Using Spatio-Temporal Continuity Constraints to Enhance Visual Tracking of Moving Objects

Brandon Bennett, and **Derek R. Magee** and **Anthony G. Cohn** and **David C. Hogg** [1]

Abstract. We present a framework for annotating dynamic scenes involving occlusion and other uncertainties. Our system comprises an object tracker, an object classifier and an algorithm for reasoning about spatio-temporal continuity. The principle behind the object tracking and classifier modules is to reduce error by increasing ambiguity (by merging objects in close proximity and presenting multiple hypotheses). The reasoning engine resolves error, ambiguity and occlusion to produce a most likely hypothesis, which is consistent with global spatio-temporal continuity constraints. The system results in improved annotation over frame-by-frame methods. It has been implemented and applied to the analysis of a team sports video.

1 INTRODUCTION

No computer vision algorithm for tracking or object classification is perfect under real-world conditions. Object trackers have difficulty with complex occlusions (e.g. in crowded pedestrian scenes) and classifier algorithms rarely give 100% accuracy, even on restricted data sets. We propose a framework for taking the imperfect output of an object tracker and classification system and refining it based on principles of logical consistency and the spatio-temporal continuity of physical objects, to produce a far more accurate scene annotation.

The low-level tracking and classification modules of our system model ambiguity and error by assigning probabilities for the presence of objects within bounding boxes in each video frame. This output is passed to a reasoning engine which constructs a ranked set of possible models, consistent with the requirements of object continuity. The final output is then a globally consistent spatio-temporal description of the scene which is maximally supported by the probabilistic classifier.

A number of researchers have attempted to deal with object occlusion (and the resultant tracking problems) by attempting to increase reliability of tracking and classification in the presence of occlusion [6, 5, 3] or to minimise occlusion by using multiple cameras [2]. Our approach has more similarity with the methods of [10, 12, 15, 7]. Rather than attempt to directly classify occluding objects, they employ some kind of semantic constraints to reason about the occlusion taking place. None of these systems performs more than frame-by-frame reasoning or allows for error in the underlying low-level tracking and recognition algorithms. By contrast, our system performs long-term reasoning about object-blob associations over extended sequences of frames. By maintaining spatio-temporal consistency over sequences, many local imperfections and ambiguities in the low-level data are eliminated. In fact this approach is closely related to works in the area qualitative spatio-temporal reasoning, which model objects in terms of their spatio-temporal histories [11, 1, 4]. But as far as we are aware this approach has not previously been applied to computer vision.

[1] School of Computer Studies, University of Leeds, Leeds LS2 9JT, UK. Email: {brandon|drm|agc|dch}@scs.leeds.ac.uk

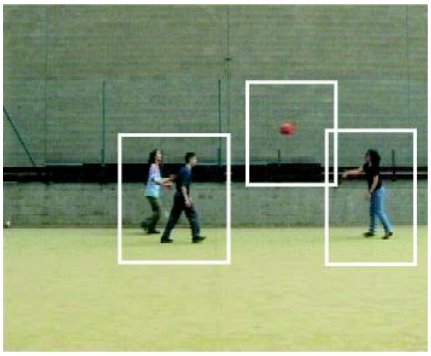

Figure 1. Example Output from Blob Tracker

2 THE TRACKER MODULE

Our tracker system is an extension of the car tracker of [8], which in turn is based on [13]. The system tracks coherent 'blobs' using a loose model of object position, size, velocity and colour distribution which is updated on the fly. In reality, a one-to-one mapping between blobs and objects cannot be assumed. Our spatio-temporal model explicitly takes account of the possibility of multiple occupancy, allowing us to increase tracker ambiguity in cases where an error may occur (e.g. when objects are close or occluding) by merging overlapping blobs (blobs are also be split horizontally or vertically if they evolve into easily separable components). This technique reduces to near zero the occurrence of tracker errors such as lost/additional objects in our chosen example domain. Other errors may be eliminated by more traditional techniques such as ignoring transient objects (i.e. objects present only for a very short period of time). An example of the tracker output for a simple basketball scene is given in figure 1. The boxes shown are a visualisation of our statistical foreground model which includes the variance of blob pixel locations about the mean blob location. The bounding box for each blob corresponds to a threshold on these distributions. In the current implementation this is set to 3 standard deviations.

3 OBJECT CLASSIFICATION

Our main priority is to have an object recognition/classification system that has low computational cost and so can be incorporated into an on-line system. We accept that such a system may not have 100% accuracy, however our higher level processing (see section 4) is designed to cope with imperfect and ambiguous input. Hence, we have implemented a system that combines a number of simple binary classifiers using a Bayesian combination mechanism [9].

Currently, we only use binary classifiers based on colour. These are extremely simple, but by combining several such classifiers usable results can be obtained. Let $\mathcal{C}$ be the vector of classifier outputs and $\mathcal{C}_n$, the output of classifier n. From a set of image samples we

can learn for each object class κ and each classifier the probabilities $P(\mathcal{C}_n = \mathbf{t} \mid \kappa)$, $P(\mathcal{C}_n = \mathbf{f} \mid \kappa)$, $P(\mathcal{C}_n = \mathbf{t} \mid \neg\kappa)$, $P(\mathcal{C}_n = \mathbf{f} \mid \neg\kappa)$.

