodes, a camera is activated if an object is detected or if it is a false alarm caused by noise in the environment. Next, a snapshot of the detected object is taken to ensure the possible presence of an object. The snapshot of the connected camera is important for accurately detecting and then tracking objects, as this multimedia data is used to estimate the approximate position of the detected object using object localization algorithms, which use the position of the sensor node, including ground height, camera lens specifications, and camera viewing angle.

The challenge of tracking an object without an associated identifier, such as GPS or RFID, is that each physical event occurring on a sensor deployed on a node is completely anonymous. Our goal is to look for relationships and correlations between entire anonymous sensor data and to identify not only the objects but also their movement over the time passing through the surveillance zone.

The main idea of our proposed object tracking approach is to make use of the position and time of the sensor node in accordance with the fusion outputs, i.e., the concept types, the features of the low-level, and speed of the object to solve the problem of whether the object has already been detected or whether it is a completely new object. The object tracking algorithm that we developed solves this problem by using the multilevel fusion approach with fuzzy rules. The first level of the fusion is the intra-node fusion which processes data for each sensor node and the second level is the fusion between the neighbors of a sensor node. Finally, the third level fusion is the construction of the object's trajectory in order to finalize the tracking and to extract the global trajectories of the objects. In the following subsections, we explain the algorithms as well as the fuzzy rule engine used in each fusion level.

### A. INTRA-NODE FUSION

Intra-node fusion is the first level trajectory fusion that uses the trajectories detected by a sensor node. The last object of each trajectory is analyzed with the current object to check if it corresponds to the existing trajectories, which are stored as time-ordered trees.

During the analysis, the geodesic distance between the current object and the last object of the trajectory is calculated. The direction of movement and the elapsed time between two objects are used during the analysis. The speed of the object is referenced according to the object type, but the actual speed is calculated using the elapsed time and distance.

**Algorithm 1** Intra-Node Fusion Algorithm

```
 1: procedure IntraNodeFusion($n$, $TT[]$)
    ▷ The trajectory tree-list ($TT[]$)] is used to identify if the
    sensed object ($n$) is already being tracked or a new object.
 2:     $matchFound \leftarrow false$                ▷ Initialize variable
 3:     for each trajectory $T$ in $TT[]$ do
 4:         $last \leftarrow$ Last sensed item in the trajectory $T$
 5:         $score \leftarrow ruleEngine(last, n)$
 6:         if $score > \lambda$ then
    ▷ New data belongs to an existing trajectory
 7:             $matchFound \leftarrow true$
 8:             insert $n$ into $T$
 9:             exit loop
10:         end if
11:     end for
12:     if $matchFound = false$ then  ▷ New trajectory found
13:         $T \leftarrow \{\}$
14:         insert $n$ into $T$
15:         insert $T$ into $TT[]$
16:     end if
17: end procedure
```

### B. INTER-NODE FUSION

Inter-node fusion is the second level trajectory fusion that looks for trajectory correlations between different sensor nodes. Each node is compared to its neighbor nodes. Moreover, only the nodes in the same direction of the compared trajectory are considered to optimize the performances. The inter-node fusion algorithm is presented in the Algorithm 2.

**Algorithm 2** Inter-Node Fusion Algorithm

```
 1: procedure InterNodeFusion($n$, $s$)
    ▷ Merges the trajectories identified by the sensor node $s$
    and its neighbor nodes using the new object ($n$).
 2:     $N[] \leftarrow findNeighbors(s)$
 3:     $\theta \leftarrow findDirection(L_s, L_n)$
 4:     for each sensor node $sn$ in $N[]$ do
 5:         $\theta_s \leftarrow findDirection(L_s, L_{sn})$
 6:         if $\theta_s$ in the same direction with $\theta$ then
    ▷ Only related neighbor which can detect the object is
    used
 7:             $T_s[] \leftarrow$ Trajectories of sensor node $s$
 8:             $T_{sn}[] \leftarrow$ Trajectories of sensor node $sn$
 9:             for each trajectory $t_s$ in $T_s[]$ and $t_{sn}$ in $T_{sn}[]$
    do
    ▷ Trajectories from two sensor nodes are compared to
    identify if they can be fused or not
10:                 $M_t \leftarrow$ Merge trajectories $t_s$, $t_{sn}$
11:                 $p \leftarrow \emptyset$
12:                 $c \leftarrow$ Last sensed item in the trajectory $M_t$
13:                 $same \leftarrow true$
14:                 while $same = true$ do
15:                     $p \leftarrow$ Previous item of $c$ in the trajec-
    tory $M_t$
16:                     if $p$ is $null$ then
17:                         exit loop
18:                     end if
19:                     $score \leftarrow ruleEngine(p, c)$
20:                     if $score < \lambda$ then
21:                         $same \leftarrow false$
22:                         exit loop
23:                     end if
24:                     $c \leftarrow p$
25:                 end while
26:                 if $same = true$ then
27:                     Apply merged trajectory $M_t$ in $T_s[]$
    and $T_{sn}[]$
28:                 end if
29:             end for
30:         end if
31:     end for
32: end procedure
```

### C. RULE ENGINE

We have designed and developed a generic rule engine for intra-node and inter-node fusions. The rule engine applies a list of rules on a given object for tracking purposes. Each rule is classified in a category that has a positive or negative effect on the score of the object being processed. Specifying different rules for the rule engine enables the use of the rule engine by different algorithms and fusion levels.

#### 1) DEFINITIONS

A rule ($R$) is a representation of an object state written in the form of a condition statement that gives a Boolean value (true/false). Each rule is associated with a rule category and defined as follows:

$$R = [cond, cat] \tag{14}$$

where

- $cond$ is the rule condition
- $cat$ is the rule category

A category of rules ($RC$) is an abstraction based on the basic properties of objects moving along trajectories. Categories allow us to group rules and assign different priorities between rules. A rule category is defined as follows:

$$RC = [e, \rho] \tag{15}$$

where

- $e$ is the identifier for positive or negative effect on the score
- $\rho$ is the weight of the rule category

The score ($S$) is the output of the rule engine that is calculated by applying all the rules for the movement of a detected object. The score value is an aggregation of all category scores and calculated using the following equations:

$$S = \sum_{n=1}^{i} f(\sum_{m=1}^{j} g(R_m, RC_n), RC_n) \tag{16}$$

where $i$ is the number of rule categories, $j$ is the number of rules and,

$$g(R, RC) = \begin{cases} 0, & \text{if R.cat} \neq \text{RC} \\ 0, & \text{if R.cat} = \text{RC}, \quad \text{R.cond} = \text{false} \\ 1, & \text{if R.cat} = \text{RC}, \quad \text{R.cond} = \text{true} \end{cases}$$

$$f(s, RC) = \begin{cases} \text{s} \times \text{RC}.\rho, & \text{if RC.e} = \text{positive} \\ \text{-s} \times \text{RC}.\rho, & \text{if RC.e} = \text{negative} \end{cases}$$

### 2) RULE GENERATION AND IMPLEMENTATION

We have developed a fuzzy logic rule generator that provides fuzzy rules generated semi-automatically using trained trajectories. The rules generated are then fine-tuned by experts.

