Universities of Leeds, Sheffield and York
http://eprints.whiterose.ac.uk/

Event Model Learning from Complex Videos using ILP

Krishna S. R. Dubba and **Anthony G. Cohn** and **David C. Hogg** [1]

Abstract. Learning event models from videos has applications ranging from abnormal event detection to content based video retrieval. Relational learning techniques such as Inductive Logic Programming (ILP) hold promise for building such models, but have not been successfully applied to the very large datasets which result from video data. In this paper we present a novel supervised learning framework to learn event models from large video datasets($\sim$ 2.5 million frames) using ILP. Efficiency is achieved via the *learning from interpretations* setting and using a typing system. This allows learning to take place in a reasonable time frame with reduced false positives. The experimental results on video data from an airport apron where events such as Loading, Unloading, Jet-Bridge Parking etc are learned suggests that the techniques are suitable to real world scenarios.

1 Introduction

The task of learning event models can be formulated as the representation of time series data from tracking objects in videos and mining this in a supervised or unsupervised fashion for patterns obeying certain constraints. These patterns can be used for analysing the videos, content retrieval, event recognition and anomaly detection. The principal assumption that we make in this work is that it is the collective behaviour and interaction of objects rather than the individual behaviour that make an event. This is the main reason why we pay more attention in our work to analysing the interactions of the objects using spatio-temporal primitives. Because of the relational nature of such data, ILP is an apt choice for learning patterns from this data.

We selected events involved in an airport apron because these events are semantically rich and involve interactions of objects. The videos are collected from an airport apron area using 8 static cameras (Fig 1) that have different views at the same scene. This setting is used to make sure that no information is lost because of occlusions and view points and is more robust than using a video from single view point. Tracking is done separately on all the 8 video streams and fused to get global 3D data for objects moving on a scene-based ground plane [7].

In our framework, the tracking data is converted to relational data where the relations are spatio-temporal relations among objects. We give minimal supervision which is in the form of approximate intervals and regions (when and where) related to the event. We term this *Deictic Supervision*. From this information, the necessary relational facts are extracted from the relational data and used for learning event models. Each positive example obtained from supervision thus naturally consists of a set of spatio-temporal facts. In order to learn from supervision of this form it is natural to use the *learning from interpretations* setting [5] since each such set of facts can be viewed as an interpretation. An advantage which accrues from the use of this setting is more efficient testing of hypotheses as these can be evaluated locally in each interpretation rather than the global database as in the standard ILP setting [3].

Learning event models from this kind of spatio-temporal data is particularly challenging because of the presence of multiple objects, uncertainty from the tracking and especially the time component as this increases the size of the relational data (the number of temporal relational facts is quadratically proportional to the number of intervals present). Since the data is huge and also noisy because of ambiguity and errors in object detection and tracking, a hypothesis can trigger many false positives. The search space is also huge because of the scale of the data. We tackle these problems by careful use of constraints and systematically using the *type hierarchy* of the objects involved in the events. The main novelty of the paper is the presentation of a deictically supervised framework to learn clausal event models from large video data sets using the *learning from interpretations* setting and using techniques from many sorted logic for representing and using a tree structured type hierarchy in getting efficient models that have better performance than when learned using classical ILP techniques. A proper type refinement operator is explored that is very generic and can be applied in any domain as long as object types have a hierarchy.

The paper is organised as follows. First we discuss the related work and formally introduce conceptual apparatus and notation that we use throughout the paper. In Section 3, we formalize the problem and in Section 4 we introduce the type refinement operator. In Section 5 we explain the experiments for learning event models in the airport apron domain and discuss and evaluate our results. We conclude and mention a few possible directions for future work in Section 6.

2 Related Work

Much of the work in event analysis, for example [1, 7, 9], for video surveillance does not involve learning the models. Event models are hand coded using different representations and are used for recognizing the events in videos. These models are biased in the sense that they are not necessarily optimally matched to target instances, and they are not adaptive to changes in the manifestation of event classes. There are several lines of work in learning event models automatically from videos depending on the techniques and representations used for learning event models. Representation of events forms the crux of the problem as it constrains the learning methodologies that can be used.

That event models and protocols can be discovered from a spatio-temporal database was demonstrated by Fernyhough et al. [6] but was constrained to handle only interactions between two objects. Moyle et al. [12] demonstrated using a simple blocks world that Domain Specific Axioms can be learned from temporal observations using an ILP framework.

