This paper is under review for *Image and Vision Computing*. It is an extended version of the paper *Using Spatio-Temporal Continuity Constraints to Enhance Visual Tracking of Moving Objects*, which has been accepted for presentation at the ECAI 2004 conference.

Enhanced tracking and recognition of moving objects by reasoning about spatio-temporal continuity

Brandon Bennett [a], Derek Magee [a], Anthony Cohn [a] and David Hogg [a]

[a] *School of Computing, University of Leeds, Leeds, LS2 9JT, UK*

Abstract

A framework for the logical and statistical analysis and annotation of dynamic scenes containing occlusion and other uncertainties is presented. This framework consists of three elements; an object tracker module, an object recognition/classification module and a logical consistency, ambiguity and error reasoning engine. The principle behind the object tracker and object recognition modules is to reduce error by increasing ambiguity (by merging objects in close proximity and presenting multiple hypotheses). The reasoning engine deals with error, ambiguity and occlusion in a unified framework to produce a hypothesis that satisfies fundamental constraints on the spatio-temporal continuity of objects. Our algorithm finds a globally consistent model of an extended video sequence that is maximally supported by a voting function based on the output of a statistical classifier. The system results in an annotation that is significantly more accurate than what would be obtained by frame-by-frame evaluation of the classifier output. The framework has been implemented and applied successfully to the analysis of team sports with a single camera.

Key words: Visual Surveillance, Temporal Reasoning, Ambiguity, Temporal Representations

1 Introduction

No computer vision algorithm for tracking or object recognition is perfect under real-world operating conditions. Object trackers have difficulty with complex occlusions (e.g. in crowded pedestrian scenes or on the sports field) and object recognition algorithms rarely give 100% accuracy, even on well posed data sets, let alone under unconstrained circumstances. This lack of reliability is one of the reasons for the slow commercial uptake of the complex visual surveillance systems presented at machine vision conferences. In this paper we propose a framework for taking the imperfect output of an object tracker and object 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 lower level tracking and recognition systems explicitly model the possibility of 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 that are consitent with the requirements of object continuity. The the final output is then a globally consistent spatio-temporal description of the scene which is maximally supported by probabalistic information given by classifier.

A number of researchers have attempted to deal with object occlusion (and the resultant tracking problems) by attempting to track through occlusion. This involves reasoning about object ordering along the camera optical axis either using ground plane information [1,2] or simply reasoning about relative spatial ordering [3]. Dynamic models such as the Kalman filter are often used to model the position of occluded objects [4,5], under the assumption of known dynamics (e.g. linear motion), when no visual information is available. Multiple cameras have also been used to bypass the occlusion problem [6], however this is not always possible or practicable.

Our approach to occlusion handling differs from this body of work and has more similarity with the methods of McKenna *et al.* [7] and Sherrah and Gong [8]. These works do not attempt to disambiguate occluding objects, but instead reason about the occlusion taking place. McKenna *et al.* track 'blobs' that may be groups or individuals. In their work it is initially assumed all objects are separate (an assumption we do not make); and when blobs merge the resultant blob is recorded as a 'group' made up of the contributing individuals. A dynamically updated model of global object colour is used to disambiguate objects at the point at which blobs split. This model is also used to reason about object occlusion within a group that makes up a single blob. This is useful when a split group consists of more than two individuals, however it relies on an assumption that no object is completely occluded during the split. Sherrah and Gong [8] present work in a highly constrained scenario where the head and hands of a single individual are tracked as blobs. The hands may occlude each other or the face (to form a single blob). A hand-built Bayesian network is used to perform frame-by-frame occlusion reasoning, based on available data (blob positions, velocities, number of blobs etc.). Perhaps the closest work to ours was presented recently by Yang *et al.* [9]. This system uses multiple cameras to provide a top view of the 'visual hull' of a crowd scene. Constraints on the number of pedestrians represented by each observed blob are determined according to the size of the blob's bounding box. These constraints are propagated from frame to frame to give an upper and lower limit on the number of objects present. All observed moving objects are assumed to be pedestrians, and no attempt is made to localise or identify individual pedestrians. Lipton *et al.* [10] present a system that uses simple object classification (pedestrian *vs.* car) to aid object tracking. Simple temporal consistency rules are used to prune transient objects resulting from noise. None of these systems performs more than frame-by-frame reasoning or allows for the possibility of error in the underlying low-level tracking and recognition algorithms. 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.

2 An Architecture for Tracking and Recognition with Error Correction based on Consistency Reasoning

Our proposed architecture consists of three parts: i) a modified 'blob tracker', ii) an object recognition/classification system, and iii) an error reasoning and consistency checking module. The relationship between these components is illustrated in figure 1.