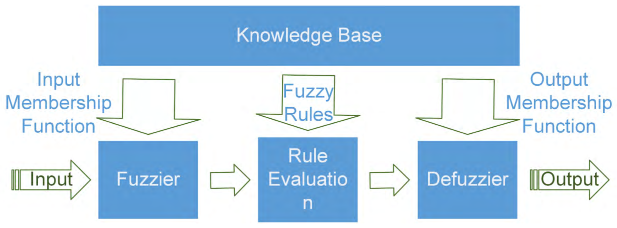


**FIGURE 2. Fuzzy logic system.**

As shown in Fig.2, the fuzzy logic system has four basic steps: fuzzification, rule evaluation, knowledge base, and defuzzification. During the fuzzification stage, the actual inputs are converted to fuzzy membership functions. The knowledge base provides the list of rules in our application domain. The big challenge is to design the correct list of rules for the problem. We use experts to filter and organize the rules generated automatically in the knowledge base. The fuzzy logic system inputs are being processed by the rule evaluation process, which uses the knowledge base to combine fuzzy inputs and produce outputs mapped to fuzzy membership functions. In the defuzzification stage, the fuzzy outputs of the rule evaluation step are mapped to crisp outputs.

Input member functions are created for each rule category. Fig.3 shows examples of fuzzy input member functions that are used to generate fuzzy rules. We can map crisp location data in meters to fuzzy values, such as "no change", "small change" "medium change" or "big change". Similar to the change of location, we compare the time difference between two sensor data and the map to fuzzy values.

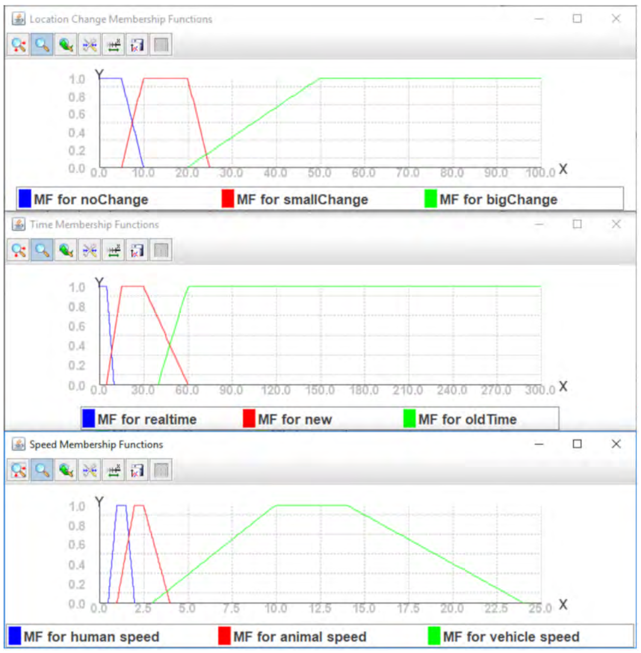


**FIGURE 3. Sample fuzzy member functions.**

After running the fuzzy rules generator on the training data, we have a list of generated rules that is the input for our domain experts. Manual filtering and a refinement process are performed to obtain the final list of the rules for the rule engine. The described semi-automatic rule generation approach produced rules for intra-node and inter-node fusion. Each rule is linked to one of the rule categories described below:

- Direction: The direction of movement of the detected object is used in the rules. (e.g., are they moving on the same direction?)
- Time: The time-based rules are associated with this category. (e.g., has the object been seen in last 2 hours?)
- Velocity: The speed of the object is used in the rules to calculate its possible location and decide whether it moves or waits in the same location. (e.g., how far this object can go with this speed in that specific time?)
- Feature: The low-level SIFT [47] features of silhouette image are used in the rules. (e.g., check the similarity of silhouette images!)
- Concept Type: The type of object concept is used in some rules. (e.g., check if the concept types are identical or not?)
- Distance: The distance-based calculations are used to create rules. (e.g., is it possible to move from previous location to current location in that specific time?)

### 3) SAMPLE SCENARIO EXECUTION

Suppose there are two sensor nodes $SN_1$ and $SN_2$ positioned in (2,0) and (2,4) in a grid-based area. Each sensor node maintains a local track store for recognized trajectories used in track fusion calculations.

Let the trajectory $T_1$ is in the local track store of the sensor node $SN_1$ while $T_2$ and $T_3$ are in $SN_2$. Three trajectories can be defined as arrays of location and time pairs like below;

- $T_1 = \{[(0,4) - 12:00:00], [(4,4) - 12:00:02]\}$
- $T_2 = \{[(4,2) - 12:00:01], [(3,0) - 12:00:07]\}$
- $T_3 = \{[(0,2) - 12:00:06], [(2,3) - 12:00:08]\}$

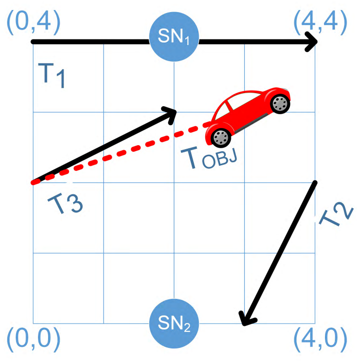


**FIGURE 4. Rule engine execution on a sample scenario.**

Fig.4 shows an example scenario area with sensor nodes ($SN_1$, $SN_2$) and recognized trajectories ($T_1$, $T_2$, $T_3$). The red dashed line represents the trajectory of the moving object ($T_{OBJ}$) where (3,3) is the last location of the object at 12:00:09.

Suppose there are three categories of rules and for each rule category, a rule is defined in the rule engine. The categories of rules can be defined as follows:

- ***RC-1:*** Velocity
  *e:* Positive(+), $\rho$: 0.4
- ***RC-2:*** Direction
  *e:* Positive(+), $\rho$: 0.3
- ***RC-3:*** Time
  *e:* Negative(−), $\rho$: 0.5

The rules can be defined as follows (refer to the Algorithm 3 for the variable definitions in *cond*);

- ***R-1:*** Check if the distance between the previous location and the location of the newly detected object is within the maximum and minimum limits that can be moved by the newly detected object
  *cond:* $d <= v_{max} \times \Delta t$ AND $d >= v_{min} \times \Delta t$
  *cat:* RC-1
- ***R-2:*** Check if the relative direction of the location of the newly detected object from the previous location is in the same direction as the trajectory
  *cond:* $\theta = p.direction$
  *cat:* RC-2

**Algorithm 3** Rule Engine Algorithm

```
1:  procedure RULEENGINE($p$, $c$)
        ▷ The last object in the trajectory, the previous object ($p$), and the current object ($c$) are compared to indicate whether the current object belongs to the trajectory or not.
2:      $L_c \leftarrow (c.x, c.y)$                    ▷ Location of the current object
3:      $L_p \leftarrow (p.x, p.y)$                    ▷ Location of the previous object
4:      $d \leftarrow distanceOnGeoid(L_c, L_p)$
5:      $\theta \leftarrow findDirection(L_c, L_p)$
6:      $v \leftarrow (p.vx, p.vy)$
7:      $\Delta t \leftarrow p.t - c.t$
8:      $R[] \leftarrow$ List of semi-automatically generated rules
9:      $S[] \leftarrow \{\}$                           ▷ Used for rule category scores
10:     for each rule $r$ in $R[]$ do
11:         $valid \leftarrow r.process(d, \theta, v, \Delta t, \varepsilon)$
12:         if $valid = true$ then                      ▷ Rule is accepted
13:             $i \leftarrow$ Index of rule $r$'s category in S[]
14:             $S[i]++$
15:         end if
16:     end for
17:     $score \leftarrow 0$
18:     for each rule category $ct$ do
19:         $i \leftarrow$ Index of rule category $ct$ in S[]
20:         if $ct.e = Positive$ then
21:             $score \leftarrow score + (ct.\rho * S[i])$
22:         else
23:             $score \leftarrow score - (ct.\rho * S[i])$
24:         end if
25:     end for
26:     return $score$
27: end procedure
```

**TABLE 1. Rule condition parameter values for each trajectory.**

| Parameter | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| $d$ | 1.4 | 3.0 | 1.0 |
| $v$ | 2 | 0.4 | 1.2 |
| $\theta$ | SW | N | E |

- ***R-3:*** Check whether the previous detection time is obsolete, as if it is earlier than the detection time of the new object movement
  *cond:* $\Delta t > 5$ (5 minutes)
  *cat:* RC-3

Table 1 shows the computed values of parameters which are used in rule conditions. If we recap the parameters, $d$ represents the distance between trajectory's last object and moving object. $v$ represents the velocity calculated by location and time pairs of the trajectory. For this sample scenario, $v_{max}$ and $v_{min}$ are computed by using a fixed 30% upper and lower bound on $v$. $\theta$ is the relative direction between trajectory's last object and moving object.

