# Technical Report: Activity Recognition in Wide Aerial Video Surveillance Using Entity Relationship Models

Jongmoo Choi, Yann Dumortier, Jan Prokaj, and Gérard Medioni
Institute for Robotics and Intelligent Systems
University of Southern California
3737 Watt Way, PHE 101, Los Angeles, CA, 90089
February, 2012
{jongmooc, medioni}@usc.edu

## Abstract

*We present the design and implementation of an activity recognition system for wide area aerial video surveillance using Entity Relationship Models (ERM). In this approach, finding an activity is equivalent to sending a query to the Relational DataBase Management System (RDBMS). By incorporating reference imagery and Geographic Information System (GIS) data, tracked objects can be associated with physical meanings, and several high levels of reasoning, such as traffic patterns or abnormal activity detection, can be performed. We demonstrate that different types of activities, with hierarchical structure, multiple actors, and context information, are effectively and efficiently defined and inferred using the ERM framework. We also show how visual tracks can be better interpreted as activities by using geo information. Experimental results on both real visual tracks and GPS traces validate our approach.*

## 1. Introduction

The ability to automatically or interactively infer meaningful activities and events from large volumes of existing video data should be of considerable help to analysts. The goal of this paper is to provide an efficient activity recognition framework for wide area aerial video surveillance where vehicular segmented tracks are the essential components.

The input to our activity recognition framework is geo-registered tracks inferred by a tracking module. Activities are defined as tracks associated with certain properties and their relationships with one or more objects (be it tracks, or other georeferenced entities). Since an activity may involve a sequence of motion patterns (events) and multiple actors, how to represent events and activities is a challenging task.

We propose to define and recognize a large number of activities with the Entity Relationship Model (ERM) framework [7]. The ERM is an appropriate framework to capture multiple relationships between elements, which allows us to efficiently represent hierarchical structures, multiple actor activities, and context information. We use a RDBMS (Relational DataBase Management System), such as Microsoft SQL [2], to store and retrieve all meta-data in our activity recognition system, including tracking results, geospatial objects and context information, and use Structured Query Language (SQL) [2] to define and recognize activities. In this approach, finding an activity is equivalent to sending a set of SQL statements to the RDBMS. As an additional benefit, RDBMS scales well to a distributed system to handle large amounts of data.

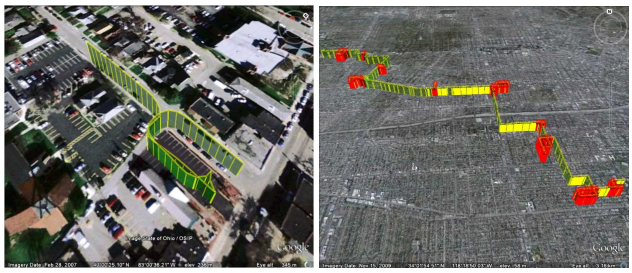

Figure 1. An example of identified "loop" from wide area imagery [4] using our method. Rendered closed-up view, using Google Earth [1], shows that the location is a parking lot (left). A GPS track (yellow color) and a set of identified "loops" (red color) (right).

Many activities can only be inferred within the context of geospatial information and the ERM framework is ideally suited to incorporate Geographic Information System (GIS) data. We use open geospatial data from Open Street Map [3] to extract geospatial objects, such as road networks and road types. To represent geospatial activities, we link visual tracks and geospatial objects.

The contribution is summarized as follows:

- We propose to use the entity relationship model (ERM), a well-established methodology in real world applications, to design and implement an activity recognition system for wide aerial video surveillance where vehicular segmented tracks are the essential components. By leveraging the powerful computational features of a RDBMS, which is an implementation of ERM, finding an activity is equivalent to sending a query to the RDBMS.
- We validate our approach on noisy visual tracks estimated from real data using a state of the art tracker [21] for wide area aerial imagery.
- We demonstrate that different types of activities, with hierarchical structure, multiple actors, and (geo) context information, can be effectively defined and inferred using the ERM framework.
- We also show that visual tracks can be interpreted as activities using geo information. By incorporating reference imagery and extensive use of GIS context, tracked objects can be associated with physical meanings, and several high levels of reasoning, such as traffic patterns or abnormal activity detection, can be performed.

The derived information can be integrated into an existing GIS, providing the capability to analyze and infer higher levels of activities. Furthermore, existing interactive visualization tools for geospatial data, such as Google Earth [1], can be immediately taken advantage of (Fig. 1).