For a classifier output $\mathcal{C}$, let $P(\mathcal{C}_n|\ \kappa)$ be the probability of obtaining the value $\mathcal{C}_n$ given the presence of object κ; and let $K = P(\neg\kappa)/P(\kappa)$, calculated over the sample set. Then using Bayes law we have:

$$P(\kappa|\mathcal{C}) = \frac{P(\mathcal{C}|\kappa)P(\kappa)}{P(\mathcal{C})} = \frac{P(\mathcal{C}|\kappa)}{P(\mathcal{C}|\kappa)+KP(\mathcal{C}|\neg\kappa)}$$

$$= \frac{\prod_{n=1}^{N} P(\mathcal{C}_n|\kappa)}{\prod_{n=1}^{N} P(\mathcal{C}_n|\kappa)+K\prod_{n=1}^{N} P(\mathcal{C}_n|\neg\kappa)}$$

A list of probabilities of objects detected with higher than a minimum threshold probability (we currently use 0.01) is passed to the spatio-temporal consistency checker. The figures allow hypothesis ranking in cases of ambiguity. A hypothesis rejected at this stage may be still be asserted in the final globally optimised output of the consistency checker (i.e. if past or future information supports it).

4 SPATIO-TEMPORAL CONTINUITY

We seek a solution that both maximises statistical correlation with this output and is globally consistent with requirements of spatio-temporal continuity of objects. For a model to be physically possible, it must satisfy the following spatio-temporal constraints:

C1) *exclusivity* — an object cannot be in more than one place at the same time;
C2) *continuity* — an object's movement must be continuous (i.e. it cannot instantaneously 'jump' from one place to another).

With any statistical classifier, an object may be detected to a high degree of probability in two widely separated locations. This kind of error is fairly easy to eliminate on a frame by frame basis. We can consider all possible assignments of different objects to the tracked boxes in each frame and choose the combination that maximises the summed probabilities of object to box correspondences.[2]

The continuity of an objects position over time is much more difficult to model; and considerable problems arise in relating continuity constraints to tracker output. The main problem is that of occlusion: if an object moves behind another it is no longer detectable by the tracker; so, under a naive interpretation of the tracker and recognition system outputs, objects will appear to be discontinuous.

As well as enforcing spatio-temporal constraints, we want to find a hypothesis that is maximally supported by the probabilistic output of the object classifier for each tracked box. However, the classifier was trained to identify single objects, whereas in tracking a dynamic scene there will often be several objects in a box. This means there is no completely principled way to interpret the output of the classifier. Nevertheless, it is reasonable to assume that although there is a large amount of error and uncertainty in the low-level output, it does give a significant indication of what objects may be present.

Local continuity information is provided by the low-level tracker. This assigns to each blob's bounding box an identification tag (a number), which is maintained over successive frames. Newly split or merged boxes are assigned new tags, but the tag of their parent box in the previous frame is also recorded. So each box is associated with sets of its *parent* and *child* boxes. The parent/child relation determines a 'box continuity graph' (see figure 2). Our algorithm exploits the structure of this directed graph.

The reasoning algorithm involves a restructuring of the tracker/classifier output. A tracked box b can be represented by a tuple, $\langle f, \mathbf{geom}, \mathbf{pa}, \mathbf{ch}, \mathbf{Class}\rangle$, where f is the frame number, **geom** is the box geometry, **pa** is the set of its parent boxes (in the frame $f-1$), **ch** is the set of its child boxes (in the frame $f+1$) and **Class** represents the statistical output of the object classifier applied to this box. Frame numbers are any continuous subset of non-negative integers. The initial frame will be denoted f_0. For convenience we introduce functions $f(b)$, $\mathbf{geom}(b)$, $\mathbf{pa}(b)$, $\mathbf{ch}(b)$ and $\mathbf{Class}(b)$ to refer to the corresponding information associated with box b. The set of all boxes in the tracker/classifier output will be denoted by BOXES.

[2] This assumes that the classifier is capable of identifying unique objects (such as particular people) rather than classes of similar objects. In situations where there may be multiple objects of the same class, the exclusivity constraint must be weakened.

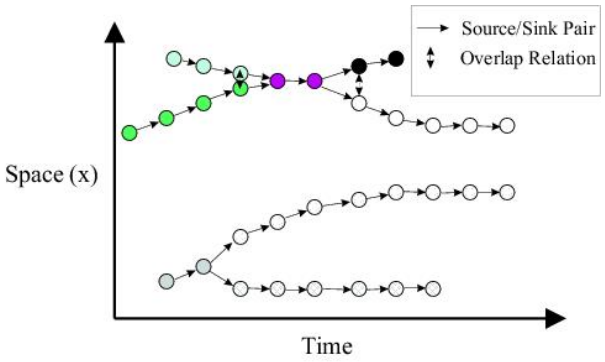

Figure 2. Tracker output expressed as a 'box continuity' graph

The parent and child boxes of a given box function as 'sources' and 'sinks' for that box. Occupants of a box must have come from its parent boxes and go to its the child boxes. If no objects enter or leave a scene, each box is associated with at least one source in the previous frame and one sink in the next frame. But where two or more boxes become merged, the merged box will have more than one source; and, when a box is split, it will have two or more sinks. Later we extend this idea to handle objects entering/leaving the scene.