**TABLE 2.** Sample rule engine execution scores for each trajectory.

| | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| *R-1* | 0.4 | 0.0 | 0.4 |
| *R-2* | 0.0 | 0.0 | 0.3 |
| *R-3* | -0.5 | 0.0 | 0.0 |
| *Score* | -0.1 | 0.0 | 0.7 |

Finally, if we run the rule engine for the scenario shown in Fig.4, we have the scores in Table 2 for each trajectory. According to the final scores, the new object belongs to the trajectory $T_3$, which has a higher probability for the trajectory of the new object than the other possible trajectories.

### D. OBJECT TRAJECTORY CONSTRUCTION

The third level of the fusion is to cluster and construct the object trajectories generated at the inter-node fusion level using the rule engine outputs. Some inter-node trajectories may be fused into other trajectories because they may overlap or be complementary to one another. The purpose of this last level of fusion is to detect the final global trajectories in the monitored area. Global object trajectories can be used as input for intra-node and inter-node fusion. Another use is to detect anomalies if a newly detected object trajectory does not correspond to any learned global trajectory. Our clustering algorithm is an unsupervised learning algorithm in which relationships are discovered from an unlabeled dataset. We use the results of previous detections as input parameters for future calculations.

The construction of the trajectory of the object is shown in Algorithm 4. The basic logic of the algorithm is based on the calculation of Hausdorff Similarity [48] between trajectories with similar lengths. The calculated value is normalized in the interval [0, 1]. Higher measures indicate a high degree of similarity and we can identify that they can be grouped together in a global candidate trajectory. If several candidate trajectories are detected, they can construct a global trajectory. We use the Dynamic Time Warping (DTW) distance to compute the centroids of the global trajectory, which is supposed to be the center of the trajectories grouped in the actual K-Means way.

Fig.5 shows how to detect a step by step trajectory using our multilevel fusion approach. Suppose there are 12 sensor nodes that monitor an area and an object passes through the surveillance zone (a). In order to describe the detection of a trajectory of an object, we can use a metaphor of the puzzle game. The inter-node fusion algorithm detects many pieces that are connected, but they belong to different parts of the puzzle. Therefore, the algorithm is not aware of the situation as a whole (b). Each sensor node detects and fuses its local data, a range limited to a single node. Intra-node fusion is a phase in which you begin to merge different groups and form shapes or images in the puzzle, but the entire image is still missing (c). The sensor data from many sensor nodes are fused in the last step and finally the construction of the overall object trajectory is performed. Thus, with this last step, all the pieces of the puzzle are connected to each other and a certain number of trajectories are determined. These trajectories detected by many sensor nodes are then fused to identify the overall trajectory.

**Algorithm 4** Object Trajectory Construction Algorithm

```
1:  procedure OBJECTTRAJECTORYCONSTRUCTION($T[]$)
        ▷ Global object trajectories are constructed using the inter-node fusion generated trajectories.
2:      for each trajectory $t$ in $T[]$ do
3:          if $t$ is a short trajectory then
4:              Merge $t$ with other trajectories
5:          end if
6:      end for
7:      $GLB[] \leftarrow \{\}$                    ▷ Global trajectories list
8:      for each trajectory $t$ in $T[]$ do
9:          $id \leftarrow t.id$
10:         $glbTraj \leftarrow GLB[id]$
11:         if $glbTraj$ is $null$ then
12:             $glbTraj \leftarrow createNewGlobalTrajectory(t)$
13:             insert $glbTraj$ into $GLB[]$ with $id$
14:         end if
15:     end for
16:     for each global trajectory $gt$ in $GLB[]$ do
17:         $ML[] \leftarrow \{\}$
18:         $T[] \leftarrow similarLengthTraj(GLB[], gt.length)$
19:         for each trajectory $t$ in $T[]$ do
20:             $m \leftarrow hausdorfSimilarity(gt, t)$
21:             if $m > SimilarityThreshold$ then
22:                 Mark as candidate trajectory
23:                 insert $t$ into $ML[]$
24:             end if
25:         end for
26:         if $ML[].size > GlobalTrajectoryThreshold$ then
27:             $glbTraj \leftarrow kmeansDTWCluster(ML[])$
28:             insert $glbTraj$ into $GLB[]$
29:         end if
30:     end for
31: end procedure
```

## V. EXPERIMENTAL EVALUATION

We have prepared a test environment to test our proposed object trajectory approach. We have developed a simulator that uses the WGS84 geodetic datum to support real world scenarios. Our experiences are realized in two different aspects.

The first aspect was to visualize the performance of our algorithm for different types of scenarios using the Scenario Builder component of our simulator. In parallel with this, we also compared our test results with the Kalman filter tracking approach [49], [50]. With the second aspect of our experience, we evaluated the performance of our method with a real world data set. Because there are many public datasets, most tracking applications are just video data sets

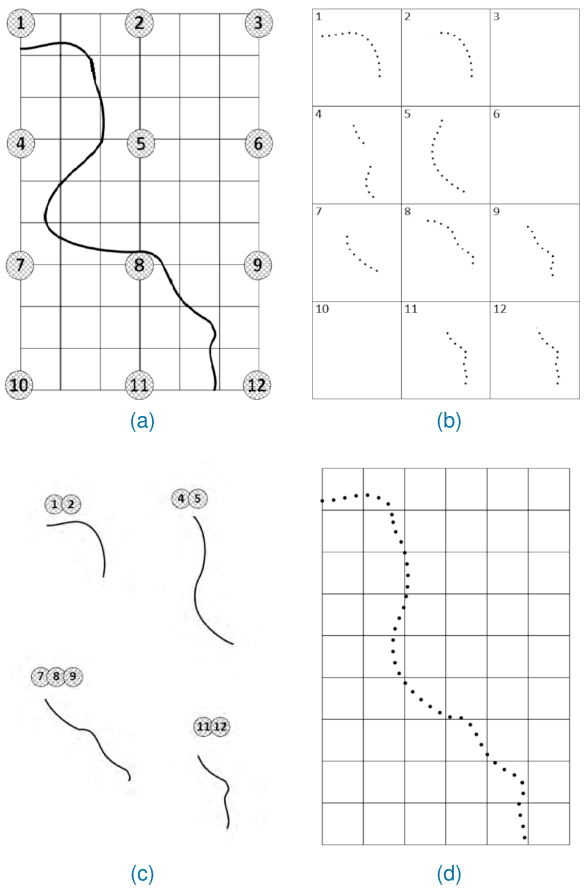


**FIGURE 5. Multilevel fusion step-by-step trajectory detection. (a) Actual trajectory. (b) Intra-node fusion. (c) Inter-node fusion. (d) Trajectory construction.**

such as Multi-Object Tracking [51] or dataset with device tag, such as GPS or RFID, that do not conform to our problem area. In addition, some data sets have focused on interconnected roads designed for wheeled vehicles and pedestrian traffic [52] but we are monitoring the rural area which has no information on roads. Therefore, to evaluate our algorithm with real data, we adopted the dataset in our application domain. We used the GeoLife trajectory data set because it has different types of transport modes and a wide variety of trajectory lengths.