In work by Needham et al. [14], the Progol system [13] was used to learn the protocols of table top games from real sensory data from

---

a video camera and microphone. ILP was used in an unsupervised setting where only positive examples were used to learn rules governing the games. A key aspect of this work are methods for spatio-temporal attention applied to the sensor data from audio and video devices. These identify subsets of the sensor data relating to discrete concepts. Symbolic description of the continuous data is obtained by clustering within continuous feature spaces from processed sensor data. The Progol ILP system is subsequently used to learn symbolic models of the temporal protocols presented in the presence of noise and over-representation in the symbolic data input to it.

Konik et al. [10] applied ILP techniques for relational learning from observations. The authors used artificially created examples and behaviour observation traces generated by AI agents. The behaviour observations were stored in an episodic database in the form of first-order logic sentences and rules and then ILP techniques were applied to learn planning rules like *go-to-door* etc. Graph based representation has also been used for relational learning especially in event learning. Recently, Sridhar et al. [17] used graphs for representing the time series data for unsupervised learning of event classes from video.

Some work has been done in using types in ILP [15]. Though a proper refinement operator was not defined, the technique relied upon replacing a generic type of an object in the hypothesis by the *lgg* (least general generalization) of the corresponding variable bindings from the Prolog query of the hypothesis in database. This operator considers only the positive examples and ignores the negative examples, so is suitable only in *positive only* mode and also expects the data to be clean (without noise), whereas we define an upward refinement operator for Progol like ILP systems that also considers negative examples and is robust to noise in type information.

There has been some work in modelling and recognition of events from an airport apron [8, 19]. In [19] the authors used an unsupervised learning approach and trajectories as primitives. Their focus is on anomaly detection. Since interactions among objects are ignored, semantically meaningful models are not obtained. In [8], the authors attempted to model a Dynamic Bayesian Network for jointly solving event recognition and broken tracks linking problems. The event model is a set of discrete states which expresses how the actors in the event interact over time. They assume the states are strictly ordered and this may limit learning some events that involve complex temporal relations like *during*, *overlaps* etc.

The important aspect to note here is that most of the work in this area has been done on either artificial data [12] or very simple real world data [6, 14] that involves few objects and the events are of short duration. In our case, the tracked data from videos is huge and at the same time more complex and noisy, with more objects; moreover a semantically rich event can be as long as 4,000 frames.

2.1 Deictic Supervision

The main problem in supervised learning is collecting the training data. Since there is in general no well defined framework for the notion of event, it is difficult to annotate videos with event labels. Also the spatial and temporal components of events have to be identified as part of the event which further complicates the labelling. A solution to this is what we term 'Deictic Supervision'. Instead of giving the exact objects involved in the training examples we only give a bounding spatial and temporal extent of each example event. The spatial extent is an approximate rectangle indicating the area where the event is happening in the video and the temporal extent is an interval and the learning algorithms should be able to induce reasonable models even with this data. This makes preparation of training data easy and the
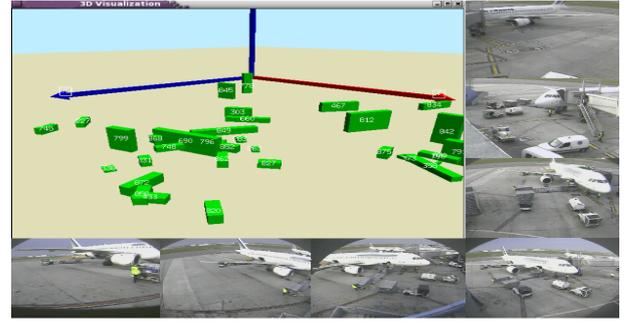
Figure 1: Experimental set-up: Different camera views and the fused 3D view (only 7 of 8 views showed).

learning process more robust and unbiased to the labelling. Delineating spatio-temporal volumes (cuboids) in videos from which to learn feature-based representations of actions such as hand gestures is not without precedent in the computer vision literature [11], but our use here extends it to multiple simultaneous actors and relational descriptions. Note that deictic supervision is different from semi-supervised learning where only part of the training data is labelled, whereas in deictic supervised learning the extent of labelling is reduced to a minimum but all positive training data is labelled. The tracked data is used to compute spatial relations among the objects in the videos. We use the *Surrounds (S), Touches (T) and Disconnected (D)* [17] spatial relations and Allen's temporal relations [2] to represent the relational data (Fig 2).