The grey arrows in figure 1 represent the potential for feedback from the high-level reasoning system to the lower level modules. This feedback is not exploited in the system presented in this paper; however this is an active area of research. The individual elements of this architecture are described in the following sections.

3 The Blob Tracker

Much effort has been put into visual object tracking methods over the last two decades, and a number of methods have been developed that are robust under certain constraining conditions. These constraints are usually on the

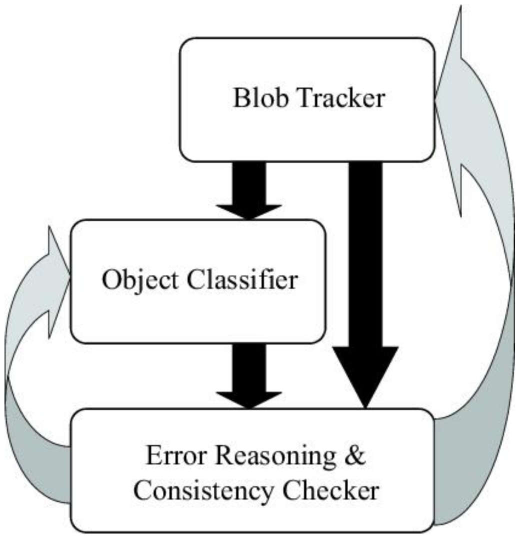

Fig. 1. An Architecture for Robust Scene Analysis

type of motion possible (e.g. Kalman filter based trackers such as [4] assume approximately constant velocity), on the total number of objects possible and on the types of multiple object occlusions possible. There is little evidence of any object tracking system that will work in completely unconstrained scenarios such as crowd scenes (e.g. in an crowded airport terminal), or for team sports surveillance (e.g. football, rugby or basketball). A popular solution to this is to present tracking results as a probability density over a space of possible model configurations, as for example the in CONDENSATION algorithm of Isard and Blake [2]. This works well for propagating short-term (frame to frame) configuration information; however these densities are usually represented as approximations to the true density, and long term configuration information is not preserved. This is compounded by the fact that many high level systems that use the output of such object trackers require as their input a single track result (for example [4,11]). In such cases the maximum a-priori probability configuration (or similar) is usually taken and the information present in the densities is lost (we do not encourage such an approach!).

In contrast, more traditional single hypothesis (e.g. Kalman filter based) object trackers are now becoming popular again for multiple object tracking due to their lower computational cost and demonstrable robustness (the system of Stauffer and Grimson [12] has been running on-line for a number of years with a very low error rate). These benefits are of course at the expense of detail in the tracker output (such as object identity or pose), however in many applications this detail is not required, or can be obtained from a separate module. The object tracker used in this paper is an extension of the car tracker of Magee [13], which in turn is based on the work of Stauffer and Grimson [12]. The principle behind these trackers is illustrated in figure 2.

Such an object tracker tracks coherent 'blobs' using a loose model of object

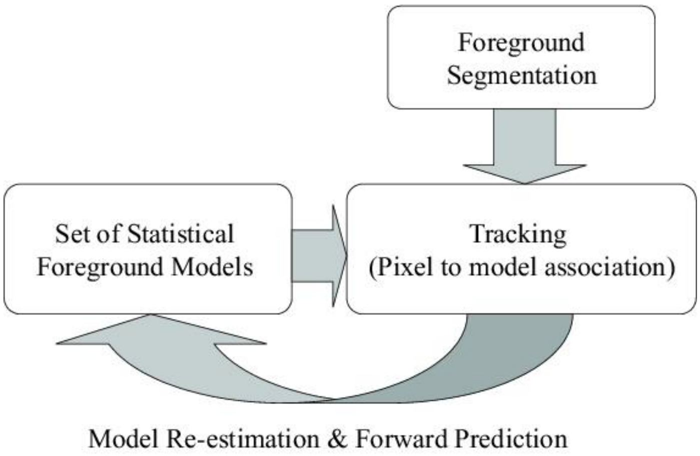

Fig. 2. Schematic of Blob Tracker Operation

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 for such a tracker (although this is usually the case). This is usually ignored as 'error'; however in our later analysis (see section 5) we explicitly do not assume this global one-to-one mapping. This allows us to explicitly 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 are a composite of two easily separable blobs). This technique reduces to near zero the occurrence of tracker errors such as lost/additional objects in our chosen example domain. The remaining 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 3.