Test environment's specifications are as follows;

- Intel i7-4710HQ Quad Core CPU
- 16 GB DDR3 RAM
- 240 GB SSD Storage
- 4 GB NVIDIA 860GTX GPU

### A. SIMULATOR AND SYNTHETIC DATA

Our simulator [53] is extended and improved to generate more realistic trajectory data for advanced analytics. The current version of the simulator consists of three main components; Network topology generator, sensor data generator, and scenario generator. The network topology generator is used to create a simulation for a WMSN infrastructure.

We assume that our WMSN is distributed with a grid layout in a simulation area. Since we have already evaluated the architecture of the system presented in Section III from the big data point of our previous study, [53], we focus on object tracking and reduce the time needed to perform hundreds of iterations for each experiment. We selected a testbed that is part of the territory of our university. Fig.6b shows a map view of the monitored area with 96 sensor nodes, composed of 4 different clusters with a gateway node and a sink node in the center. The distance between the two sensor nodes is estimated at about 60 meters.

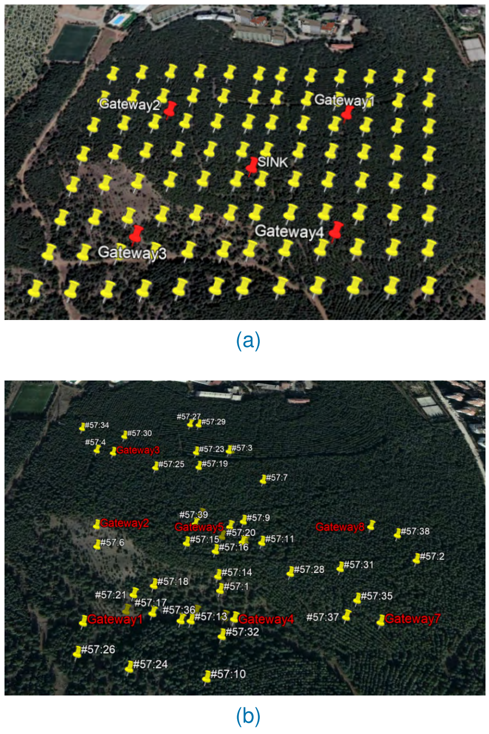


**FIGURE 6. Simulated wireless sensor network. (a) Grid-based deployment. (b) Random deployment.**

Since we have developed our method specifically for network-based wireless sensor networks, we are also measuring the performance of our method for randomly distributed wireless sensor networks. That is, we have upgraded our simulator to randomly deploy the sensor nodes and generate synthetic data accordingly (Fig.6a). To make it more realistic and well distributed, we position 10% of the nodes from each other and distribute the rest of the nodes without any limitation, except that two sensor nodes cannot be closer than 2 meters.

Sensor Data Generator produces synthetic data with position, PIR, seismic and acoustic data. The data flow is simulated as if the data were detected by sensor nodes, close to the position of the data generated. Raw sensor data is created for each sensor node with the scalar data calculated as a function of the distance to the position of the raw data. From a sensor node to a gateway and from a gateway to a sink node,

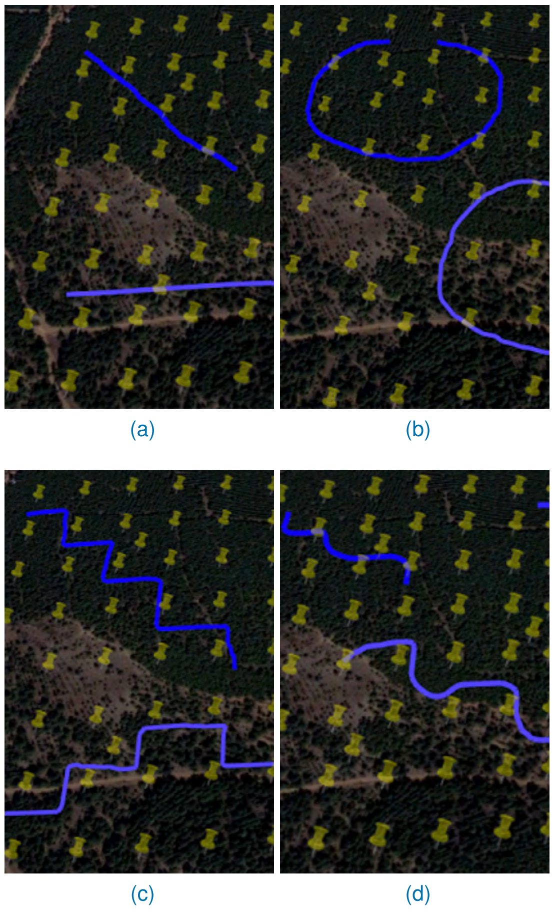


**FIGURE 7.** Scenario samples generated using developed simulator. (a) Straight trajectories. (b) Circular trajectories. (c) Zig zag trajectories. (d) Wavy trajectories.

the data stream comprising the multilevel data fusion process is simulated to generate more realistic data.

Fig.7 shows example scenarios created using Scenario Builder, one of the main components of our simulator. We created 21 different scenarios using different trajectory models and concept types to evaluate the performance of our method. We used the following formulas to calculate the precision/recall values and the F-measure.

$$Precision = \frac{TP}{TP + FP} \quad (17)$$

$$Recall = \frac{TP}{TP + FN} \quad (18)$$

where $TP$ is the number of true positives, $FP$ is the number of false positives, and $FN$ is the number of false negatives. For a better evaluation, we use F-measure or balanced F-score that combines precision and recall values in a metric.

$$F = 2.\frac{Precision.Recall}{Precision + Recall} \quad (19)$$

Fig.8 represents the result of experiments by calculating the F-measure scores of each scenario group for both grid-based and random deployment of sensors. Our method detects

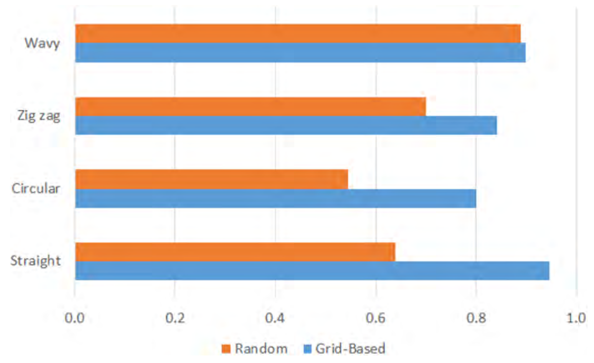
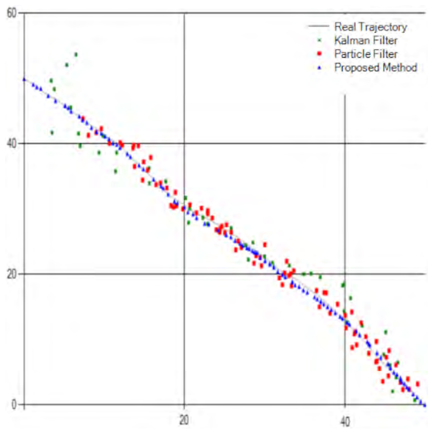