Def 1. Temporally Extended Spatial Relation: *A temporally extended spatial relation between obj1 and obj2 over an interval [t1,t2] is represented as*

$$rel(obj1, obj2, intv(t1, t2))$$
$$where \ rel \in \{S, T, D\},$$
$$intv(t1, t2) \ is \ the \ time \ interval \ and$$
$$t1, t2 \ are \ the \ bounding \ frame \ numbers.$$
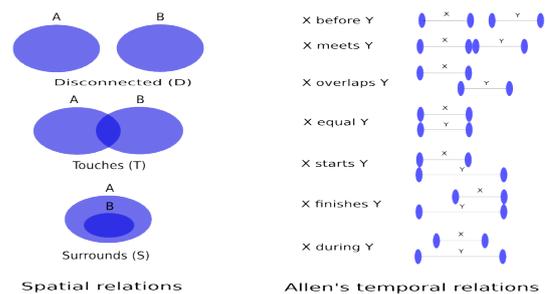
Figure 2: Spatial & Allen's temporal relations

Note that a spatial relation between two solid objects is actually the spatial relation between their bounding boxes on the ground plane obtained from tracking.

For supervised learning, we need positive and preferably negative examples too of event instances. In order to reduce the supervision involved, we only get information of when and where the event hap-

pened, i.e. a deictic interval and a deictic zone. Note that the user does not explicitly give exactly which objects are involved in the event.

Def 2. Deictic interval and region: *Let $T_{(e,v)}$ and $R_{(e,v)}$ be the deictic temporal and spatial terms for an event instance e in video v. $T_{(e,v)}$ is an interval indicating when the event happened and $R_{(e,v)}$ is a polygon on the ground plane indicating where the event happened.*

$R_{(e,v)}$ is obtained by hand-delimiting the event in the image plane with a rectangle and back-projecting this rectangle automatically into a minimum enclosing rectangle on the ground plane using homography. In Fig(3), a video is visualized in 2D space-time as a cuboid. A positive example for an event is another cuboid contained in it. The complement of this inner cuboid can be considered as negative spatio-temporal region for the event. Each inner cuboid corresponds to a Herbrand Interpretation in the *learning from interpretations* setting.

Def 3. Herbrand Interpretation of an Event: *Let $T_{(e,v)}$ and $R_{(e,v)}$ be the deictic spatial and temporal terms for an event e in video v. Let Γ_v be the set of spatial relational facts of v and O_v be the set of objects in v. The set of facts $E_I \subseteq \Gamma_v$ is called the Herbrand Interpretation of the event iff it contains all the facts in Γ_v whose temporal interval is not disjoint with the deictic interval except those relations whose objects are disconnected from the deictic region.*

$$
\begin{aligned}
E_I = \{ &SR(obj1, obj2, intv(t1, t2)) : \\
&\{obj1, obj2\} \in O_v \wedge \\
&SR \in \{S, T, D\} \wedge \\
&TR(T_{e,v}, intv(t1, t2)) \text{ where } TR \notin \{before, after\} \wedge \\
&S1(R_{e,v}, obj1, intv(M, N)) \wedge S2(R_{e,v}, obj2, intv(X, Y)) \wedge \\
&\{S1, S2\} \in \{S, T\} \wedge \\
&TR1(T_{e,v}, intv(M, N)) \wedge TR2(T_{e,v}, intv(X, Y)) \wedge \\
&\{TR1, TR2\} \notin \{before, after\} \\
\}
\end{aligned}
$$

The set of Herbrand Interpretations of an event form the positive examples for the learning phase. The rest of the relations in the videos form the negative region where if an event model fires an instance in this region, it is considered as a false positive.

3 Learning from Interpretations setting

In the *learning from interpretations* setting, every example is a set of facts. The mapping of each example to a class is independent of other examples. To specify formally [3]:
Given:

- A set of classes C (each class label c is a nullary predicate).
- A set of classified examples E (each element of E is of the form (E_i, c) with E_i a set of facts and c a class label)
- and a background theory B,

Find: a hypothesis H (a Prolog program), such that for all $(E_i, c) \in E$:

- $H \wedge E_i \wedge B \models c$, and
- $\forall c' \in C - \{c\} : H \wedge E_i \wedge B \nvDash c'$

The hypothesis is a Prolog rule that has a head and body, where the head's predicate is same as the class label. The search process for hypothesis starts with an initial hypothesis which has a nullary
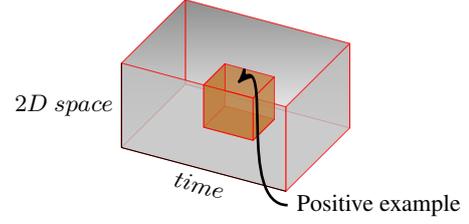
Figure 3: Positive example of an event in a video in 2D space-time dimensions.

predicate as head and an empty body. This initial hypothesis is refined using Progol's refinement operator [13] and also by the type refinement operator which will be explained below. Progol's refinement operator is based on the *most-specific clause* also called *bottom clause*. The *Most-specific clause* can be found from training examples, mode declarations and background knowledge. Mode declarations are user defined syntactic biases in the form of predicates that specify what predicates from the background knowledge are expected in the target hypothesis and also the nature of the variables (input, output or constant). Once this *most-specific clause* is formed, the sub-lattice bounded from below by the *most-specific clause* is searched using an A* like search algorithm and selecting the hypothesis that has a maximum score calculated based on number of positive and negative examples covered, length of hypothesis etc.