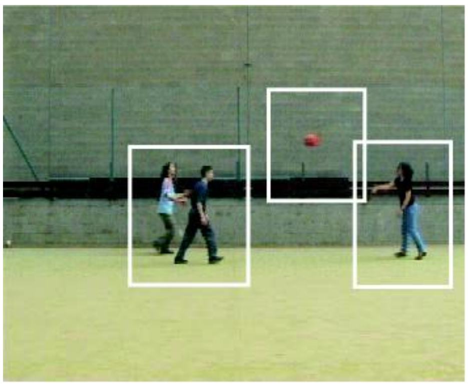

Fig. 3. Example Output from Blob Tracker

The boxes in figure 3 represent a visualisation of the statistical foreground models in figure 2 which includes the variance of blob pixel locations about the blob mean in 2 perpendicular directions (vertical and horizontal in this case). The box represents a threshold on these distributions that defines the area containing the blob (typically 3 standard deviations).

4 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 5) is designed to cope with imperfect and ambiguous input. To this end we have implemented a system that combines a number of simple exemplar based classifiers. The classification module uses the positional and foreground segmentation information provided by the tracker to form a feature description of each object tracked.

A colour histogram (YU colour representation) is formed for each object tracked, using the colour values of the foreground pixels identified as belonging to that object by the object tracker. A training video sequence is automatically tracked (using the object tracker), and the objects identified hand labelled with their identity (at approximately 1 second intervals). Around 50 examples of each object are used. The resultant set of histograms is clustered, using the K-means clustering algorithm, to give a set of exemplar histograms (corresponding to the cluster means). Each cluster is assigned an object label based on the identity of the majority of training examples in that cluster. The number of clusters is selected such that each cluster contains at least 99% examples of a single identity, and there is at least one cluster relating to each possible identity. These labelled exemplars are used (on unseen sequences) as an object classifying model. The advantage of the exemplar approach is that new objects/classes can be added to the system easily without a full re-training stage. This is near essential if training is to be performed on-line as new objects appear (although we currently work with a 'closed-world' offline scenario where all possible objects are known a-priori).

To apply our exemplar based model to an unseen tracked object a YU colour histogram is formed for the unseen object in exactly the same way as for the training examples. A model exemplar histogram represents the probability distribution over colour for a single pixel ($P(colour|model_n)$). A data histogram represents the normalised frequency distribution of colour values. Therefore, to calculate the probability of the novel data observed with respect to a single exemplar ($P(observation|model_n)$) it is simply a matter of representing the two histograms as matrices and calculating the correlation between these two matrices (equation 1).

$$P(observation|model_n) = \sum_{b=1}^{N_{bins}} D_b M_b \qquad (1)$$

Where:
D_b = Bin n of the data histogram

M_b = Bin n of the model histogram
N_{bins} = The number of bins in the histograms

Using Bayes law:

$$P(model_n|observation) = \frac{P(observation|model_n)P(model_n)}{P(observation)} \quad (2)$$

$$= P(model_n) * \sum_{b=1}^{N_{bins}} \frac{D_b M_b}{P_b} \quad (3)$$

Where:
P_b = Bin n of the mean histogram over all training examples
$P(model_n)$ = The a-priori probability of the exemplar [1]

To obtain the probability that an observation is of a particular identity, $P(identity_m|observation)$, the maximum value of $P(model_n|observation)$ over all exemplars relating to that identity is taken. Taking the maximum value is appropriate as the different exemplars relating to a partcular identity generally relate to different object configurations or viewpoints. As such, exemplars relating to object configurations or viewpoints other than the actual ones provide little information. These probabilities (along with the object locations and sizes from the object tracker) are passed on to the reasoning engine (described in section 5).

5 Ensuring Spatio-Temporal Continuity

The error reasoning and consistency checking module (figure 1) is designed to reduce error and ambiguity in the output of the lower level models by identifying a solution that both maximises statistical correlation with this output and is also globally consistent with respect to requirements of spatio-temporal continuity of objects. Specifically, 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).

[1] We use 1/no of exemplars, as each exemplar represents approximately the same number of training data items. However, this could be calculated as the normalised frequency distribution of cluster membership from the K-means clustering.

In the output given by any statistical classifier, it is quite possible that an object is detected to a high degree of probability in two locations that are widely separated. 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 chose 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 ensuring spatio-temporal constaints are respected, we also want to find an object labelling which is maximally supported by the frame-by-frame tracker output and the probabilistic output of the object recogniser for each tracked box. However, the recognition system was trained to identify single objects, whereas in tracking a dynamic scene there will often be several objects in a box. This means that there is no completely principled way to interpret the output figures from the recogniser. Nevertheless, it seems reasonable to assume that although there is a large amount of error and uncertainty in the low-level output, it does give a singnificant indication of what objects may be present. We shall explain below exactly how our system converts the low-level statistics into a metric of the likelihood of any given set of objects being in a box.