**FIGURE 8.** F-measure scores for both grid-based and randomly deployed nodes.

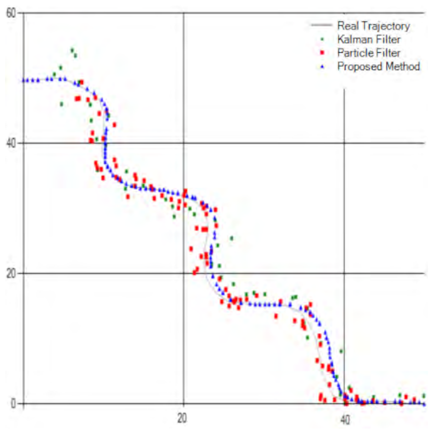
straight trajectories because they are easy to track with respect to more complex trajectories. The detection performance of the circular and zigzag trajectories are slightly lower than those of the other types. The zigzag pattern has sharp turns and is not expected and treated as new objects. For circular patterns, the behavior is not expected to move back and forth at the same point. However, the overall performance looks promising. From the point of distribution of the nodes, if we deploy the sensor nodes randomly, the scores decrease. The loss of performance in random positioning has two main causes. First, there are undetected areas and if the trajectory of the object falls in this area, it causes disconnection. Second, some densely positioned areas create multiple possible trajectories for a single trajectory and make inter-node fusion more difficult to manage. To summarize, randomly deployed nodes are not as good as grid positioning, but this is an expected result.

Since Kalman filters and particle filters are widely used in object tracking [54], [55], we compare the performance of our method with these two filters. The Kalman filter tries to balance the motion model and the measurements to provide a better estimate of trajectories. It uses linear projections with Gaussian noise to increase efficiency while the particle filter uses a sequential Monte Carlo method. Both algorithms recursively update the state estimate. Kalman filter uses the system model and the sensor observations to estimate the current state from the previous states. Particle filter uses random sampling to generate different system states, then assigns high weights to the states supported by the sensor data.

The results of performance comparisons and error rates are given in Fig.9 and 10. The results show that our proposed method better detects the trajectory of the object, with error rates lower than those of Kalman and particles filters. In addition, it appears from the experimental results that even object maneuvers, our object tracking algorithm can continue to track objects with a relatively low error rate. Kalman filter and particle filter algorithms need early recovery time to produce optimized predictions. For this reason, some of the initial values of the particle filter are ignored.



(a)



(b)

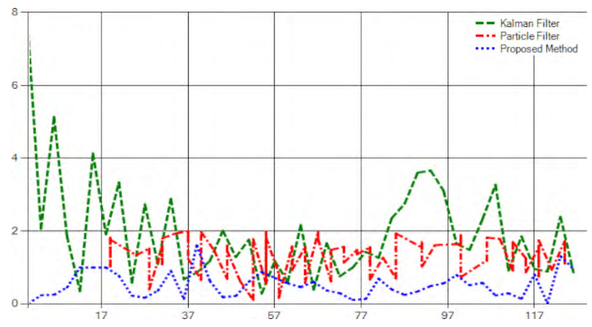
**FIGURE 9.** **Comparison with Kalman filter and particle filter. (a) Straight trajectory. (b) Wavy trajectory.**

## *B. REAL WORLD DATASET*

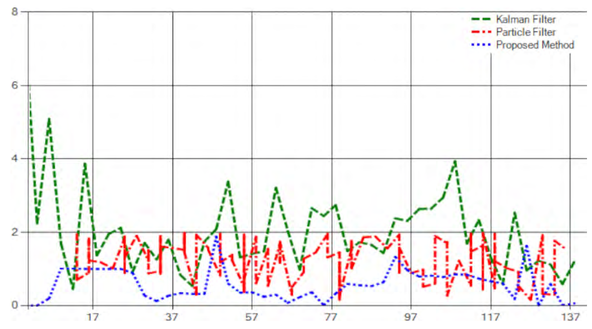
To test our algorithm with a real dataset, we use GeoLife Trajectory Dataset [41], a GPS trajectory dataset from the Microsoft Research GeoLife project, collected by 182 users from April 2007 to August 2012. This set of data contains 17,621 trajectories, with a total distance of 1,251,654 kilometers and a total duration of 48,203 hours. The trajectories are recorded in a dense representation that is every 1-20 seconds per point.

The GeoLife Trajectory dataset is not exploited by multiple sensors. Instead, each trajectory corresponds to a single object carrying a GPS device. To make it usable for our experiments, we transform the original trajectories into sensor data that can be detected by several sensors via our simulator. The simulator reads GeoLife's actual trajectory as an input and generates motion at the exact location of the actual trajectory based on the type of object. Then, the sensor nodes that can detect motion at that location are triggered as if there is an object.

In order to evaluate the performance of our proposed object tracking algorithm, we use a confusion matrix. The selection of the $\varepsilon$ and *NodeBuffer* parameters affects the sensitivity of the algorithm. $\varepsilon$ is the threshold of the distance between the actual distance and the calculated distance using speed and time. *NodeBuffer* is the parameter to specify the number of historical sensor data used to identify new sensor data. that is, whether these data correlate with previous detections or not.



(a)



(b)

**FIGURE 10.** **Error rates comparison with Kalman filter and particle filter. (a) Straight trajectory. (b) Wavy trajectory.**

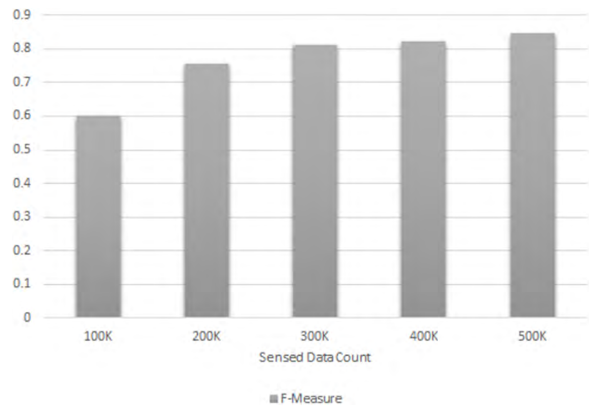
**TABLE 3.** **Algorithm scores with various parameters.**

| $\varepsilon$ | NodeBuffer=5 | NodeBuffer=10 |
|---|---|---|
| 0.35 | 0.322058681 | 0.483088022 |
| 0.40 | 0.999631145 | 0.999631145 |
| 0.45 | 0.999631145 | 0.481668622 |
| 0.50 | 0.499815572 | 0.321112415 |

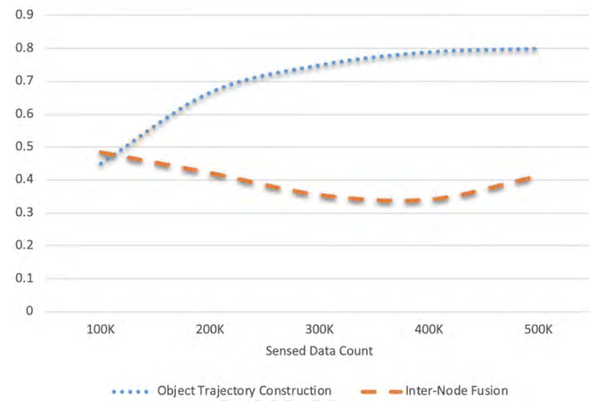
The scores of the parameters are calculated by normalizing the Hausdorff Distance between real trajectory and extracted trajectory with the number of tracks found. Table 3 illustrates the computed scores for different parameter values of $\varepsilon$ and *NodeBuffer*. The results show that our tracking algorithm works the best with $\varepsilon = 0.40$. *NodeBuffer* depends on the $\varepsilon$ parameter as a pivot. For bigger $\varepsilon$ values *NodeBuffer* is negatively correlated and with lower $\varepsilon$ values *NodeBuffer* is positively correlated.