Once a satisfactory hypothesis is found, an argument representing temporal information can be introduced into the head in order to explicitly represent *when* the event occurs – this is useful when using the hypothesis for event monitoring – it allows the interval during which the event occurs to be explicitly flagged when viewing the video. An issue to be resolved is exactly when an event occurs given that it consists of multiple overlapping temporal intervals. Given the list of all intervals λ occurring in the body of the hypothesis, various possibilities present themselves. One could take the maximal interval which exactly spans all intervals in λ. Or one could take the interval which exactly spans the interval from the first transition (i.e. pair of meeting intervals involving the same pair of objects) to the last such transition. Clearly there are other possibilities too. Ultimately this is probably a domain dependent decision. For our purposes here, we take the list of all intervals in λ. We also note that this head can be used as a relational fact in some other higher level model (e.g. a global model that describes the whole turn-around).

4 Typed ILP

When the input data is huge and noisy, there are several problems an ILP system can face. One of these is that hypothesis evaluation can take a lot of time because of the size of the data. Also as the noise will tend to make the hypothesis over general: this can trigger many false positives. Using a typed ILP system can speed up evaluation because of typed arguments in the hypothesis and also reduce the number of false positives because of avoiding certain cases where the types of the arguments do not match. In most ILP systems, any type hierarchy of objects is not integrated tightly into the learning process. For example in Progol, types of the objects are used only in mode declarations and since it assumes a flat type hierarchy of the domain, the search procedure cannot take any type hierarchy into consideration. However, a learning system can exploit a type hierarchy to reduce over generalization. For example, if the tracking system sometimes confuses two types of objects (τ_1, τ_2) such that some objects of type τ_1 are misclassified as type τ_2, then standard Progol

can only generalize to an arbitrary object without type restrictions. A variable without type restrictions will be satisfied by any type of object when instantiating the Prolog rule. However a more specific generalization can be enforced by the learning system with a type variable $\tau_1 \sqcup \tau_2$ from the type hierarchy which is satisfied[2] only by objects of type τ_1 or τ_2 thereby reducing false positives. In this section, we explain how the type hierarchy of the domain can be tightly integrated into any ILP system.

If we wish to use an existing Prolog engine for hypothesis evaluation then some way of encoding type using terms must be found. There are several ways of doing this depending on whether the structure of the object type hierarchy is a tree or a lattice. We use the type representation method presented in [4] that can deal with tree structured type hierarchy and develop a refinement operator and incorporate this representation in the hypothesis search procedure.

We will write $\tau_i \sqsubset \tau_j$, if τ_i is a subtype of τ_j and $\tau_i \neq \tau_j$. For each type τ_i, we introduce a rank 1 functor symbol τ_i'. Then every object O of type τ_n in a hypothesis can be represented by the term $\tau_1'(\tau_2'(\ldots \tau_n'(O)\ldots))$ where $\tau_1, \ldots, \tau_n$ is the maximal sequence of types such that $\tau_n \sqsubset \ldots \sqsubset \tau_2 \sqsubset \tau_1$. We denote this representation function by Υ.

For example, let $s1, s2, s3, s4$ be types such that $s3 \sqsubset s2 \sqsubset s1$ and $s4 \sqsubset s2 \sqsubset s1$. Then any object O of type $s3$ can be represented as follows:
$$\Upsilon(O) = s1(s2(s3(O)))$$
An advantage of using this representation is that ordinary unification can be used to determine whether two types are compatible. For example, the types of two objects, O_i and O_j are not compatible if $\Upsilon(O_i)$ is not unifiable with $\Upsilon(O_j)$, for example: $s1(s2(s3(O_i)))$ will not unify with $s1(s2(s4(O_j)))$ but will unify with $s1(s2(O_k))$.