## 2. Related Work

We now present an overview of methods to represent activities. A recent survey [23] describes actions, simple motion patterns usually performed by a single actor, represented by non-parametric (e.g. template matching and dimensional reduction), volumetric (e.g. space-time filtering and tensors), and parametric (e.g. HMMs and linear dynamic systems) methods. Activities, which are complex sequences of actions, are represented using graphical models (e.g. Dynamic Bayesian Nets and Petri nets [5]), Syntactic (e.g. Context Free Grammars and Attribute Grammars), and Knowledge Based (Constraint Satisfaction, Logic Rules, and Ontologies) approaches.

Classical pattern classification formulations, employing a fixed dimensional input vector, cannot handle complex activities which need to consider temporal relationships and context information. In [13], Gaur *et al*. propose a "sting of feature graphs" model to represent local collections of features, which are matched using graph-based spectral techniques and dynamic programming. It allows variability in sampling rates and speed of activity execution. HMM-based approaches are widely applied to speech recognition to recognize a sequence of features. However, the assumption of Markovian dynamics and the time-invariant nature of the model restrict the method to simple stationary temporal patterns [23].

Graphical models, including Bayesian networks (BN) and Dynamic belief networks (DBN), have been widely applied for higher level representation and reasoning since Bayesian networks present conditional dependencies between random variables [14]. For a large scale, real world problem, defining and learning the structure of the DBNs is difficult because of the large number of variables with complex dependencies, and the closed world assumption.

AND-OR graph, an equivalent representation to context-free grammars, is applied to image interpretation [16]. Because deterministic grammars (e.g. FSA, CFG) cannot code low-level uncertainties, Stochastic Context Free Grammars (SCFG) are presented for complex activities [17]. The limitation of SCFG is that complex temporal constraints, such as parallelism, overlap, and synchrony, cannot be expressed. For example, the rule $A \rightarrow abB$ does not specify whether the events can overlap or not, as mentioned in [9]. Damen and Hogg propose attribute grammars [10] to constrain the spatial relationships in visual scenes although the method is able to handle a single event.

Logic-based approaches rely on formal logic rules to describe activities related with common sense knowledge [19]. "Video Event Recognition Language" (VERL) is proposed based on an ontology and first-order logic [11]. Fung *et al*. [12] presented a combination of predicate logic and probability for information fusion and decision support.

We now briefly review the literature on activity in wide area surveillance. Reilly *et al*. [22] shows object detection and tracking in a wide area surveillance domain, where bipartite graph matching and linking tracks were applied to detection results, and grid cells were employed to provide a set of local scene constraints such as road orientation and object context for tracking. Pollard *et al*. [20] presented activity detection results using a complex probabilistic framework but only single activity, convoys, was presented and geospatial constraints were not considered. In [15], high-level complex event inference from multimodal data using Markov Logic Networks is presented for wide area surveillance.

Given our domain, where vehicular segmented tracklets are the essential components, we believe that activities can be effectively and efficiently inferred using a relational database model.

## 3. ERM-based Activity Recognition

The input to our system is a set of tracks $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_N\}$, where $N$ is the total number of

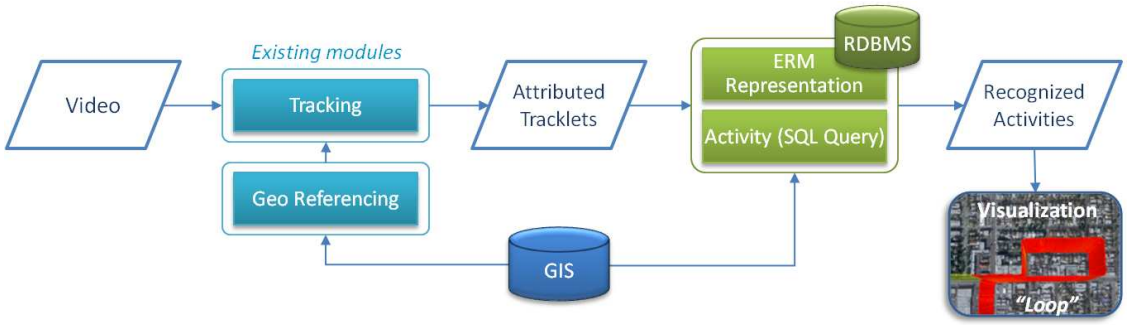


Figure 2. Overview of the proposed approach.

tracks. Each track contains $M$ tracked points $\mathbf{o} = (p_1, p_2, \cdots, p_M)$. In activity recognition, multiple tracks can be associated with a single activity and a single track can contribute to multiple activities. Hence, it is a many-to-many mapping from a set of input data to a set of class symbols. The activity recognition suggested in this paper is defined to automatically find a subset of the input data $(\mathbf{O})$, which matches an activity[1] $\mathbf{a}_j$, $(\mathbf{O}, \mathbf{a}_j) \rightarrow \{\mathbf{o}'_1, \mathbf{o}'_2, \cdots, \mathbf{o}'_N\}$ where $\mathbf{o}'_i$ is a subset of the track $\mathbf{o}_i$. Figure 2 shows an overview of our approach.