Therefore, we use parameters $\varepsilon = 0.40$ and *NodeBuffer* = *5* as default values to monitor the performance of our algorithm in GeoLife Trajectories dataset.

The measurement scores F are given in Fig.11a for different sizes of datasets. The algorithm duplicates small pieces



(a)



(b)

**FIGURE 11. Object trajectory construction. (a) F-Scores. (b) Precision comparison with inter-node fusion.**

of longer tracks as different tracks. As a result, tracks are correctly detected with smaller, noisy tracks. This provides lower accuracy performance, but the effect of noisy data decreases as more sensor data is processed.

To show the positive effect of multilevel tracking, especially that of constructing the trajectory of the object, we measure the results for fusion with and without third level trajectory fusions. Improvement of the performance of inter-node fusion over intra-node fusion has very little effect since it is impossible to continuously track an object without inter-node fusion. Indeed, this is because intra-node tracking is simply limited by the detection capability of the sensor nodes. The result of our experiments relating to the effect of the construction of the trajectory of the object is presented in Fig.11a. From the experimental results, it has been observed that the construction of the object's trajectory significantly improves the overall performance of our object tracking approach.

In order to see the improved performance of our object tracking method with different types of concepts, we map the transport mode labels in the dataset to our concept types. Table 4 shows the mapping of the concept types "Vehicle" and "Human" because there is no label associated with the type of concept "Animal" in the dataset.

Table 5 displays the F-measure scores for a selected set of datasets based on the scenarios chosen for validating

**TABLE 4. Mapping from transportation mode to concept type.**

| Transportation Mode | Concept Type |
|---|---|
| Walk | Human |
| Car | Vehicle |
| Taxi | Vehicle |
| Bus | Vehicle |
| Train | Vehicle |

**TABLE 5. F-measure comparison for concept types.**

| Dataset ID | Vehicle | Human |
|---|---|---|
| 062 | 0.89 | 0.91 |
| 085 | 0.87 | 0.87 |
| 128 | 0.89 | 0.91 |
| 153 | 0.91 | 0.93 |

our method. These datasets have trajectories of the concept types "Vehicle" and "Human". Depending on the results obtained, the trajectories belonging to the human type are well detected with respect to the type of vehicle. The best performance of human objects is the difference in speed between moving objects. Since a human typically moves more slowly than a vehicle, a sensor node can track humans more easily and more accurately than a vehicle. In other words, the vehicle sensor data collected by the sensors is much more scarce than the human sensor data.

### C. TRAJECTORY ANALYSIS

Extracted global trajectories are used to form well-known classification algorithms for the purpose of advanced analysis such as trajectory prediction and anomaly detection in real-time streaming data. The referenced classification algorithms are Random Forest [56], Decision Table [57], Sequential Minimal Optimization (SMO) [58], Naive Bayes [59], Logistic Regression [60], and Multilayer Perceptron (MLP) [61].

Fig.12a shows the trajectories trained for the analytics. We created three global trajectories that are similar at the beginning and then differentiate. Therefore, it is easier to see if the algorithm can correctly identify trajectory transitions or not.

#### 1) PREDICTION

Classification algorithms are used for prediction in many studies [62], [63]. We train the well-known classification algorithms using the global trajectories extracted by our proposed algorithm. Once the learning phase is over, we classify the new sensor data to predict the possible trajectory of the object.

In addition, we tune the parameters of the classification algorithms to increase the success rates of the predictions of the trajectories. Fig.13 shows the performance of each classification algorithm with and without tuned parameters. The success rate is calculated using the number of correct identifications of the trajectories. The SMO algorithm is more

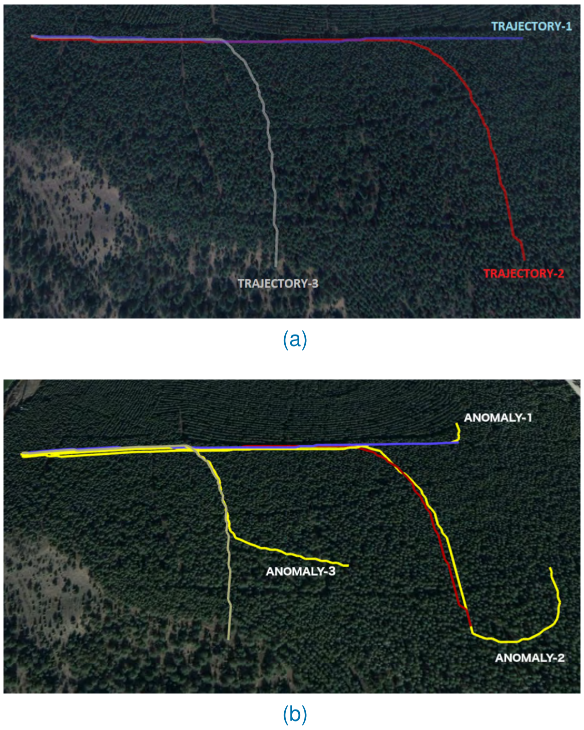


**FIGURE 12.** (a) Trained trajectories for analytics. (b) A sample anomaly in which the object leaves suddenly while moving on Trajectory-3.

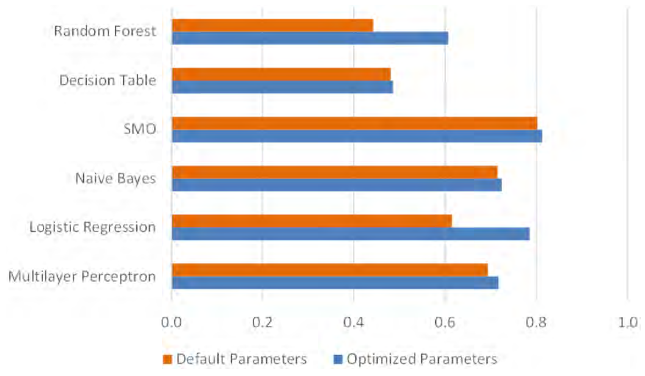


**FIGURE 13.** Success rates of classification algorithms in prediction with/without parameter tuning.

efficient than the other machine learning algorithms used in our experiments and which have been very successful. Moreover, it is important to emphasize that only the SMO algorithm can identify an equal possibility of overlapping trajectories, which is an important case for the prediction. Executing algorithms with optimized parameters gives better results than expected, but optimizing logistic regression is one of the most affected by overall success.

### 2) ANOMALY DETECTION

Another big data analytics that we study and evaluate experimentally is anomaly detection in real-time streaming data. By monitoring forecasts and actual data, we can determine whether an object is lined up on a path or not [64]. If the object deviates from the trajectory, we can suppose that there is an anomaly sign (Fig.12b).

**TABLE 6.** Anomaly detection in streaming data ([A]:Anomaly, [+]:Detected, [–]:Not detected).

| Algorithm | A-1 | A-2 | A-3 |
|---|---|---|---|
| Random Forest | + | + | + |
| Decision Table | + | + | + |
| SMO | - | - | - |
| Naive Bayes | + | - | + |
| Logistic Regression | + | - | + |
| Multilayer Perceptron | - | - | + |

Table 6 shows the success rates of anomaly detection for different movements. The Decision Table and Random Forest algorithms respond better to abnormal movements and sudden changes in anomaly classification results. SMO never notices the change of movement and does not detect any anomalies. The multilayer perceptron and the logistic regression do not perform well against the success of their predictions.