Within the box continuity graph are chains of boxes linked by a unique source-sink relationship. These are shown in figure 2 as subgraphs, shaded in a particular colour, corresponding to the 'history' of a tracked box between splitting and merger events. If the tracker were perfectly accurate, the objects occupying any box would be constant along any of these sub-graphs. Because of this, it is useful to introduce an abstract data object which we call a *spatio-temporal box* (*ST-box* for short). ST-boxes will normally be denoted by symbols S_i. In the tracker output, an ST-box corresponds to a temporally continuous sequence of boxes which have the same tag. In terms of the **ch** function, this is a maximal sequence $[b_1, \ldots, b_n]$ such that for each i in the range $1 \le i \le n-1$ we have $\mathbf{ch}(b_i) = \{b_{i+1}\}$.

An ST-box is an ordered set S of boxes indexed by frame numbers in the range $\mathbf{s}(S) \ldots \mathbf{e}(S)$, where $\mathbf{s}(S)$ and $\mathbf{e}(S)$ are the start and end frames of the period over which S exists. The partial function **box-at**(f, S) denotes the box $b \in S$ whose frame number is f (only defined for f s.t. $\mathbf{s}(S) \le f \le \mathbf{e}(S)$). Each box $b \in$ BOXES is a member of a unique ST-box denoted by $\mathbf{st}(b)$. ST-BOXES $= \{S \mid (\exists b \in \text{BOXES})[\mathbf{st}(b) = S]\ \}$ is the set of all ST-boxes.

4.1 Grouping Boxes into 'Envelopes'

The box continuity graph is only indirectly related to the trajectories of actual moving objects in the scene. Several issues complicate the relationship. Firstly, there is the basic problem caused by proximal and/or occluding objects, which means that a box may be occupied by several objects. This is compounded by the possibility that (because of occlusion or limited resolution) objects sometimes transfer between tracker boxes without being independently tracked.

These problems suggest the idea that more reliable tracking might be achieved by modelling object locations at coarser level than individual boxes. Hence, we introduce a data object that we call an *envelope*. Intuitively, an envelope is simply a (maximal) cluster of overlapping ST-boxes. This is illustrated in figure 3. Solid lines indicate the positions of box boundaries over a sequence of frames, so the area swept out by a moving box represents an ST-box. Dashed lines show the division of this structure into envelopes.

The definition of an envelope depends upon specifying an overlap relation between boxes, which we write as $\mathbf{Overlap}(b_1, b_2)$. Since,

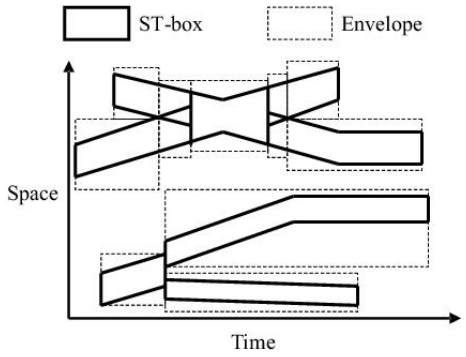

Figure 3. Deriving Spatio-temporal Envelopes from Tracker Output

the boxes bound the positions of objects to a high statistical probability, the simple geometric overlap relation between boxes is highly correlated to the possibility of object transfer between them. Because of the statistical nature of the bounding boxes, we may vary the scale of the boxes in order to yield stricter or weaker overlap relations. In the current work we have not attempted to tweak the scaling to produce the optimal output for our data. We have chosen a scale where the box bounds the object with 97% probability. Limited experimentation indicates that this threshold is a good choice for our data.

To formally define an envelope we need to describe the spatial clustering of ST-Boxes. A *local group* is a set of ST-boxes that are related by the transitive closure of the geometric $\mathbf{Overlap}$ relation. The relation $\mathbf{Overlap}^*(b_1, b_n)$ holds just in case there is some sequence of boxes $b_1, \ldots, b_n$, such that for $1 \leq i \leq n-1$ we have $\mathbf{Overlap}(b_i, b_{i+1})$. At each frame over which it exists, any given ST-box stands in the $\mathbf{Overlap}^*$ relation to a set of other ST-boxes, which is its local group at that frame. Since $\mathbf{Overlap}^*$ is reflexive, symmetric and transitive, at each frame the local groups form equivalence classes over the set of ST-boxes. Given an ST-box S and a frame f (with $\mathbf{s}(S) \leq f \leq \mathbf{e}(S)$), we can define the function **local-g**$(f, S) = \{S' \mid \mathbf{Overlaps}^*(\textbf{box-at}(f, S), \textbf{box-at}(f, S'))\}$.