### 3.1. Estimating tracks

Recognizing meaningful activities from vehicle tracks requires that the estimated tracks be useful – long enough to capture interesting motion patterns. Any tracking algorithm that is robust enough to provide such tracks can be used in our work. In our implementation, we use a state of the art real-time tracker for wide area imagery introduced in [21], which has been used by Lawrence Livermore National Lab on real data.

### 3.2. Tracklets from tracks

The atomic spatio-temporal information in our system is a tracklet, a segmented portion of a track, representing vehicle's "instantaneous" motion. Each tracklet has a collection of attributes $x_i = \{\lambda_1, \lambda_2, \cdots, \lambda_m\}$, where an element $\lambda_i$ presents a physical property such as time, location, and speed.

Tracks, as computed from the tracking module or captured using GPS, are represented as a set of points that are uniformly sampled in time, but not uniformly sampled in space. In other words, the distance between two adjacent point samples varies, depending on the target's speed. This distance may be long enough to prevent accurate geospatial activity inference in locations between the point samples. Therefore, we cannot generate tracklets by directly enumerating the segments between raw point samples. The trajectories need to be resampled, but we do not want a very high density sampling, as that increases storage and computational requirements.

Clearly, the most important points of the track are those where the direction of travel changes. Between these points, the motion pattern is constant and predictable. Therefore, we segment the track's trajectory into segments which are accurately approximated by lines (linear model). Points on the trajectory between these segments are where the direction of travel changes, often as a result of the vehicle making a turn. Before segmentation, we filter out noise in the trajectories output from the tracker by minimizing a robust cost function (Huber) over a sliding window of 5 locations.

We determine the trajectory segmentation optimally, using a classic dynamic programming algorithm "segmented least squares" [6]. Before the segmentation, the trajectories are densely resampled so that the distance between adjacent sample points is constant ($\sim 4$ meters).

Tracklets are determined from the resulting segmentation by creating one tracklet for each segment ("straight" tracklet), as well as one for the path between every two adjacent segments ("turn" tracklet). Furthermore, straight tracklets longer than 100 meters are broken into shorter 50 meter segments. Now, for each tracklet, we compute a collection of attributes, such as location, heading (applies to straight tracklets), heading change (applies to turn tracklets), speed, acceleration, and accumulated distance traveled so far.

### 3.3. Activity Representation Using ERM

We use ERM (Entity Relationship Models) to capture multiple relationships between elements. Such a framework has been extensively used and validated for a long period of time in real world applications [8, 7]. The basic entity-relationship modeling approach is based on describing data in terms of the three parts: entities, relationships between entities, and attributes of entities or relationships. The relationships include "Belonging to", "Set and subset relationships", "Parent-child relationships", and "Component parts

[1] We do not distinguish between simple actions (or events) and activities since our ERM framework can represent both consistently

Table 1. ERM representation

| | |
|---|---|
| **Entity** | |
| | track point, tracklet, track, traffic rule, |
| | road, building, area, $\cdots$ |
| **Relationship** | |
| | tracklet -is on- road |
| | road -has- traffic rule |
| | must_stop -is a- traffic rule, $\cdots$ |
| **Event** can be represented by a relationship | |
| | tracklet (track_id, $\cdots$, speed=95) |
| | tracklet (track_id, $\cdots$, road_id) |
| | road (road_id, $\cdots$, speed_limit) |
| | speeding: tracklet.speed $>$ road.speed_limit |

of an object".

Hence, we represent track points $\{p\}$, tracklets $\{x\}$, and tracks $\{\mathbf{o}\}$ as entities and link the three entities: $\{p\} \subset \{x\} \subset \{\mathbf{o}\}$. The collection of physical properties of each tracklet is represented as the attributes of the tracklet entity (a RDBMS table).

We also represent geospatial data (traffic rules, roads, buildings, and areas) the same way. An entity "road" is a collection of road segments and each segment has a set of attributes such as type, name, and speed-limit. Table 1 illustrates our ERM representation.

An activity $a_j$ is defined as a collection of tracklets obeying certain properties: $a_j = \{x | x \in \Omega_j, C_j(x) > \theta_j\}$, where $\Omega_j$, $C_j(x) \in [0, 1]$, and $\theta_j$ represent the relationship associated with the activity, the confidence function and the recognition threshold, respectively. The relationship $\Omega_j$ links between the attributes of entities, which include both the physical properties of tracklets and the geospatial data. We can define relationships that are not explicitly represented in the ERM.