## VI. CONCLUSION

In this study, we propose a new unsupervised object tracking approach, developed using the big graph-based data model. We focus on tracking objects for surveillance applications. The object tracking approach includes three main algorithms: intra-node fusion, inter-node fusion, and object trajectory construction. To validate our algorithms, we performed several experiments on different scenarios using a real dataset, namely GeoLife Trajectories Dataset, and a dataset created synthetically using the simulator developed for this study. The results of our experiments on both datasets show that our object tracking approach is working well and is robust for use in other application domains. From the experiences we have observed, our multi-level fusion approach improves the performance of object tracking.

Trajectory analyzes, such as prediction and anomaly detection, are applied using extracted global trajectories. Well-known classification algorithms are compared for different trajectories and the experimental results show that the best algorithm for predicting all possible trajectories does not exist in all cases. When one algorithm works better in one case, it may not work better in another.

A future research topic that we plan to investigate is to automatically calculate the $\varepsilon$ and *NodeBuffer* parameters based on the dataset. In addition, as another possible research topic, the use of FastDTW instead of DTW can be studied to increase the performance of the construction of global trajectories.

## REFERENCES

[1] G. Hernández-Peñaloza, A. Belmonte-Hernández, M. Quintana, and F. Álvarez, "A multi-sensor fusion scheme to increase life autonomy of elderly people with cognitive problems," *IEEE Access*, vol. 6, pp. 12775–12789, 2017.

[2] M. Björkbom, J. Timonen, H. Yigitler, O. Kaltiokallio, J. M. V. García, M. Myrsky, J. Saarinen, M. Korkalainen, C. Çuhac, R. Jäntti, R. Virrankoski, J. Vankka, and H. N. Koivo, "Localization services for online common operational picture and situation awareness," *IEEE Access*, vol. 1, pp. 742–757, 2013.

[3] Z. Zhang, H. Zhu, S. Luo, Y. Xin, and X. Liu, "Intrusion detection based on state context and hierarchical trust in wireless sensor networks," *IEEE Access*, vol. 5, pp. 12088–12102, 2017.

[4] L. Shu, Y. Chen, Z. Huo, N. Bergmann, and L. Wang, "When mobile crowd sensing meets traditional industry," *IEEE Access*, vol. 5, pp. 15300–15307, 2017.

[5] H. T. Kung and D. Vlah, "Efficient location tracking using sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, vol. 3. Mar. 2003, pp. 1954–1961.

[6] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng, "Efficient in-network moving object tracking in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 8, pp. 1044–1056, Aug. 2006.

[7] A. Prati, R. Vezzani, L. Benini, E. Farella, and P. Zappi, "An integrated multi-modal sensor network for video surveillance," in *Proc. 3rd ACM Int. Workshop Video Surveill. Sensor Netw.*, Nov. 2005, pp. 95–102.

[8] N. Boudriga, "A WSN-based system for country border surveillance and target tracking," *Adv. Remote Sens.*, vol. 5, no. 1, pp. 51–72, 2016.

[9] S. Berrahal, J.-H. Kim, S. Rekhis, N. Boudriga, D. Wilkins, and J. Acevedo, "Border surveillance monitoring using quadcopter UAV-aided wireless sensor networks," *J. Commun. Softw. Syst.*, vol. 12, no. 1, pp. 67–82, Mar. 2016.

[10] F. Talantzis, A. Pnevmatikakis, and L. C. Polymenakos, "Real time audio-visual person tracking," in *Proc. IEEE Workshop Multimedia Signal Process.*, Oct. 2006, pp. 243–247.

[11] X. Zou and B. Bhanu, "Tracking humans using multi-modal fusion," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Sep. 2005, p. 4.

[12] J. Luo, Z. Zhang, C. Liu, and H. Luo, "Reliable and cooperative target tracking based on WSN and WiFi in indoor wireless networks," *IEEE Access*, vol. 6, pp. 24846–24855, 2018.

[13] P. Laube and R. S. Purves, "How fast is a cow? Cross-scale analysis of movement data," *Trans. GIS*, vol. 15, no. 3, pp. 401–418, Jul. 2011.

[14] R. Gallotti, R. Louf, J.-M. Luck, and M. Barthelemy, "Tracking random walks," *J. Roy. Soc. Interface*, vol. 15, no. 139, Feb. 2018, Art. no. 20170776.

[15] V. D. Colceriu, T. Stefanut, V. Bâcu, and D. Gorgan, "Low grade sensor data based annotation of topological railway maps," in *Proc. 21st Int. Conf. Control Syst. Comput. Sci.*, May 2017, pp. 167–174.

[16] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: Enhancing driving directions with taxi drivers' Intelligence," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 220–232, Jan. 2013.

[17] R. Xu, Y. Guan, and Y. Huang, "Multiple human detection and tracking based on head detection for real-time video surveillance," *Multimedia Tools Appl.*, vol. 74, no. 3, pp. 729–742, Feb. 2015.

[18] Q. Huang, "Mining online footprints to predict user's next location," *Int. J. Geograph. Inf. Sci.*, vol. 31, no. 3, pp. 523–541, 2017.

[19] K. Mechitov, S. Sundresh, Y. Kwon, and G. Agha, "Cooperative tracking with binary-detection sensor networks," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst.*, Nov. 2003, pp. 332–333.

[20] B. Song, H. Choi, and H. S. Lee, "Surveillance tracking system using passive infrared motion sensors in wireless sensor network," in *Proc. Int. Conf. Inf. Netw.*, Jan. 2008, pp. 1–5.

[21] S. Xiao, W. Li, H. Jiang, Z. Xu, and Z. Hu, "Trajectroy prediction for target tracking using acoustic and image hybrid wireless multimedia sensors networks," *Multimedia Tools Appl.*, vol. 77, no. 10, pp. 12003–12022, May 2018.

[22] M. Bocca, O. Kaltiokallio, N. Patwari, and S. Venkatasubramanian, "Multiple target tracking with RF sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 8, pp. 1787–1800, Aug. 2014.

[23] W.-P. Chen, J. C. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 3, pp. 258–271, Jul./Aug. 2004.

[24] V. Cevher, A. C. Sankaranarayanan, J. H. McClellan, and R. Chellappa, "Target tracking using a joint acoustic video system," *IEEE Trans. Multimedia*, vol. 9, no. 4, pp. 715–727, Jun. 2007.

[25] Q. Zhou and J. Aggarwal, "Object tracking in an outdoor environment using fusion of features and cameras," *Image Vis. Comput.*, vol. 24, no. 11, pp. 1244–1255, Nov. 2006.

[26] R. Olfati-Saber and P. Jalalkamali, "Collaborative target tracking using distributed Kalman filtering on mobile sensor networks," in *Proc. Amer. Control Conf.*, Jun./Jul. 2011, pp. 1100–1105.

[27] X. Wang, M. Fu, and H. Zhang, "Target tracking in wireless sensor networks based on the combination of KF and MLE using distance measurements," *IEEE Trans. Mobile Comput.*, vol. 11, no. 4, pp. 567–576, Apr. 2012.

[28] J. Kang, I. Cohen, and G Medioni, "Tracking people in crowded scenes across multiple cameras," in *Proc. Asian Conf. Comput. Vis.*, vol. 7, Jan. 2004, pp. 1–6.