4.1 Type refinement operator

A refinement operator is used to traverse through the hypothesis lattice. There are two types of refinement operators: upward and downward. We write $H_g \succeq H_s$ if H_g is a more generic hypothesis than H_s. If we assume that the top most element of the hypothesis lattice is the most generic hypothesis and the bottom most hypothesis is the most specific hypothesis, then the upward refinement operator can be defined as follows (the downward refinement operator can be defined in a similar fashion):

Let Ł be the set of all possible hypotheses. An (upward) refinement operator ρ is defined such that for a hypothesis H, ρ produces only generalizations of H, $\rho(H) = \{H_g \mid H_g \succeq H, H_g \in Ł\}$.

We define the (upward) type refinement operator ρ_t as an operator that generalizes the object types of H. Apart from object types, the structure of H and members of $\rho_t(H)$ is identical.

Def 4. *We define the type generalizing operator as follows:*
$$generalize\_type(\tau_1'(\tau_2'(\ldots \tau_{n-1}'(\tau_n'(O))\ldots)))) =$$
$$\tau_1'(\tau_2'(\ldots \tau_{n-1}'(O)\ldots))$$

The type refinement operator, ρ_t, applies the $generalize\_type$ operator to a selected object term present in a hypothesis resulting in a more generic hypothesis.

The type refinement operator ρ_t is *non-optimal* as it generates redundant hypotheses because type generalization may occur in multiple orders. Since the cost of generating hypotheses is negligible when compared to the cost of evaluating [18] them, this is not of great concern if care is taken to identify and discard duplicate hypotheses.

5 Experimental Results and Evaluation

Evaluation of this approach to event learning was performed in the airport apron domain. For experiments, 7 turn-arounds [3] were used where each turn-around was shot using 8 cameras from different angles and each video is on average 50,000 frames long (15 frames per sec). Tracking was done on each of the 8 videos of a turn-around and fused together to get 3D data on ground plane. The tracking data is noisy because of low quality, bad light and weather conditions and low contrast of cctv videos. The noise can be presence of phantom objects, some objects missing, wrong types of vehicles, inaccurate bounding boxes, broken trajectories, object identity inconsistencies etc. which are standard problems in any computer vision tracking system. Each turn-around is separately processed to get relational data that consists of a set of spatial relations among the vehicles and zones (Fig 4) on the apron which are drawn according to the International Air Transport Association (IATA) specifications. Some automatic post processing is needed to remove spurious relations that are present because of noise in the tracked data. For example if some object disappears for a while (because of occlusion) and reappears, there will be a break in the relational continuity and some very short duration relations are present because of flickering in the bounding boxes. Prolog rules that decide the temporal rules among intervals are considered as background information in the ILP system.
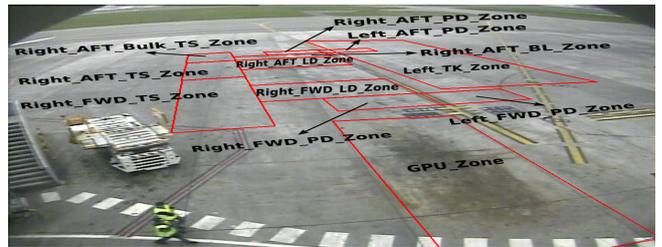
Figure 4: Zones defined in the airport apron area

Currently we are working with 5 IATA events namely Rear Loading, Rear Unloading, Aircraft Arrival, Aircraft Departure, Jet Bridge Positioning. Each event is of different duration and involves different types of vehicles and occurs at different places in the apron. Within each event, there is high variability because of noise in tracking and also because of irrelevant objects entering the event scene. Positive example instances are provided by knowledge engineers who have expertise in the IATA protocols and apron activities. Note that some events might not be present or may occur multiple times in some turn-arounds.

The Herbrand Interpretations of an event obtained using the deictic intervals and regions are used as positive examples and the complement as negative region of the event to start the system. An example of such an interpretation for the event Rear Loading is given in Fig(5). The learning algorithm constructs the *most-specific clause* and uses Progol's refinement operator for hypothesis refinement. Besides this, we also use our type refinement operator. The object type hierarchy in our domain can be represented as a tree (Fig 6). So an object O of type *Loader* can be represented as *Object(Vehicle(Heavy_Vehicle(Loader(O))))* and is not compatible with *Object(Vehicle(Heavy_Vehicle(Mobile_Stairs(Y))))* but compatible with *Object(Vehicle(heavy_Vehicle(Z)))*. Note that intervals and zones are not included in the type hierarchy since these do not suffer

---

[2] A variable of type $\tau_1 \sqcup \tau_2$ can unify with variable of type τ_1 or τ_2