For example, *"Speeding"* can be seen as an activity defined by the relationship between the attributes of tracklets (e.g. speed) and geospatial objects (e.g. speed-limit):

$$speeding := \{x | \quad r \in \mathcal{G}_{road}, x.roadID = r.ID, x.s > r.s, \\ C(x.s, r.s) > \theta\} \tag{1}$$

where $r, r.ID, x.roadID, x.s, r.s$ represent a road from GIS data ($\mathcal{G}_{road}$), its ID, the road ID of tracklet $x$, the speed of $x$, and the speed limit of $r$, respectively. $C(x.s, r.s)$ describes the activity confidence, which increases with the gap between $x.s$ and $r.s$. The confidence measure is used to ensure the reliability of composite activities as well as offering users a way to tune the system.

An ERM has these desirable properties:

- **Hierarchical structure**: a complex activity composed of many sub-actions can be represented by a "component parts of an object" relationship (Section. 3.4.2).
- **Multiple actors**: the mathematical foundation of ERM is set theory, so that an activity involving multiple actors can be defined naturally (Section. 3.4.3).
- **Context information**: low level observations can be represented by uncertain numerical values while background knowledge often needs to be described by symbolic representations. Observation and prior knowledge can easily be integrated (Section. 3.4.4).
- **Scalability**: queries can be executed efficiently and in parallel (Section. 4.4).

While it is limited by the maturity of the underlying technologies (tracking, *etc*.), ERM offers a graceful degradation in performance. Tracks that are fragmented will not be detected for activities defined over long time periods. These tracks, however, will still be detected for activities happening on shorter time scales. Activity definitions do not have to be modified for future improvements in tracking. These will be immediately reflected in the recognition performance.

ERM, as we have defined it, is a rule-based, rather than a probabilistic framework. However, we believe there is nothing inherent in the framework that prevents the addition of probabilistic reasoning to handle uncertainty [18]. In this paper, our focus is on the fundamental design of a scalable activity recognition framework for wide area aerial imagery.

### 3.4. Activity Inference

The ERM-based representation implies that inferring an activity is a search problem to find a subset of tracklets from entire data set, which satisfies certain conditions.

ERM is implemented as a standard RDBMS and we can express set operations by SQL to find an activity from our database. The activity recognition problem is equivalent to sending queries to the RDBMS.

A basic SQL statement has *SELECT*, *FROM*, and *WHERE* clauses: The *SELECT* command specifies the output attributes of entities, *FROM* defines the domain entities associated with the activity, and *WHERE* describes the set of relationships to define the activity and also its confidence. Activity definitions can easily be expressed by SQL statements.

In the following section we explain how to define different types of activities using ERM and how to infer those activities using SQL commands.

#### 3.4.1 Example I: Simple Activity

Activities associated with motion patterns, such as *"U-turn"*, *"Loop"*, and *"3-point-turn"*, are easily defined and inferred by the ERM framework and its corresponding SQL statements.

**Definition.** A *"Loop"* is defined as a segmented track where there exist two tracklets $\{x_i, x_j\}$ whose Euclidean distance $\|x_i.pos - x_j.pos\|$ is smaller than the traveling distance: $Loop = \{x_i, x_j | (1 - \frac{\|x_i.pos - x_j.pos\|}{x_j.acc - x_i.acc}) > \theta, i < j, x_i.ID = x_j.ID\}$, where $(x_j.acc - x_i.acc)$ represent the traveling distance between $x_i$ and $x_j$. The traveling distance is computed as the difference of the accumulated distances between these two tracklets.

The above definition is represented by SQL as shown in Table 2, where RDBMS tables T1 and T2 come from the input tracklet table (e.g *SELECT* * INTO T1 *FROM* tracklet) and dist$(\cdot, \cdot)$ is a user defined function [2] to compute the Euclidean distance.

Table 2. SQL: *"Loop"*

| |
|---|
| *SELECT* * *FROM* T1, T2 |
| *WHERE* |
| T1.track_id = T2.track_id *AND* |
| (1 - (dist(T1.pos, T2.pos)/(T2.acc - T1.acc))) $> \theta$ |

Figure 1 shows a result of *"Loop"* recognition, where a track contains several loops. Its corresponding SQL definition provides a set of tracklets associated with the activity.

#### 3.4.2 Example II: Composite Activity

Suppose that we have three independent events identified as three entity sets: *"Entry"* $(a_{En})$, *"Stay"* $(a_{St})$ and *"Exit"* $(a_{Ex})$. *"Visit"* is a composite activity that can be described as a combination of these events.