Local continuity information is provided by the low-level tracker module. The tracker output assigns to each blob's bounding box an identification tag (a number), which is maintained over successive frames. For newly split or merged boxes new tags are assigned but the tag of their parent box in the previous frame is also recorded. Thus each box is associated with a set of *child* boxes in the next frame. Conversely each box can be associated with a set of its *parent* boxes in the previous frame. The parent/child relation determines a directed graph structure over the set of boxes, which we call a 'box continuity graph'. Such a graph is illustrated in figure 4. Our algorithm depends on the structure of this graph, which will be examined in more detail later.

The continuity-based reasoning algorithm involves a somewhat complex restructuring of the available information. To describe its operation we start by formally specifying the information associated with a tracked box and the

[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.

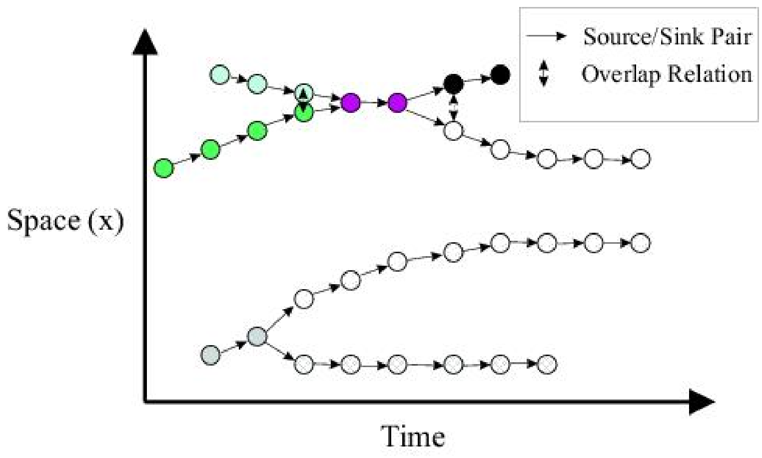

Fig. 4. Tracker output expressed as a 'box continuity' graph

output of the classifier when applied to that box. Formally, a tracker 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 elements of a continuous subset of the non-negative integers with the usual ordering. The initial frame will be denoted f_0. For convenience we introduce the 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/recogniser output will be denoted by BOXES.

From the point of view of object continuity, the parent and child boxes of a given box can be regarded respectively as 'sources' and 'sinks' of that box. The occupants of a box must have come from its parent boxes and must go to its the child boxes. Assuming no objects enter or leave a scene, each box is associated with at least one source box in the previous frame and one sink box in the next frame. However, 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. We do also allow object to enter or leave so there may be some boxes with no source or no sink. [3]

Within the box continuity graph are chains of boxes linked by a unique source-sink relationship. These are shown in figure 4 as linear sub-graphs shaded in a particular colour. Each of these corresponds to the 'history' of a tracker box between splitting and merger events. If the tracker were perfectly accurate, the

[3] Such boxes would typically occur at one or other side of the scene, but in the current implementation we have not attempted to enforce this constraint. A more sophisitcated model would explicitly model all the entry and exit points to a scene.

objects occupying any box would be constant along any of these linear sub-graphs. Hence, assuming such perfect accuracy one can enforce a localised continuity constraint by simply requiring that box labellings are consistent along such 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 $1 \leq i \leq n-1$ we have $\mathbf{ch}(b_i) = \{b_{i+1}\}$.

It is convenient to represent an ST-box as an ordered set S indexed by frame numbers in the range $\mathbf{s}(S) \ldots \mathbf{e}(S)$, where the functions $\mathbf{s}(S)$ and $\mathbf{e}(S)$ denote respectively the start and end frames of the period over which S exists. We also define the partial function $\mathbf{box\text{-}at}(f, S)$ to denote the box $b \in S$ whose frame number is f. This function is only defined for f in the range $\mathbf{s}(S) \leq f \leq \mathbf{e}(S)$. Clearly each box $b \in \mathsf{BOXES}$ is a member of a unique ST-box which we denote by $\mathbf{st}(b)$. The set $\mathsf{ST\text{-}BOXES} = \{S \mid (\exists b \in \mathsf{BOXES})[\mathbf{st}(b) = S]\ \}$ is the set of all ST-boxes.

5.1 Coarse Object Grouping with 'Envelopes'

Although the tracker output enables us to derive a graph representing temporal continuity between tracked boxes, this structure is only indirectly related to the trajectories of actual moving objects in the scene. There are several issues that complicate the relationship between tracker boxes and objects. 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 objects sometimes transfer between tracker boxes without themselves being independently tracked. This can occur because of occlusions among objects or because of limited resolution of the tracker (or a combination of the two). When a box containing more than one object is close to or ovelaps another box, an object from the multiply occupied box can easily transfer to the neighbouring box without being detected. Conversely undetected transfers are almost always between boxes whose geometries overlap.