[3] A turn-around is the duration of a plane entering and leaving the apron area
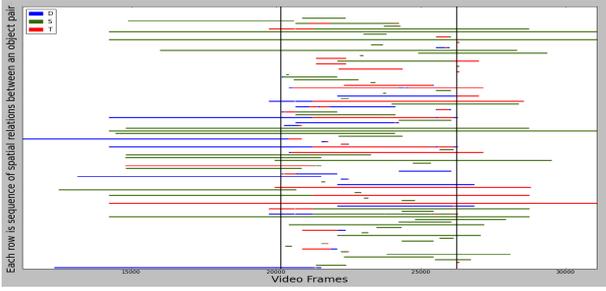
Figure 5: Visualization of a Herbrand Interpretation for a Rear Loading event instance. Each row represents a sequence of spatial relations between an object pair. Objects involved and their types are not shown (best viewed in colour).

from visual noise. While traversing the hypothesis lattice, candidate hypotheses are evaluated on positive examples to get their positive score. To get the negative score the candidate hypothesis is queried in the complement of the positive region and the number of instances satisfying the query is considered as the negative score. This means the negative examples are based on the hypothesis that is being evaluated. An alternative way of getting negative examples would be to randomly get some chunks from the negative region and use them as negative examples. This may be appropriate for getting hypotheses for classification purposes, but may be of little use for hypotheses that have to be used for recognition purposes in continuous video data.
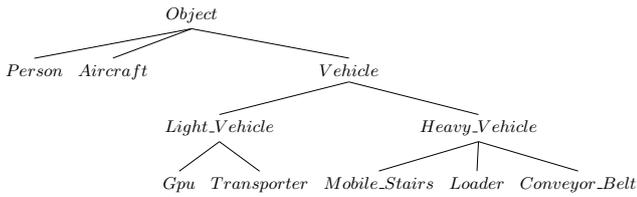
Figure 6: Tree structured object type hierarchy in the airport apron domain.

We used very generic constraints to prune the hypothesis space including pruning hypotheses with dangling variables; hypotheses whose terms in the body are not well connected; those having temporal relations with intervals that do not appear in spatial relations etc.

The whole system is implemented in Python. For speed, some modules are implemented in Cython and SWI-Prolog is used as the underlying Prolog engine for evaluating hypotheses. Sample rules learned for Aircraft Arrival and Rear Loading events are given below. For example the Aircraft Arrival rule can be interpreted as: an aircraft arrives when the aircraft bounding box *surrounds* the right_AFT_Bulk_TS_Zone and then moves forward thereby changing the relation to *touches*. This happens when the aircraft arrives and is moving to its position. The rule also correctly identifies that this bounding box should belong to an object of type aircraft.

```
aircraft_arrival([intv(T1,T2),intv(T3,T4)]) :-
          surrounds(obj(aircraft(V)),
                    right_AFT_Bulk_TS_Zone,
                    intv(T1,T2)),
          touches(right_AFT_Bulk_TS_Zone,
                    obj(aircraft(V)),
                    intv(T3,T4)),
          meets(intv(T1,T2),intv(T3,T4)).
```

```
rear_loading([intv(T1,T2),intv(T3,T4)]) :-
          touches(left_TK_Zone,
                    obj(veh(heavy_veh(V1))),
                    intv(T1,T2)),
          touches(obj(veh(V2)),
                    left_TK_Zone,
                    intv(T3,T4)),
          during(intv(T3,T4),intv(T1,T2)).
```

We followed the standard machine learning *leave-one-out* testing methodology for testing performance. All turn-arounds except one are used for training and the remaining one is used as test case. This process is iterated until each turn-around is used as test case exactly once. The results of our experiments are summarised in Tables 1 & 2. The second column shows the number of positive instances in the 7 turn-arounds. The third column shows the number of times the event hypothesis fired in positive spatio-temporal regions in all 7 testing iterations. The last column gives the number of times the event hypothesis fired in the negative spatio-temporal region in each of the testing iterations.

From the tables it is clear that using type information can increase the overall performance. Some events are hard to recognize because of bad tracking data, for example, Jet Bridge Positioning.