A *constant-local-group spatio-temporal box* (*CLG-ST-box*) is a temporal sub-division of an ST-box, over which it is a member of a constant local group. The beginning and end frames of CLG-ST-boxes are determined by changes in the overlap relation between boxes. The division of ST-boxes into CLG-ST-boxes is indicated in figure 3. Divisions coincide with temporal boundaries of envelopes.

Formally, a CLG-ST-box is a subset of an ST-box corresponding to a maximal temporally continuous sequence of frames over which **local-g**(f, S) takes a constant value. Each ST-box determines a unique CLG-ST-box **clg-st**(f, S) for each frame over which it exists. As with ST-boxes we use the functions $\mathbf{b}(C)$ and $\mathbf{e}(C)$ to refer to the beginning and end frame of a CLG-ST-box, C. We also write $\mathbf{st}(C)$ to refer to the unique ST-box of which C is a sub-sequence. Since CLG-ST-boxes participate in the same local group throughout their history, we can define **local-g**$(C) =$ **local-g**$(\mathbf{b}(C), \mathbf{st}(C))$.

We now define an envelope as a maximal set of overlapping CLG-ST-boxes. Any CLG-ST-box is a member of a unique envelope, given by $\mathbf{env}(C) = \{$**clg-st**$(\mathbf{b}(C), S) \mid S \in$ **local-g**$(C)\}$. For an envelope E, the functions $\mathbf{b}(E)$ and $\mathbf{e}(E)$ denote the beginning and end frames of its existence. By definition, all its constituent CLG-ST-boxes have the same beginning and end frames. Each box in the tracker input is related to a unique envelope given by $\mathbf{env}(b) = \mathbf{env}($**clg-st**$(f(b), \mathbf{st}(b)))$. The set of envelopes derived from the tracker is given by $\mathsf{ENVELOPES} = \{E \mid (\exists b \in \mathsf{BOXES})[E = \mathbf{env}(b)]\}$. The set of envelopes that exist at frame f is give by **envs-at**$(f) = \{E \in \mathsf{ENVELOPES} \mid \mathbf{b}(E) \leq f \leq \mathbf{e}(E)\}$.

4.2 Continuity of Envelope Occupancy

Although envelopes give a coarser demarcation of object locations than individual boxes, they provide a much more reliable basis for determining continuity. By definition, two different envelopes cannot spatially overlap (otherwise they would be parts of a larger envelope). So there is an very low probability that an object can transfer between envelopes without being detected. Hence, our algorithm assumes that the occupancy of an envelope is constant throughout its existence.

The relation $\mathbf{Occ}(\mathbf{l}, E)$ means that object **l** is present in envelope E. The exclusivity constraint **C1**, corresponds to the requirement that no object can occupy two distinct spatio-temporal envelopes that overlap in time. Temporal overlapping of envelopes is defined by $\mathbf{TOE}(E_1, E_2) \equiv_{def} \exists f[\mathbf{b}(E_1) \leq f \leq \mathbf{e}(E_1) \wedge \mathbf{b}(E_2) \leq f \leq \mathbf{e}(E_2)]$. So the **C1** constraint is equivalent to

C1) $(\mathbf{TOE}(E_1, E_2) \wedge \mathbf{Occ}(\mathbf{l}, E_1) \wedge \mathbf{Occ}(\mathbf{l}, E_2)) \rightarrow (E_1 = E_2)$

Envelopes have a continuity graph structure similar to that of boxes (see figure 3). In fact an *envelope continuity graph* can be formed directly from the box continuity graph by unifying all nodes derived from boxes in the same envelope. Functions $\mathbf{pa}(E)$ and $\mathbf{ch}(E)$ returning parent and child sets for envelopes can be derived from the corresponding relations between their constituent boxes.

Since we allow objects to enter or leave the scene we must keep track of off-scene objects. We do this by introducing virtual, *off-scene envelopes* to our model. For simplicity, in the current implementation we assume there is only one off scene location. Transfer to and from off-scene envelopes can only occur when a tracker box is either created or disappears. Off-scene envelopes do not have any spatial structure; so we identify them simply with a frame pair $\langle f_b, f_e \rangle$ representing the beginning and end frames of their existence. f_b must be the beginning frame of an envelope or the frame immediately after the end frame of an envelope. f_e is either the end frame of an envelope or the frame before the beginning of an envelope. The set OS-ENVELOPES contains all beginning/end frame pairs with no other beginning or end frames occurring between them. Thus the off scene envelopes form a partition of the frame sequence.

The set including both on-scene and off-scene envelopes will be denoted by $\mathsf{ENVELOPES}^+ = \mathsf{ENVELOPES} \cup \mathsf{OS\text{-}ENVELOPES}$. The predicate $\mathbf{OS}(E)$ holds just in case E is an off-scene envelope.