A consideration of these problems led us to the idea that in order to get a more accurate identification of the locations and movements of closely grouped or occluding objects, we need to employ a representation in which the locations of objects are modelled at a coarser level than that of individual boxes. Hence, we introduced a higher level abstract data object that we call an *envelope*. Intuitively we can regard an envelope as a maximal clusters of overlapping

ST-boxes. The concept of an evelope is illustrated diagramatically in figure 5. Here the thicker 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. The dashed lines show the division of this structure into envelopes.

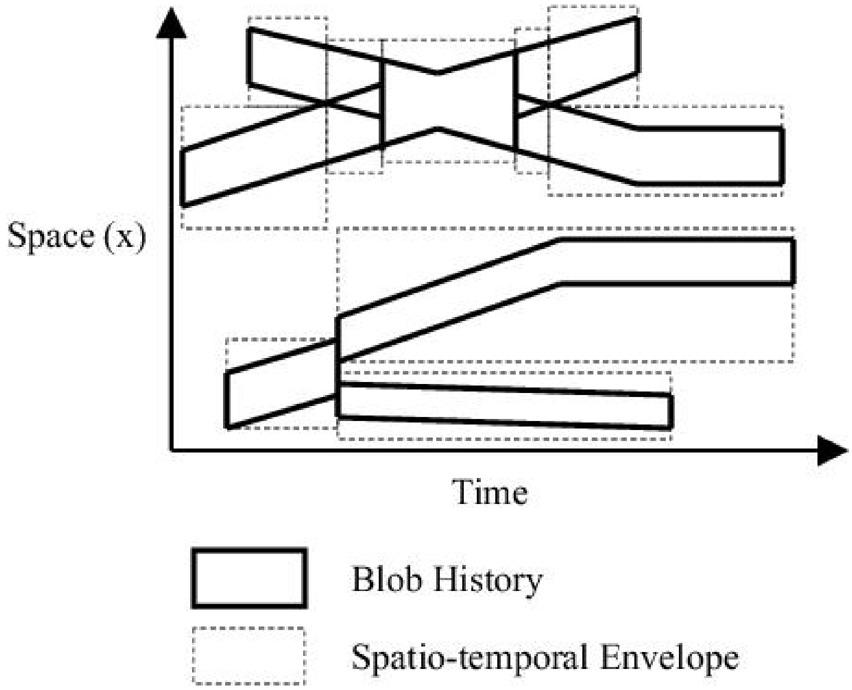

Fig. 5. Deriving Spatio-temporal Envelopes from Tracker Output

The definition of an envelope depends upon specifying an overlap relation between boxes, which we write formally as $\mathbf{Overlap}(b_1, b_2)$. Since, 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. Actually, because of the statistical nature of the bounding boxes, we may vary the scale of the boxes in order to yeild stricter or weaker overlap relations. In the current work we are concerned with demonstrating the general method by which enforcing continuity constraints improves tracking/recognition accuracy, so 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 characterise what constitutes an envelope we first introduce the concept of a *constant-local-group spatio-temporal box* or *CLG-ST-box*. This is a temporal sub-division of an ST-box, over which it is a member of a constant 'local group' of spatially overlapping ST-boxes. Thus it is part of the same cluster of ST-boxes throughout this period. Whereas the begining and end frames of ST-boxes are determined by merger and splitting, 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 5. The divisions coincide with the temporal boundaries of the envelopes.

Local groups are sets of ST-boxes that are related by the transitive closure of the **Overlap** relation. This relation is written as $\mathbf{Overlap}^*(b_1, b_2)$.[4] 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

$$\textbf{local-g}(f, S) = \{S' \mid \mathbf{Overlaps}^*(\textbf{box-at}(f, S), \textbf{box-at}(f, S'))\} \ .$$

A CLG-ST-box can now be defined as a subset of an ST-box corresponding to a maximal temporally continuous sequence of frames $F = [f_m, \ldots, f_n]$, such that for each $f_i \in F$, the function $\textbf{local-g}(f, S)$ takes a constant value. Each ST-box determines a unique CLG-ST-box for each frame over which it exists. Hence we can define the function

$$\begin{aligned}&\textbf{clg-st}(f, S) = \{b \in S \mid \textbf{local-g}(f(b), S) = \textbf{local-g}(f, S) \wedge \\ &\neg\exists f'[((f(b) < f' < f) \vee (f < f' < f(b))) \wedge \textbf{local-g}(f', S) \neq \textbf{local-g}(f, S)\} \ .\end{aligned}$$

The set of all CLG-ST-boxes is given by

$$\begin{aligned}\mathsf{CLG\text{-}ST\text{-}BOXES} = \{C \mid (\exists S \in \mathsf{ST\text{-}BOXES})(\exists f)[(\mathbf{b}(S) \leq f \leq \mathbf{e}(S)) \wedge \\ \textbf{clg-st}(f, S) = C\}\end{aligned}$$