Events	pos_ex	△	□
Rear Loading	10	8	[0, 1, 1, 1, 1, 3, 0]
Rear Unloading	5	4	[1, 2, 0, 0, 0, 0, 1]
Aircraft Arrival	7	7	[1, 0, 1, 2, 0, 2, 3]
Aircraft Departure	7	4	[14, 2, 0, 0, 1, 1, 0]
Jet Bridge Positioning	7	4	[0, 2, 0, 2, 1, 1, 1]

△ True Positives (all iterations) □ False Positives (in each iteration)

Table 1: Recognition results using types (*leave-one-out* testing)

Events	pos_ex	△	□
Rear Loading	10	7	[1, 0, 2, 1, 4, 0, 2]
Rear Unloading	5	1	[4, 1, 0, 0, 1, 1, 0]
Aircraft Arrival	7	2	[2, 3, 0, 8, 1, 6, 16]
Aircraft Departure	7	4	[3, 3, 7, 4, 1, 1, 1]
Jet Bridge Positioning	7	1	[0, 0, 2, 11, 8, 1, 4]

△ True Positives (all iterations) □ False Positives (in each iteration)

Table 2: Recognition results *without* using types (*leave-one-out* testing)

The detailed recognition results for some events are given in Fig 7. Each subplot is for an event and in each subplot a strip represents the time line in a video. The ground truth (GT) is represented as blue patch and recognized events (RE) are represented as green and transparent patch for true positive and false positive respectively. In some cases the temporal extent of recognized event instances is long because of some spatial relations that are important in the event extends beyond the deictic interval of the event.

6 Conclusion and Future Work

In this paper, we have proposed and successfully applied a novel supervised framework for learning event models from a huge, complex and noisy video dataset. We also presented an upward type refinement operator that reduces false positives and gives semantically meaningful hypotheses. This type refinement operator is generic and can be applied in any domain where the object types form a hierarchy (though the specific current implementation using functors requires a tree structured hierarchy; having a tree structured hierarchy is beneficial from a computational viewpoint in limiting the type generalisations i.e. there are no multiple ancestors). The experimental results are
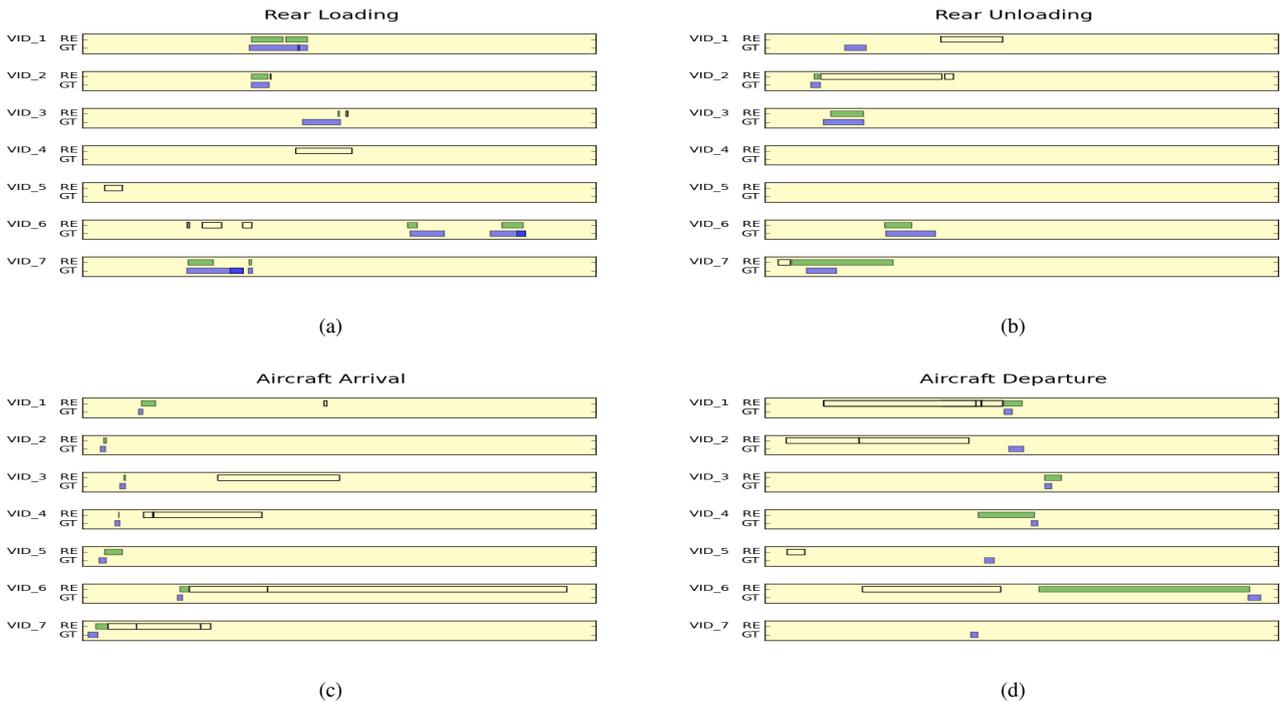
Figure 7: Recognition of events using the learned event models using *leave-one-out* testing (best viewed in colour). In each strip: GT denotes ground truth intervals (blue), RE denotes retrieved event intervals (green: true positive, transparent:false positive). Length of all the strips is set to length of the longest video.

promising and since the data used is from a real world scenario, the system has potential for deployment in the real world.