To define the continuity relation between envelopes, we first define a successor relation $\mathbf{Suc}(E_1, E_2) \equiv_{def} ((\mathbf{e}(E_1) + 1) = \mathbf{b}(E_2))$, which holds when the end of E_1 is at the frame immediately before the beginning of E_2. We can now define the relation **Source-Sink**(E_1, E_2), meaning that envelope E_1 is a source for the objects in envelope E_2. The following formula handles continuity for both on and off-scene envelopes:

$$\begin{aligned}&\textbf{Source-Sink}(E_1, E_2) \equiv_{def} (E_1 \neq E_2) \wedge \\ &\quad [\exists b_1 b_2 [b_1 \in \mathbf{pa}(b_2) \wedge \mathbf{env}(b_1) = E_1 \wedge \mathbf{env}(b_2) = E_2] \\ &\qquad \vee (\mathbf{OS}(E_1) \wedge (\mathbf{pa}(E_2) = \emptyset) \wedge \mathbf{Suc}(E_1, E_2)) \\ &\qquad \vee (\mathbf{OS}(E_2) \wedge (\mathbf{ch}(E_1) = \emptyset) \wedge \mathbf{Suc}(E_1, E_2)) \\ &\qquad \vee (\mathbf{OS}(E_1) \wedge \mathbf{OS}(E_2) \wedge \mathbf{Suc}(E_1, E_2))\,]\end{aligned}$$

In terms of this relation, the continuity constraint **C2**, as applied at the envelope level, is represented by

$$\begin{aligned}\textbf{C2)}\quad &\mathbf{Occ}(\mathbf{l}, E_1) \rightarrow \exists E_2 [\textbf{Source-Sink}(E_1, E_2) \wedge \mathbf{Occ}(\mathbf{l}, E_2)] \\ &\wedge \mathbf{Occ}(\mathbf{l}, E_2) \rightarrow \exists E_1 [\textbf{Source-Sink}(E_1, E_2) \wedge \mathbf{Occ}(\mathbf{l}, E_1)] \\ &\wedge \textbf{Source-Sink}(E_1, E_2) \rightarrow \exists l [\mathbf{Occ}(\mathbf{l}, E_1) \wedge \mathbf{Occ}(\mathbf{l}, E_2)]\end{aligned}$$

Our algorithm will generate possible assignments of object labels to envelopes that satisfy both **C1** and **C2**. It will then choose the one that we consider 'best supported' by the classifier outputs.

4.3 Likelihood of Box and Envelope Occupancy

Applied to a box b, the output of the classifier module takes the form $\mathbf{Class}(b) = \{\langle \mathbf{l}_1, p_1 \rangle, \ldots, \langle \mathbf{l}_n, p_n \rangle\}$, where p_i is the probability of $\mathbf{l}_i$ being a correct label for the object(s) in box b. Each statistic is an independently computed probability based on the assumption that there is only one object in the box. Thus, the figures are not normalised and cannot be reliably applied to multiple-occupancy boxes. However, we assume that, even in multi-object cases, the figures give

an approximate indication of the likelihood of an object being in the box. Hence, we regard the number p_i directly as a 'vote' for the presence of object $\mathbf{l}_i$ in box b. We denote this vote value by $\mathbf{vote}(\mathbf{l}, b)$.

Our continuity constraints operate at the level of envelopes rather than boxes. Thus, we need to convert the votes for box occupancy into votes for an object being in a given envelope. Dependencies between observations mean that a statistically valid function would be extremely difficult to define. Hence we define a plausible but *ad hoc* voting function. Devising and evaluating more realistic voting functions is a subject of ongoing work. Nevertheless, as will be seen below, our crude voting metric is already good enough to significantly improve tracking reliability.

For an envelope E and an object $\mathbf{l}$, we first compute for each CLG-ST-box $C \in E$ the sum of the box votes for $\mathbf{l}$ over all frames for which C exists: $\mathbf{vote}(\mathbf{l}, C) = \sum_{\mathbf{b}(C) \leq f \leq \mathbf{e}(C)} \mathbf{vote}(\mathbf{l}, \textbf{box-at}(f, C))$ To get the vote for the object to be in an envelope we take the maximum of its votes for each CLG-ST-box in the envelope:

$$\mathbf{vote}(\mathbf{l}, E) = \mathsf{Max}\{\, v \mid (\exists C \in E) \wedge \mathbf{vote}(\mathbf{l}, C) = v \} \,.$$

To normalise label support values we compute **frac-vote**$(\mathbf{l}, E)$, which is the fractional vote for $\mathbf{l}$ with respect to the total votes for all labels in the classifier domain $\{\mathbf{l}_1, \ldots, \mathbf{l}_N\}$.

In determining the support given by an envelope to a given set of labels, we wish to impose a strong bias that favours the smallest possible number of objects being assigned to the box. We use the following vote function, where the number n of labels assigned to an envelope directly reduces the overall vote. Let $\mathbf{dur}(E) = \mathbf{e}(E) - \mathbf{b}(E) + 1$ be the duration of the envelope in frames. We then define

$$\mathbf{vote}(\{\mathbf{l}_1, \ldots, \mathbf{l}_n\}, E) = \left(\sum_{i=1\ldots n} (\textbf{frac-vote}(E, \mathbf{l}_i) + N - n\right) \cdot \mathbf{dur}(E)$$

4.4 Finding the Best Global Labelling

In all but the simplest cases, it is infeasible to evaluate votes for all spatio-temporally consistent object labellings of an extended scenario. Whenever a box splits, there are several possible assignments of the original box occupants to the newly created boxes. Thus the number of possible solutions grows exponentially with time. However, by taking a *dynamic programming* approach, the optimal solution can be found by an algorithm whose complexity is linear in time. Thus, if the number of objects is small, solutions for arbitrarily long sequences can be computed effectively.