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 the function $\textbf{local-g}(C)$, which does not require a frame argument:

$$\textbf{local-g}(C) = \textbf{local-g}(\mathbf{b}(C), \mathbf{st}(C)) \ .$$

We can 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

$$\mathbf{env}(C) = \{\textbf{clg-st}(\mathbf{b}(C), S) \mid S \in \textbf{local-g}(C)\} \ .$$

For an envelope E, the functions $\mathbf{b}(E)$ and $\mathbf{e}(E)$ denote the beginning and

[4] That is, $\mathbf{Overlap}^*(b_1, b_n)$ 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 $\mathbf{Overlap}(b_i, b_{i+1})$.

end frames of the envelope's existence. By definition, all its consituent CLG-ST-boxes have the same beginning and end frames.

Each box in the tracker input is related to a unique envelope given by:

$$\mathbf{env}(b) = \mathbf{env}(\mathbf{clg\text{-}st}(f(b), \mathbf{st}(b)))$$

And the set of envelopes derived from the tracker input is given by:

$$\mathsf{ENVELOPES} = \{E \mid (\exists b \in \mathsf{BOXES})[E = \mathbf{env}(b)]\}$$

Later we shall want to refer to the set of envelopes that exist at a given frame. This is given by:

$$\mathbf{envs\text{-}at}(f) = \{E \in \mathsf{ENVELOPES} \mid \mathbf{b}(E) \leq f \leq \mathbf{e}(E)\}$$

5.2 Enforcing Exclusivity and Continuity at the Envelope Level

Although envelopes give a coarser demarcation of object locations than do individual boxes, they provide a much more reliable basis for determining continuity. By definition, two different envelopes cannot spatially overlap (otherwise they would just be parts of a larger envelope). This means that there is an extremely low probability that an object can transfer between envelopes without being detected. Hence, our algorithm makes the assumption that the occupancy of an envelope is constant throughout its existence. The presence of object $\mathbf{l}$ in envelope E will be formally specified by the relation $\mathbf{Occ}(\mathbf{l}, E)$.

The exclusivity constraint **C1**, correspondes to the requirement that no object can occupy two distinct spatio-temporal envelopes that overlap in time.

$$\mathbf{C1)} \quad \forall E_1 E_2\, \mathbf{l}\, f\ [\ (\mathbf{b}(E_1) \leq f \leq \mathbf{e}(E_1)) \wedge (\mathbf{b}(E_2) \leq f \leq \mathbf{e}(E_2)) \wedge \mathbf{Occ}(\mathbf{l}, E_1) \wedge \mathbf{Occ}(\mathbf{l}, E_2)) \rightarrow (E_1 = E_2)\ ]$$

It will be seen in figure 5 that the set of envelopes has a continuity graph structure similar to that of boxes. In fact an *envelope continuity graph* can be formed direclty from the box continuity graph by collapsing all nodes derived from boxes in the same envelope into a single node.

Functions returning parent and child sets for envelopes can be derived directly from the corresponding relations between tracker boxes:

$$\mathbf{pa}(E) = \{E' \mid E' \neq E \wedge (\exists b b' \in \mathsf{BOXES})[(\mathbf{env}(b) = E) \wedge (\mathbf{env}(b') = E')(\wedge b' \in \mathbf{pa}(b))]\}$$

$$\mathbf{ch}(E) = \{E' \mid E' \neq E \wedge (\exists b b' \in \mathsf{BOXES})[(\mathbf{env}(b) = E) \wedge (\mathbf{env}(b') = E')(\wedge b' \in \mathbf{ch}(b))]\}$$

Because we allow objects to enter and leave the scene we also need to keep track of off-scene objects. We do this by introducing virtual, *off-scene envelopes* to our model. We could have different sets of off-scene envelopes for different entry/exit points but 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. The limitting frames of off-scene envelopes are defined as follows:

$$\begin{aligned}
\mathsf{B\text{-}OFFENV} = \{f \mid (\exists E \in \mathsf{ENVELOPES})[\ & ((\mathbf{pa}(E) = \emptyset) \wedge (f = \mathbf{b}(E))) \vee \\
& ((\mathbf{ch}(E) = \emptyset) \wedge (f = \mathbf{b}(E) + 1))\ ]\ \} \\
\mathsf{E\text{-}OFFENV} = \{f \mid (\exists E \in \mathsf{ENVELOPES})[\ & ((\mathbf{ch}(E) = \emptyset) \wedge (f = \mathbf{e}(E))) \vee \\
& ((\mathbf{pa}(E) = \emptyset) \wedge (f = \mathbf{b}(E) - 1))\ ]\ \}
\end{aligned}$$

So the set $\mathsf{OS\text{-}ENVELOPES}$ contains all beginning/end frame pairs that have no other beginning or end frames occuring between them:

$$\begin{aligned}
\mathsf{OS\text{-}ENVELOPES} = \{\langle f_b, f_e \rangle \mid\ & f_b \in \mathsf{B\text{-}OFFENV} \wedge f_e \in \mathsf{E\text{-}OFFENV} \wedge \\
& \neg\exists(f \in (\mathsf{B\text{-}OFFENV} \cup \mathsf{E\text{-}OFFENV})[(f > f_b) \wedge (f < f_e)]\ \}
\end{aligned}$$

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.