**Definition.** We define *"Visit"* as the sequence of $a_{En}$, $a_{St}$, and $a_{Ex}$: $visit = \{x_j | i = j - 1, k = j + 1, x_i \in a_{En}, x_j \in a_{St}, x_k \in a_{Ex}, C(x_i)_{En} C(x_j)_{St} C(x_k)_{Ex} > \theta\}$, with $x_i, x_j, x_k$, three tracklets from the same track. This definition can be represented by SQL as shown in Table 3. Each identified event is a table, $Activity(ID, Confidence)$, that contains the IDs of tracklets and the confidence values. The confidence of *"Visit"* is defined as $C_{En}(x_i) C_{St}(x_j) C_{Ex}(x_k)$.

#### 3.4.3 Example III: Multiple Actors Activity

Activities associated with multiple actors, such as *"Source"*, *"Sink"*, *"Convoy"*, and *"Following"*, can also be defined and inferred by ERM and SQL statements. We identify a source of tracks by finding a set of tracks that have the same starting location in different time periods.

**Definition.** Let us first define *"2-Source"* as a temporary set of 2 tracklets which exit from the same location: $2src = \{(x_i, x_j) | x_i.trackID \neq x_j.trackID, x_i \in a_{Ex}, x_j \in a_{Ex}, \|x_i.pos - x_j.pos\| < \omega\}$, where $\omega$ is a threshold. It provides a set of tracklet pairs which appear as many times as they are involved in a 2-tuple source. Extracting N-tuple source needs to count the number of occurrence for each tracklet: $source = \{x_i | S_i = \{(x_i, \cdot) \in 2src\}, |S_i| > \theta\}$, where $|S_i|$ is the cardinality of each subset $S_i$ which contains the same tracklets in the pairs of the 2src set. Note that this definition provides all "SOURCE" , where the number of tracklets is greater than two and we can count the number of tracklets as a confidence measure as shown in Table 4, where "Exit" action is represented a table, Ex(id, conf), that contains the id of tracklets.

Table 3. SQL: *"Visit"*

| |
|---|
| *SELECT* * *FROM* T1, T2, T3, En, St, Ex |
| *WHERE* |
| T1.track_id = T2.track_id *AND* |
| T2.track_id = T3.track_id *AND* |
| T1.id + 1 = T2.id *AND* |
| T2.id + 1 = T3.id *AND* |
| T1.id = En.id *AND* |
| T2.id = St.id *AND* |
| T3.id = Ex.id *AND* |
| (En.conf * St.conf * Ex.conf) $> \theta$ |

Table 4. SQL: *"Source"*

| |
|---|
| *SELECT* T1.id as id INTO tmp *FROM* T1, T2, Ex |
| *WHERE* |
| T1.track_id $\neq$ T2.track_id AND |
| T1.id = Ex.id AND |
| T2.id = Ex.id AND |
| dist(T1, T2) $> \theta$; |
| *SELECT* id, count(id) as confidence *FROM* tmp |
| GROUP BY id |

#### 3.4.4 Example IV: Geospatial Activities

The ERM framework is ideally suited to incorporate GIS information, as was shown for *"Speeding"*(Section. 3.3). Many activities can only be inferred within the context of geospatial information. We can find "all tracklets on a specific road" by looking at the correspondences between the locations of tracklets and the locations of known road segments.

**Definition.** *"On-road-X"* is a set of tracklets which are on the same road: $on\_road\_X = \{x | x.roadID = r.ID, r.name = "X", 1/\|x.pos - r.pos\| > \theta \forall r \in \mathcal{G}_{road}\}$, where $r$ and $r.name$ designate a road segment and its name, and $\|x.pos - r.pos\|$ the Euclidian distance between the road segment and the tracklet. This definition is represented by SQL as shown in Table 5, where $T1$ and $Road$ are the tracklet and road segment tables, respectively. Here, we compute the location of each tracklet in advance,

[2] For simplicity, we use an abstract notation and the function can be implemented using a common RDBMS [2].

and store the id of road segment into the tracklet table. The optional condition (1/dist(T1.pos, Road.pos) $> \theta$) provides a confidence measure.

Table 5. SQL: *"On-road-X"*

*SELECT* * *FROM* T1, Road
*WHERE*
T1.id = Road.id *AND*
Road.name = "X" *AND*
1/dist(T1.pos, Road.pos) $> \theta$

Note that most spatial activities can also be enriched by having a geospatial attribute. For instance, a *"convoy"* becomes a "convoy traveling on highway X" when the spatial tracks are associated with geospatial information.