Our algorithm first computes the set of envelopes and the **Source** relation from the tracker output. An *envelope change frame* (ECF) is a frame that is the start of some envelope, or is a frame immediately after some envelope ceases to exist (e.g. when it moves off the scene). We build a model by starting at the initial frame and progressing through each successive ECF. Since envelope occupancy remains constant between ECFs, this model determines a complete assignment to all envelopes at all frames. For any ECF f (including the initial frame f_0) the next ECF after f is denoted by $\mathbf{necf}(f)$. When f is the last ECF in the frame sequence we let $\mathbf{necf}(f) = \mathsf{end}$.

A consistent assignment to all envelopes starting at or before a given ECF f will be called a *partial model* (up to f) and will be denoted $\mathcal{P}_i$ (i is an optional distinguishing index). $\mathbf{lcf}(\mathcal{P}_i)$ denotes the last change frame of envelopes assigned by $\mathcal{P}_i$. A partial model $\mathcal{P}$ is identified with a set $\{\ldots, \langle E_i, A_i\rangle, \ldots\}$, where A_i is a set of labels. The set of labels assigned by $\mathcal{P}$ to envelope E is written $\mathbf{ass}(\mathcal{P}, E)$.

The requirement of spatio-temporal consistency of a partial models means that the exclusivity and continuity constraints **C1** and **C2** must be satisfied, where the occupancy relation **Occ** determined by a model $\mathcal{P}$ is given by $\mathbf{Occ}(l, E)$ iff $l \in \mathbf{ass}(\mathcal{P}, E)$. To compute a spatio-temporally consistent extension of a partial model, we need only know its assignments to the latest envelopes. Hence we define

last-ass$(\mathcal{P}) =$

$$\{\langle E_i, A_i\rangle \mid \langle E_i, A_i\rangle \in \mathbf{ass}(\mathcal{P}) \wedge \mathbf{b}(E_i) \leq \mathbf{lcf}(\mathcal{P}) \leq \mathbf{e}(E_i)\} \,.$$

In order to formalise the notion of one partial model's being an (immediate) extension of another, we define $\mathbf{Extends}(\mathcal{P}', \mathcal{P}) \equiv_{def} ((\mathcal{P}' \setminus \textbf{last-ass}(\mathcal{P}')) = \mathcal{P})$. So the set of all possible (spatio-temporally consistent) extensions of a partial model $\mathcal{P}$ is then $\mathbf{extensions}(\mathcal{P}) = \{\mathcal{P}' \mid \mathbf{Extends}(\mathcal{P}', \mathcal{P})\}$.

$\mathbf{extensions}(\mathcal{P})$ is a key function of our algorithm. To calculate it we first compute and store the **Source-Sink** relation for the tracker input, using the definitions given above. All we need to do is generate all assignments to the envelopes in $\mathcal{E}' = \textbf{envs-at}(\mathbf{necf}(\mathbf{lcf}(\mathcal{P})))$ which are consistent with the assignment to the envelopes in $\mathcal{E} = \textbf{envs-at}(\mathbf{lcf}(\mathcal{P}))$. It is then straightforward to compute the **Source-Sink** relation between envelopes in $\mathcal{E}$ and $\mathcal{E}'$.

Since our model construction proceeds in the direction of the flow of time, we wish to know where the assigned occupants of envelopes in last assignment of the partial model go to after the next ECF. The possible 'sinks' for these objects are: $\mathbf{sinks}(E) = \{\, E' \mid \textbf{Source-Sink}(E, E')\}$. To satisfy **C2** we only need to ensure that for every $E \in \mathcal{E}$, every label in $\mathbf{ass}(\mathcal{P}, E)$ is assigned to some envelope in $\mathbf{sinks}(E)$, and also at least one label in $\mathbf{ass}(\mathcal{P}, E)$ is assigned to each $E' \in \mathbf{sinks}(E)$. Moreover, it is easy to see that if the last assignment of $\mathcal{P}$ satisfies the exclusivity condition **C1** then any extension constructed in this way will also satisfy the **C1**.