There is scope for much further work. The models that we learn are local and do not see a global picture when recognizing an event i.e. there is no account taken of global constraints such as event ordering etc (for example Aircraft Arrival occurs before Aircraft Departure). One possible direction is to learn and use a global model (or higher level event models) to decide when to fire the individual events. The spatial relations that we used are very coarse grained. It is possible that fine grained spatial relations might give more semantically meaningful models whilst using fuzzy spatial relations [16] might cope better with the noise present in the tracking data. Event models can be made more efficient by learning additional constraints like duration of the events, number of vehicles involved etc and searching more of the hypothesis space using parallel ILP techniques. Some domains might not have a well defined tree-like object type hierarchy. In such cases a lattice structured type hierarchy is more suitable though this will increase the search space enormously.

7 Acknowledgements

REFERENCES

[1] M. Albanese, V. Moscato, A. Picariello, VS Subrahmanian, and O. Udrea, 'Detecting Stochastically Scheduled Activities in Video', in *Proc. of IJCAI*, pp. 1802–1807, (2007).

[2] J.F. Allen, 'Maintaining knowledge about temporal intervals', *Communications of the ACM*, **26**(11), 832–843, (1983).

[3] H. Blockeel, L. De Raedt, N. Jacobs, and B. Demoen, 'Scaling up inductive logic programming by learning from interpretations', *Data Mining and Knowledge Discovery*, **3**(1), 59–93, (1999).

[4] A. Bundy, L. Byrd, and CS Mellish, 'Special-purpose, but domain-independent, inference mechanisms', in *Progress in Artificial Intelligence*, pp. 93–111. London: Ellis Horwood, (1985).

[5] L. De Raedt, 'Logical settings for concept-learning', *Artificial Intelligence*, **95**(1), 187–201, (1997).

[6] J. Fernyhough, A. G. Cohn, and D. C. Hogg, 'Constructing qualitative event models automatically from video input', *Image and Vision Computing*, **18**(2), 81–103, (2000).

[7] J. Ferryman, M. Borg, D. Thirde, F. Fusier, V. Valentin, F. Bremond, M. Thonnat, J. Aguilera, and M. Kampel, 'Automated scene understanding for airport aprons', *LNCS*, **3809**, 593, (2005).

[8] Anthony Hoogs and A. G. Amitha Perera, 'Video activity recognition in the real world', in *Proc. of AAAI*, pp. 1551–1554, (2008).

[9] Y.A. Ivanov and A.F. Bobick, 'Recognition of visual activities and interactions by stochastic parsing', *IEEE Trans. PAMI*, **22**(8), (2000).

[10] T. Könik and J.E. Laird, 'Learning goal hierarchies from structured observations and expert annotations', *Machine Learning*, **64**(1), 263–287, (2006).

[11] I. Laptev and P. Pérez, 'Retrieving actions in movies', in *ICCV*, (2007).

[12] S. Moyle and S. Muggleton, 'Learning programs in the event calculus', *LNAI 1297*, 205–212, (1997).

[13] S. Muggleton, 'Inverse entailment and Progol', *New Generation Computing*, **13**(3&4), 245–286, (1995).

[14] C.J. Needham, P.E. Santos, D.R. Magee, V. Devin, D.C. Hogg, and A.G. Cohn, 'Protocols from perceptual observations', *Artificial Intelligence*, **167**(1-2), 103–136, (2005).

[15] Y. Sasaki, 'Applying type-oriented ILP to IE rule generation', in *Proc. Workshop on Machine Learning and Information Extraction*, (1999).

[16] S. Schockaert, M. De Cock, and E. E. Kerre, 'Spatial reasoning in a fuzzy region connection calculus', *Artif. Intell.*, **173**(2), 258–298, (2009).

[17] M. Sridhar, A. G. Cohn, and D. C. Hogg, 'Learning functional object-categories from a relational spatio-temporal representation.', in *Proc. of ECAI*, pp. 606–610, (2008).

[18] A. Srinivasan, 'A study of two sampling methods for analyzing large datasets with ILP', *Data Mining & Knowledge Discovery*, **3**(1), (1999).

[19] Tao Xiang and Shaogang Gong, 'Video behavior profiling for anomaly detection', *IEEE Trans. PAMI.*, **30**(5), 893–908, (2008).