### 3.5. Scalability

One of the benefits of using an ERM model for activity recognition is that there are highly optimized commercial implementations [2]. Furthermore, there has been a lot of effort in making RDBMS perform equally well in distributed environments, under high load, and with limited downtime. Therefore, by expressing activity definitions in SQL, we can take advantage of existing, distributed, industrial parsers, making our proposed system very scalable.

Some of the ways that RDBMS achieve high efficiency is through the use of indexing and data slicing. We take advantage of both of these strategies. For example, when a query requires joining of several tables on "id" attribute, an index is built on this attribute in any (temporary) tables taking part in the join. Similarly, when recognizing an activity happening over a relatively short time interval, we first slice the database into multiple time intervals (with overlap), and then query each interval in turn for this activity. This is much faster than querying the whole database at once (Sec. 4.4). Of course, all known query optimization strategies can be adopted here.

## 4. Experimental results

We have implemented our framework using a standard RDBMS (MS-SQL [2]), and validated the approach on real visual tracks and GPS datasets. We define 7 activities for evaluation (including *"Loop"*): a three point turn (*"3PT"*) consists of two neighbor turns $\{x_i, x_j | ((x_i.\phi/\pi)(x_j.\phi/\pi))/(\|x_i.pos - x_j.pos\|) > \theta\}$; a two point turn (*"2PT"*) has an acute angle: $\{x | x.\phi > \theta\}$; *"Stay"* is defined by the ratio between the time and travel distance between two points $\{x_i | (\|(x_j.time - x_i.time)\|)/(\|x_j.acc - x_j.acc\|) > \theta\}$; *"U-turn"* has an acute angle turn between two tracklets which are located on the same road $\{x_j | (\|x_j.\phi < \pi/4, \|x_i.pos - x_k.pos\| < \omega_1, (x_k.acc - x_i.acc) > \omega_2\}$; *"Entry"* and *"Exit"* is defined with a stop, turns, speed changes and the travel distance. *"Entry"* is defined as $\{x_k | x_k.end = True, x_k.s < x_i.s, x_j.\phi > \omega_1, x_k.acc > \omega_2\}$. $\{x_i, x_j, x_k\}$, $x.\phi$, and $\omega_i$ represent different tracklets from the same track, such as $i < j < k$, the turn angle attribute, and internal thresholds associated with the definition, respectively.

Table 6. Confusion Matrix (Real Data Set)

| Out \ Actual | Loop | 2PT | 3PT | Entry | Exit | None |
|---|---|---|---|---|---|---|
| Loop | 2 | 0 | 0 | 2 | 0 | 0 |
| 2PT | 0 | 2 | 1 | 1 | 1 | 0 |
| 3PT | 0 | 0 | 2 | 0 | 0 | 0 |
| Entry | 1 | 0 | 0 | 7 | 1 | 0 |
| Exit | 0 | 0 | 0 | 1 | 6 | 0 |
| None | 0 | 0 | 0 | 0 | 1 | - |

### 4.1. Real dataset (CLIF 2006)

**Data.** The dataset is a set of tracking results extracted from the CLIF 2006 dataset [4]. This dataset contains wide area motion imagery captured from an airborne sensor. The sensor is composed of a matrix of 6 cameras, where the size of each image tile is $4008 \times 2672$. The video is captured at roughly 2Hz, and it is in grayscale. The sequence is an example of persistent surveillance imagery, where the airplane makes several circular flyovers around the campus of Ohio State University. The footprint of the area where we computed tracks is about $1km^2$, and its duration is about 8 minutes. Each track is on average 1 minute long. The total number of tracks estimated in the sequence of interest is more than 8000.

The main challenge is the sheer number of objects, or points of interest, one must consider to determine whether an activity is happening. Furthermore, an activity is not a static concept that one can identify at a glance. Instead, one must verify that a whole sequence of actions is happening to label something a particular activity.

**Method.** Our input is a set of tracks extracted by the tracking module described in Section. 3.2. To build a set of ground truth data, from a set of automatically extracted tracks, we manually selected individual tracks that include pre-defined activities and assign labels for each data. In our dataset, we used 2 loops, two 2 point turns, three 3 point turns, 8 entry and 7 exits. We inserted all selected tracks into a single table in our RDBMS and infer activities using pre-defined SQL statements.

Some tracks have more than one activity (e.g. a loop and a 3 point turn) but the locations associated with specific activities can be different. To evaluate the result of an activity, we extracted all tracklets, compared to the activity definition, from entire dataset, visualize the result using Google Earth, and then, verify manually whether the extracted tracklets represent the actual activity or not.