In order to define the conitnuity relation between envelopes, it is convenient to first define a successor relation $\mathbf{Suc}(E_1, E_2)$, which holds when the end of E_1 is at the frame immediately before the beginning of E_2:

$$\mathbf{Suc}(E_1, E_2)\ \equiv_{def}\ ((\mathbf{e}(E_1) + 1) = \mathbf{b}(E_2))$$

We can now define the relation $\mathbf{Source\text{-}Sink}(E_1, E_2)$, 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 and also for tansfers between the two kinds of envelope.

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

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

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

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.

5.3 Observational Likelihood of Box and Envelope Occupancy

There is a finite set of potential explanations for the output presented by the blob tracker and object classifier that are consistant with the continuity constraints described in the previous section. But the number of possible explanations is extremely large even in simple scenarios. A metric is required to rank these based on the symbolic and probabalistic output of these lower level processes.

As described above, the low-level classifier computes probabilites based on a Bayesian combination of learned binary colour-histogram classifiers. Applied to a box b the output 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. The set of all labels known to the classifier will be denoted LABELS.

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 cases where there is more than one object in a box. However, we assume that, even in multi-object cases, the figures give an measure, albeit approximate and uncertain, of the likelyihood of an object being in the box. Hence, we can 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 spatio-temporal continuity constraints operate at the level of envelopes rather than individual boxes. Thus, we need to convert the votes for box occupancy into votes for an object being in a given envelope. There are a number of ways this could be done, none of which is really statistically valid. The function we have chosen is as follows: for an envelope E and an object $\mathbf{l}$, we 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. Thus

$$\mathbf{vote}(\mathbf{l}, C) = \sum_{\mathbf{b}(C) \leq f \leq \mathbf{e}(C)} \mathbf{vote}(\mathbf{l}, \mathbf{box\text{-}at}(f, C))$$

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:

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

This would be reasonable on the assumption that the object stays in the same CLG-ST-box throughout its time within the envelope. This is not necessarily true, so the vote may be unreliable. Devising and evaluating more realistic voting functions is a subject of ongoing work. (Nevertheless, as will be seen below, our crude voting function is already a good enough metric to significantly enhance the reliability of recognition.)

In order to reduce the support value given to a label to the range $[0 \ldots 1]$, we compute the fractional vote for $\mathbf{l}$ with respect to the total votes of all objects in the classifier domain $\{\mathbf{l}_1, \ldots, \mathbf{l}_N\}$. This is given by:

$$\mathbf{frac\text{-}vote}(\mathbf{l}, E) = \frac{\mathbf{vote}(\mathbf{l}, E)}{\sum_{i=1 \ldots N} \mathbf{vote}(\mathbf{l}_i, E)}\ .$$

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. In many cases a lower limit on envelope occupancy will be ensured by the

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

where $\mathbf{dur}(E) = \mathbf{e}(E) - \mathbf{b}(E) + 1$ is the duration of the envelope in frames.

The multiplication by $\mathbf{dur}(E)$ weights each envelope's vote according to its duration. This (roughly) means that each frame will be given equal weight in determining the best solution.

For off screen envelopes, we take the vote for any set of labels to be 0.

The previous section defined a method for calculating a metric for ranking potential explanations of the output of the lower level systems. In principle we could use this to evaluate all possible spatio-temporally consistent sequences of object labels. However, this would be infeasible for all but the shortest and simplest sequences. Whenever a tracker box splits into two 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 (as well as being an exponential function of the number of objects involved). However, by taking a dynamic programming approach to the problem, the optimal solution can in fact be found by an algorithm whose complexity is linear in time. Thus, as long as the number of objects is relatively small, solutions for arbitrarily long sequences can be computed effectively.

An envelope labelling solution for a given tracker input assigns a set of labels to each envelope that satisfies the constraints **C1** and **C2**. Our algorithm first computes the set of envelopes and the **Source** relation from the complete set of boxes output by the tracker.

An *envelope change frame* (ECF) is an 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 shall build a model by starting at the initial frame of the tracker output and progressing through each successive ECF. Since envelope occupancy remains constant between ECFs, this model actually 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 input frame sequence we let $\mathbf{necf}(f) = \mathsf{end}$.