To choose the most likely (partial) model according to the object recognition software, we compute a vote measure as follows:

$$\mathbf{vote}(\mathcal{P}) = \sum\{v_i \mid \langle E_i, A_i\rangle \in \mathbf{ass}(\mathcal{P}) \wedge \mathbf{vote}(A_i, E_i) = v_i\} \,.$$

We may have several different partial models that agree on their last assignment — i.e. **last-ass**$(\mathcal{P}_1) =$ **last-ass**$(\mathcal{P}_2)$. These, represent different assignment paths leading up to the same end state. Typically, one will have a higher vote support than the other, which gives an indication that one is the more likely of the two. In such a case we say that the more likely partial model *subsumes* the other:

$\mathbf{Subsumes}(\mathcal{P}_1, \mathcal{P}_2) \leftrightarrow$
$\quad((\textbf{last-ass}(\mathcal{P}_1) = \textbf{last-ass}(\mathcal{P}_2)) \wedge \mathbf{vote}(\mathcal{P}_1) > \mathbf{vote}(\mathcal{P}_2))$

Given a set $\mathfrak{M}$ of partial models, we can 'prune' it to retain only the 'best' models that lead up to any given final assignment.

$$\textbf{prune-subs}(\mathfrak{M}) = \{\mathcal{P} \in \mathfrak{M} \mid \neg\exists(\mathcal{P}' \in \mathfrak{M})[\mathbf{Subsumes}(\mathcal{P}', \mathcal{P})]\}$$

We initialise the set of partial models by considering all possible assignments of the domain objects to the initial envelopes, with the additional requirement that each envelope must be assigned a number of objects that is as least as many as the number of CLG-ST-boxes it contains. This initial partial model set will be denoted $\mathfrak{M}_0$. We then run the following algorithm, which iterates through successive ECFs to generate the set of all consistent models. We initially set $f := f_0$ and $\mathfrak{M} := \mathfrak{M}_0$; then run the following loop:

while $(f \neq \mathsf{end})$ {
  $\mathfrak{M}' := \emptyset$
  foreach $(\mathcal{P}_i \in \mathfrak{M})$ {
    $\mathfrak{M}' := \mathfrak{M}' \cup \mathbf{extensions}(\mathcal{P})$ }
  $\mathfrak{M} :=$ **prune-subs**$(\mathfrak{M}')$
  $f := \mathbf{necf}(f)$ }

After this procedure terminates, the optimal labelling is the members of $\mathfrak{M}$ with the maximal vote. This model assigns labels to envelopes rather than boxes. To get a box assignment we assign each label in each envelope to the CLG-ST-box for which gives it the highest vote summed over all its constituent boxes.

The number of ways of putting m objects into n boxes is exponential in m. Hence the complexity of our algorithm is exponential in the number of objects in the scenario. However, because of the pruning operation, there is a fixed limit on the number of models that need to be stored, whatever the length of the frame sequence. This means that the complexity is linear in the sequence length.

This algorithm has been implemented using the SICStus Prolog [14], which allows easy creation and manipulation of the data structures required and a natural coding of the continuity reasoning al-

gorithm. It currently runs in an 'off-line' mode — i.e. it processes a whole frame sequence and then produces a globally consistent assignment for the complete sequence. In future we plan to implemented an 'on-line' mode, which would continually output the best hypothesis for the current state of an ongoing video sequence. Since our present implementation processes the whole sequence in a time only slightly longer than its actual duration, on-line real-time processing is certainly possible on present day hardware.

5 EVALUATION AND CONCLUSION

The system was evaluated on approximately two and a half minutes of basketball video (figure 1). This scene involves four objects (three players and a ball), variable numbers of which may be in the scene at any one time. It contains much interaction and occlusion that a conventional object tracker would find hard to track. The sequence was tracked and classified at 25fps and the results fed into the reasoning engine. The system was applied to a sequence of 2200 frames (88 seconds real time) which generated all possible spatio-temporally consistent labellings.[3] The model with the highest overall score was compared to a hand-annotated labelling which gives the ground truth at every 10th frame (plus some extra frames added at particularly dynamic parts of the video). Over the length of the sequence a total of 612 tracked boxes were compared.

Comparing our algorithm's output with the raw classifier output is problematic, because the raw output just gives ranked probabilities of individual objects being present but does not determine the number of objects. For purposes of comparison we must treat this data as somehow identifying a definite set of labels. To do this we use a rough heuristic: we say that the assignment includes any label which has the highest (or joint highest) probability of all those listed in the output, and also any other label identified with probability higher than 0.5. Figures computed from the resulting box-label assignments are given in the "Raw + Heur" column of our tables of statistics.

Another way of interpreting the raw output is to assume that the occupancy of each tracked box is known by an independent oracle. The "Raw + OCC" column shows these figures. Here, when evaluating a box which we know (from ground-truth data) contains n objects, we take the n labels that are assigned the highest probabilities in the raw tracker/classifier output.[4] Although this occupancy information cannot easily be obtained in practise, the statistics derived under this assumption are useful for comparison. They show that the improvement gained by the reasoner goes well beyond that obtained by simply being able to determine occupancy.

Table 1 compares the accuracy of assignments obtained from the two interpretations of the classifier output with labellings given by the model generation algorithm. Figures are given for the object detection rate, the percentage of assigned labels that are correct and the assigned occupancy. The figure giving the percentage of boxes which were given a completely correct labelling is perhaps the most intuitive and best overall performance metric.