Table 7. Confusion Matrix (GPS Data Set I)

| Out ╲ Actual | Loop | 3PT | U-turn | Stay | None |
|---|---|---|---|---|---|
| Loop | 17 | 0 | 0 | 0 | 0 |
| 3PT | 0 | 7 | 0 | 0 | 0 |
| U-turn | 0 | 0 | 13 | 0 | 1 |
| Stay | 0 | 0 | 0 | 3 | 0 |
| None | 0 | 2 | 2 | 0 | - |

**Results.** Table 6 shows the confusion matrix among 5 activities, where "None" is a NULL activity to count missing and false alarms. Result shows that we can identify all simple activities, such as "2 point turn", "3 point turn", "Entry", "Exit", and "loop", which can be easily seen in real data set.

In addition, we identified a number of geospatial activities, such as "on road X", "speeding", and "approaching X", as well as some complex activities including multiple actors, such as "sink (or sink) around X". The extracted activities and geospatial objects can be visualized using Google Earth, where we can identify both activities and associated geospatial objects.

### 4.2. GPS trajectory dataset I

**Data.** We also evaluated our method on labeled data from GPS acquisition. We used a standard GPS to record short trips between 10 and 40 minutes long. GPS filters were deactivated, so only raw data have been recorded. Compared to the results we manually labeled from our tracking module, GPS tracks do not differ a lot. First the localization error is mostly the same in both systems, with a 5 meters accuracy for the video geo-registration against 1 to 10 meters at 95% for GPS data. Second, the GPS acquisition frequency (1Hz) is only half our video framerate (2Hz).

**Method.** We use the same tracking module to extract tracklets from the GPS dataset. To build a set of ground truth data, from a set of automatically extracted tracks, we manually selected individual tracks that include pre-defined activities and assign labels for each data. The dataset includes 17 loops, 7 three point turns, and 13 u-turns. Since GPS tracks are much longer than the real tracks and each track includes many activities, we inserted a single track into a single table in our RDBMS and inferred activities using pre-defined SQL statements. To evaluate the result of an activity, we extracted a set of tracklets, corresponding to the activity definition, from a single track, visualize the result using Google Earth, and then, verify manually both missing and false alarms.

**Results.** Table 7 shows the confusion matrix among 4 activities. Result shows that we can identify all simple activities, such as "Loop", "3 point turn", "U-turn", and "Stay", which can easily seen in real data set.

Table 8. Recognition performance on GPS trajectory dataset II.

| Activity | Precision | Recall |
|---|---|---|
| Brushpass | 0.44 | 0.65 |
| Coordinated Movement | 0.64 | 0.38 |
| Dead Drop | 0.16 | 0.54 |

### 4.3. GPS trajectory dataset II

**Data.** We were able to acquire another proprietary GPS trajectory dataset. This dataset was collected over 7 hours, and contains about 50 tracks. These tracks were generated by directing several groups of vehicles to execute various types of activities over the data collection period. These activities ranged in complexity from single-actor, composite, to multiple-actor activities. Each activity was supposed to be executed in a 15-minute interval, but the location and time were otherwise unconstrained. Therefore, the ground truth that is available for this dataset (directions for drivers) is only approximate.

**Method.** Tracklets were estimated from GPS trajectories and stored in a database. To evaluate the recognition performance, we executed a query for each activity, and measured (automatically) the precision and recall using the supplied ground truth.

**Results.** The results are shown in Table 8. The best performance is obtained for brushpass and coordinated movement activities. After analyzing the retrieved tracks in detail, we observed that the ground truth does not always match perfectly to the actual executed activities. However, we did not alter the supplied ground truth in any way. Furthermore, activities labeled as coordinated arrival were sometimes "mistakenly" detected as coordinated movement. This indicates the need for an ontology in evaluating activity recognition performance. Given these circumstances, the recognition performance is encouraging.

### 4.4. Scalability

To verify the scalability of our proposed system, and examine its behavior with respect to data slicing, we used GPS trajectory dataset 2. We used a "brushpass" activity for the purposes of evaluation. This activity requires the join of two tables, twice, so it is a good representative for evaluation.

The scalability of our system can be evaluated by measuring query completion time with respect to the number of tracklet rows. We varied the number of rows by temporally dividing the dataset into a various number of intervals (28,14,7,3,1). Query completion time was then measured in each such time interval. This experiment was performed twice, each time with a different group of vehicles. The results are shown in Figure 3. It shows that query time increases quadratically with the number of rows in the worst case, but the growth is linear for smaller database sizes. This kind of low-order polynomial growth allows the ERM

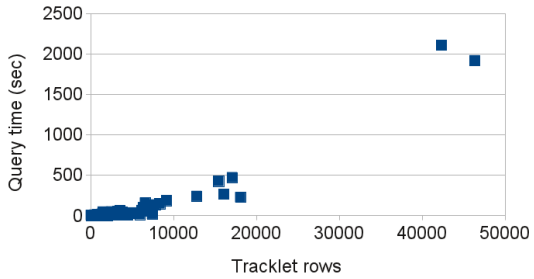


Figure 3. Query completion time rises slowly with the size of the dataset.