[29] N. Ahmed, M. Rutten, T. Bessell, S. S. Kanhere, N. Gordon, and S. Jha, "Detection and tracking using particle-filter-based wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 9, pp. 1332–1345, Sep. 2010.

[30] S. Oh, "A scalable multi-target tracking algorithm for wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 8, no. 9, Sep. 2012, Art. no. 938521.

[31] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1806–1819, Sep. 2011.

[32] S. Oh, S. Sastry, and L. Schenato, "A hierarchical multiple-target tracking algorithm for sensor networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 2005, pp. 2197–2202.

[33] Y. A. U. Rehman, M. Tariq, and T. Sato, "A novel energy efficient object detection and image transmission approach for wireless multimedia sensor networks," *IEEE Sensors J.*, vol. 16, no. 15, pp. 5942–5949, Aug. 2016.

[34] X. Wang, J. Ma, S. Wang, and D. Bi, "Distributed energy optimization for target tracking in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 1, pp. 73–86, Jan. 2010.

[35] N. Anjum and A. Cavallaro, "Trajectory association and fusion across partially overlapping cameras," in *Proc. 6th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Sep. 2009, pp. 201–206.

[36] D. Cheng, Y. Gong, J. Wang, Q. Hou, and N. Zheng, "Part-aware trajectories association across non-overlapping uncalibrated cameras," *Neurocomputing*, vol. 230, pp. 30–39, Mar. 2017.

[37] M. Fayyaz, "Classification of object tracking techniques in wireless sensor networks," *Wireless Sensor Netw.*, vol. 3, no. 4, pp. 121–124, Apr. 2011.

[38] J. D. Mazimpaka and S. Timpf, "Trajectory data mining: A review of methods and applications," *J. Spatial Inf. Sci.*, no. 13, pp. 61–99, 2016.

[39] A. Valsamis, K. Tserpes, D. Zissis, D. Anagnostopoulos, and T. Varvarigou, "Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction," *J. Syst. Softw.*, vol. 127, pp. 249–257, May 2017.

[40] C. Küçükkeçeci and A. Yazici, "A graph-based big data model for wireless multimedia sensor networks," in *Proc. INNS Conf. Big Data*, Thessaloniki, Greece, Oct. 2016, pp. 205–215.

[41] Y. Zheng, X. Xie, and W.-Y. Ma, "GeoLife: A collaborative social networking service among user, location and trajectory," *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, Jun. 2010.

[42] E. Masazade, R. Niu, and P. K. Varshney, "Dynamic bit allocation for object tracking in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5048–5063, Oct. 2012.

[43] O. Ozdemir, R. Niu, and P. K. Varshney, "Dynamic bit allocation for target tracking in sensor networks with quantized measurements," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2010, pp. 2906–2909.

[44] H. Oztarak, K. Akkaya, and A. Yazici, "Lightweight object localization with a single camera in wireless multimedia sensor networks," in *Proc. IEEE Global Telecommun. Conf.*, Nov./Dec. 2009, pp. 1–6.

[45] S. B. Nadler, Jr., *Hyperspaces of Sets* (Monographs and Textbooks in Pure and Applied Mathematics), vol. 49. New York, NY, USA: Marcel Dekker Inc., 1978.

[46] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2000, pp. 285–289.

[47] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Sep. 1999, pp. 1150–1157. doi: 10.1109/ICCV.1999.790410.

[48] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 850–863, Sep. 1993.

[49] S. Vasuhi and V. Vaidehi, "Target tracking using interactive multiple model for wireless sensor network," *Inf. Fusion*, vol. 27, pp. 41–53, Jan. 2016.

[50] K. Hirpara and K. Rana, "Energy-efficient constant gain Kalman filter based tracking in wireless sensor network," *Wireless Commun. Mobile Comput.*, vol. 2017, Apr. 2017, Art. no. 1390847.

[51] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," Mar. 2016, *arXiv:1603.00831*. [Online]. Available: https://arxiv.org/abs/1603.00831
[52] M. A. Brovelli, M. Minghini, M. Molinari, and P. Mooney, "Towards an automated comparison of openstreetmap with authoritative road datasets," *Trans. GIS*, vol. 21, no. 2, pp. 191–206, Apr. 2017.
[53] C. KüÇükkeÇeci and A. Yazici, "Big data model simulation on a graph database for surveillance in wireless multimedia sensor networks," *Big Data Res.*, vol. 11, pp. 33–43, Mar. 2018.
[54] T. Zhang, S. Liu, C. Xu, B. Liu, and M.-H. Yang, "Correlation particle filter for visual tracking," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2676–2687, Jun. 2018.
[55] P. Prasad and A. Gupta, "Moving object tracking and detection based on Kalman filter and saliency mapping," in *Data Engineering and Intelligent Computing* (Advances in Intelligent Systems and Computing), vol. 542. Singapore: Springer, 2018, pp. 639–646.
[56] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[57] R. Kohavi, "The power of decision tables," in *Proc. 8th Eur. Conf. Mach. Learn.* Berlin, Germany: Springer-Verlag, Apr. 1995, pp. 174–189. doi: 10.1007/3-540-59286-5_57.
[58] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Proc. Adv. Neural Inf. Process. Syst.*, 1998, pp. 507–513.
[59] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. 11th Conf. Uncertainty Artif. Intell.* San Mateo, CA, USA: Morgan Kaufmann, Aug. 1995, pp. 338–345.
[60] S. Le Cessie and J. C. van Houwelingen, "Ridge estimators in logistic regression," *Appl. Statist.*, vol. 41, no. 1, pp. 191–201, 1992.
[61] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 683–697, Sep. 1992.
[62] P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *Proc. IEEE Intell. Vehicles Symp.*, Gold Coast, QLD, Australia, Jun. 2013, pp. 797–802.
[63] B. T. Morris and M. M. Trivedi, "Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis," in *Proc. IEEE 5th Int. Conf. Adv. Video Signal Based Surveill.*, Sep. 2008, pp. 154–161.
[64] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Comput. Sci.*, vol. 60, pp. 708–713, Jan. 2015.

**CIHAN KÜÇÜKKEÇECI** received the B.S. degree in computer engineering from Hacettepe University, Ankara, Turkey, in 2005, and the M.S. degree in computer engineering from Bilkent University, Ankara, in 2007. He is currently pursuing the Ph.D. degree in computer engineering with Middle East Technical University, Ankara. He is currently with Luciad-Hexagon Geospatial, Leuven, Belgium, as a Senior Software Engineer. His research interests include spatio-temporal data mining, big data analytics, the Internet of Things, and graph databases.

**ADNAN YAZICI** received the Ph.D. degree in computer science from the EECS Department, Tulane University, USA, in 1991. He is currently with the Computer Science Department, Nazarbayev University, Astana, Kazakhstan. He has published more than 200 international technical articles and coauthored three books. His current research interests include data science, multimedia databases and information retrieval, wireless multimedia sensor networks, and fuzzy data modeling. He received the IBM Faculty Award, in 2011, and the Parlar Foundation Young Investigator Award, in 2001. He was a co-chair of the 23rd IEEE International Conference on Data Engineering, in 2007, the 38th Very Large Data Base (VLDB), in 2012, and the 23rd IEEE International Conference on Fuzzy Systems, in 2015. He is currently an Associate Editor of the IEEE Transactions on Fuzzy Systems and a member of the ACM, the IEEE Computational Intelligence Society, and the Fuzzy Systems Technical Committee.