These figures show that the spatio-temporal consistency algorithm results in a significant improvement in identification accuracy. This is most significant in the case of multiply occupied boxes. Hence we divide the statistics into single and multiple box cases. Of the 612 boxes compared, 377 contained a single object and 235 contained multiple objects. The single occupancy figures are somewhat degenerate because both raw data interpretations almost invariably assign a single label to single occupancy boxes. But the reasoner sometimes assigns more than one label to a single occupancy box. The multiple box statistics give a more informative comparison. It will be seen that the exact match score for the spatio-temporal consistency algorithm is over 60%; whereas, even when given the ground occupancy,

[3] This took approximately 5 minutes on a 500 MHz Pentium III; so real time performance is certainly possible on currently existing hardware.

[4] The raw output may occasionally give fewer labels than there are objects in the box (because it discards labels below a minimal threshold of probability). In this case we just take all labels in the raw output.

All boxes	Raw + Heur	Raw + Occ	Reasoner
Objects detected	44.0%	61.6%	82.5%
Labels correct	64.9%	62.3%	82.6%
Box occupancy correct	61.6%	98.5%	83.5%
Box labels all correct	39.5%	44.6%	68.6%
Single occupancy boxes only			
Objects detected	64.2%	64.2%	84.1%
Labels correct	64.2%	64.2%	74.6%
Box labels all correct	64.2%	64.2%	73.7%
Multiple occupancy boxes only			
Objects detected	29.6%	59.8%	81.4%
Labels correct	66.1%	60.1%	89.8%
Box labels all correct	0%	13.2%	60.4%

Table 1. Accuracy statistics for box labellings.

the raw classifier output rarely gives a completely correct labelling. Without being given the occupancy our heuristic interpretation gave no completely correct assignment for any multiply occupied box.

Our results show that the use of a mechanism for reasoning about and enforcing spatio-temporal consistency can significantly enhance the performance of a statistically-based object tracking and classification system. We believe that this kind of combination of statistical and logical approaches may be fruitful in other areas of AI.

ACKNOWLEDGEMENTS

This work was funded by the European Union under the Cognitive Vision Project (Contract IST-2000-29375).

REFERENCES

[1] B. Bennett, A.G. Cohn, P. Torrini, S.M. Hazarika, 'Describing rigid body motions in a qualitative theory of spatial regions', in *Proceedings of AAAI-2000*, eds., H.A. Kautz and B. Porter, pp. 503–509, (2000).

[2] S. Dockstader and A. Tekalp, 'Multiple camera fusion for multi-object tracking', in *Proc. IEEE Workshop on Multi-Object Tracking*, (2001).

[3] A. Elgammal and L. Davis, 'Probabilistic framework for segmenting people under occlusion', in *Proc. International Conference on Computer Vision*, pp. 9–12, (2001).

[4] S. M. Hazarika and A. G Cohn, 'Abducing qualitative spatio-temporal histories from partial observations', in *Proceedings of the Eight Conference on Principles of Knowledge Representation and Reasoning (KR 2002)*, eds., D. Fensel, F. Guinchiglia, D. McGuinness, and Mary-Anne Williams, pp. 14–25. Morgan Kaufmann, (2002).

[5] M. Isard and A. Blake, 'Condensation-conditional density propagation for visual tracking', *Intl. Jour. of Computer Vision*, **29**, 5–28, (1998).

[6] D. Koller, J. Weber, and J. Malik, 'Robust multiple car tracking with occlusion reasoning', in *Proc. European Conference on Computer Vision*, pp. 189–196, (1994).

[7] A. Lipton, H. Fujiyoshi, and R. Patil, 'Moving target classification and tracking from real-time video', in *Proc. IEEE Workshop on Applications of Computer Vision*, pp. 8–14, (1998).

[8] D. Magee, 'Tracking multiple vehicles using foreground, background and motion models', in *Proc. ECCV Workshop on Statistical Methods in Video Processing*, pp. 7–12, (2002).

[9] D. Magee, 'A sequential scheduling approach to combining multiple object classifiers using cross-entropy', in *Proc. IAPR International Workshop on Multiple Classifier Systems*, pp. 135–145, (2003).

[10] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, 'Tracking groups of people', *Computer Vision and Image Understanding*, **80(1)**, 42–56, (2000).

[11] P. Muller, 'A qualitative theory of motion based on spatio-temporal primitives', in *Principles of Knowledge Representation and Reasoning: Proc. of the 6th International Conference (KR-98)*, eds., A.G. Cohn, L. Schubert, and S. Shapiro, pp. 131–141. Morgan Kaufman, (1998).

[12] J. Sherrah and S. Gong, 'Resolving visual uncertainty and occlusion through probabilistic reasoning', in *Proc. British Machine Vision Conference*, pp. 252–261, (2000).

[13] C. Stauffer and W. Grimson, 'Adaptive background mixture models for real-time tracking', in *Proc. Computer Vision and Pattern Recognition*, pp. 246–252, (1999).

[14] Swedish Institute of Computer Science. SICStus Prolog. `http://www.sics.se/sicstus/`.

[15] D. Yang, H. Gonzalez-Banos, and L. Guibas, 'Counting people in crowds with a real-time network of simple sensors', in *Proc. IEEE International Conference on Computer Vision*, pp. 122–129, (2003).