Table 9. Data slicing reduces the query completion time.

| Intervals | Avg. Query Time (sec) |
|---|---|
| 28 | 233.25 |
| 14 | 279.95 |
| 7 | 499.02 |
| 3 | 897.65 |
| 1 | 2010.95 |

framework to scale to large database sizes which appear in practice.

To understand the effects of data slicing on query completion time, we use the same data as in the previous experiment. We measure query completion time as the sum of all interval query completion times. These are averaged for the two groups of vehicles. The results are shown in Table 9. It is clear that data slicing reduces the total query completion time. Therefore, when recognizing activities using ERM, it is more efficient to temporally divide the data into intervals (even with overlap) and run the query within each interval than to run the query on the entire database at once.

## 5. Conclusion

Our results show that using Entity Relationship Models we can identify simple activities, such as "U-turn", "2 point turn", "3 point turn", "Entry-Exit", "loop", and "speeding", as well as some complex activities including multiple actors, such as "source" and "sink" in wide area aerial imagery. Extracted activities are visualized using widely available software for viewing geospatial data, such as Google Earth, where we can identify associated geospatial objects.

## References


[1] http://earth.google.com. 1, 2

[2] http://www.microsoft.com/sqlserver/. 1, 5, 6

[3] http://www.openstreetmap.org. 1

[4] http://www.sdms.afrl.af.mil/index.php?collection=clif2006. 1, 6

[5] Petri nets for modeling of dynamic systems: A survey. *Automatica*, 30(2):175 – 202, 1994. 2

[6] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Commun. ACM*, 4(6):284, 1961. 3

[7] P. P. Chen. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976. 1, 3

[8] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970. 3

[9] D. Damen. Activity analysis: Finding explanations for sets of events. *PhD Thesis. University of Leeds*, 2009. 2

[10] D. Damen and D. Hogg. Recognizing linked events: Searching the space of feasible explanations. In *CVPR*, pages 927–934, 2009. 2

[11] A. R. J. François, R. Nevatia, J. R. Hobbs, and R. C. Bolles. Verl: An ontology framework for representing and annotating video events. *IEEE MultiMedia*, 12(4):76–86, 2005. 2

[12] F. Fung, K. Laskey, M. Pool, and M. Takikawa. Plasma: combining predicate logic and probability for information fusion and decision support. paper presented at the aaai spring symposium, 2005. 2

[13] U. Gaur, Y. Zhu, B. Song, and A. Roy-Chowdhury. A "string of feature graphs" model for recognition of complex activities in natural videos. In *ICCV '11*, 2011. 2

[14] A. Gupta and L. Davis. Objects in action: An approach for combining action understanding and object perception. In *CVPR '07*, pages 1 –8, june 2007. 2

[15] H. Gupta, L. Yu, A. Hakeem, T. E. Choe, N. Haering, and M. Locasto. Multimodal complex event detection framework for wide area surveillance. In *Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Conference on*, pages 47 –54, june 2011. 2

[16] R. Hamid, S. Maddi, A. Bobick, and M. Essa. Structure from statistics - unsupervised activity analysis using suffix trees. In *ICCV*, pages 1 –8, 2007. 2

[17] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE TPAMI*, 22(8):852–872, 2000. 2

[18] B. T. Lise Getoor. *Introduction to statistical relational learning*. MIT Press, 2007. 4

[19] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE TPAMI*, 23(8):873 –889, Aug. 2001. 2

[20] E. Pollard, B. Pannetier, and M. Rombaut. Convoy detection processing by using the hybrid algorithm (gmcphd/vs-immc-mht) and dynamic bayesian networks. In *Information Fusion, 2009. FUSION '09. 12th International Conference on*, pages 907–914, 2009. 2

[21] J. Prokaj, M. Duchaineau, and G. Medioni. Inferring tracklets for multi-object tracking. In *Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Conference on*, pages 37 –44, june 2011. 2, 3

[22] V. Reilly, H. Idrees, and M. Shah. Detection and tracking of large number of targets in wide area surveillance. In *ECCV (3)*, pages 186–199, 2010. 2

[23] P. Turaga, R. Chellappa, V. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1473 –1488, 2008. 2