A spatio-temporally consistent assignment to all envelopes starting at or before some given ECF f will be called a *partial model* (up to f) and will be denoted $\mathcal{P}_i$ (where the optional i is a distinuishing 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 $A_i = \{\mathbf{l}_1, \ldots, \mathbf{l}_n\} \subseteq \mathsf{LABELS}$. The set must contain an assignment for all envelopes starting at or before some ECF. The function $\mathbf{ass}(\mathcal{P}, E)$ will denote the set of lables assigned by $\mathcal{P}$ to envelope E.

The requirement of spatio-temporal consistency of 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

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

In order to formalise the notion of one partial model's being an (immediate) extension of another, we define

$$\textbf{Extends}(\mathcal{P}', \mathcal{P})\ \equiv_{def}\ ((\mathcal{P}' \setminus \textbf{last-ass}(\mathcal{P}')) = \mathcal{P})$$

The set of all possible (spatio-temporally consistent) extensions of a partial model $\mathcal{P}$ is then given by

$$\textbf{extensions}(\mathcal{P}) = \{\mathcal{P}' \mid \textbf{Extends}(\mathcal{P}', \mathcal{P})\ \}\ .$$

extensions($\mathcal{P}$) is a key function of our algorithm. This function is straightforward to compute. 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 $\mathcal{E}' = \textbf{envs-at}(\textbf{necf}(\textbf{lcf}(\mathcal{P})))$ which are consistent with the assigmnet to the evelopes in $\mathcal{E} = \textbf{envs-at}(\textbf{lcf}(\mathcal{P}))$. Using the definitions given above, it is straightforward to compute the **Source** relation between envelopes in $\mathcal{E}$ and $\mathcal{E}'$.

Since our model construction proceedes in the direction of the flow of time, we whish to know where the assigned occupants of envelopes in last assignment of the partial model go to after the next ECF. The possible destinations are 'sinks' for these objects:

$$\textbf{sinks}(E) = \{\ E' \mid \textbf{Source-Sink}(E, E')\}$$

To satisfy **C2** we only need to ensure that for every $E \in \mathcal{E}$, every label in $\textbf{ass}(\mathcal{P}, E)$ is assigned to some envelope in $\textbf{sinks}(E)$, and also at least one label in $\textbf{ass}(\mathcal{P}, E)$ is assigned to each $E' \in \textbf{sinks}(E)$. Moreover, it is easy to see that if the last assignment of $\mathcal{P}$ satisfies the exclusivity condition **C1** then the assignment of any extension constructed in this way will also satisfy the **C1**.

In order to choose which are most likely according to the object recognition software, we compute a vote measure for each model. The total support for a partial model is just the sum of the support for each of its envelope assignments:

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

In all but the simplest cases, we may have several different partial models that agree on their last assignment — i.e. $\textbf{last-ass}(\mathcal{P}_1) = \textbf{last-ass}(\mathcal{P}_2)$. 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:

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

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.

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

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.

$f := f_0$
$\mathfrak{M} := \mathfrak{M}_0$
while $(f \neq \mathsf{end})$ {
 $\mathfrak{M}' := \emptyset$
 foreach $(\mathcal{P}_i \in \mathfrak{M})$ {
  $\mathfrak{M}' := \mathfrak{M}' \cup \textbf{extensions}(\mathcal{P})$ }
 $\mathfrak{M} := \textbf{prune-subs}(\mathfrak{M}')$
 $f := \textbf{necf}(f)$
}

We will not give a rigorous analysis of the complexity of our algorithm. However, we can establish by informal argument that the model building algorithm (and hence also the selection of the optimal labelling) is essentially linear in the length of the video sequence and exponential in the number of objects involved in the scene.

The key observation is that there are a finite number of ways that a finite set of objects can be distributed among a finite set of envelopes. This number corresponds to the number of partitions into disjoint non-empty subsets of a set of given finite cardinality. It can be computed by a recursive algorithm and is bounded by an exponential. For example, in our main test data we track a scene with four moving objects (which can move on and off the screen).

This means at any frame there are from 0 to 4 envelopes. Our pruning of low scoring models that are subsumed by better models with the same end state, means that number of partial models stored is limited to the number of distinct assignments that can be made to the end state of the sequence. For each of the cases of 0 to 4 envelopes, the number of assignments is respectively: 1, 14, 50, 60, 24. So the maximum number of models that must be stored is 60. [5]

At each cycle of the algorithm we extend each partial model to the next change frame. Again, although the number of possible extensions is exponential in the number of objects involved, for a fixed number of objects it is striclty bounded to a finite number of possibilities. Computing the new vote total for an extended envelope can be done in constant time since it depends only on the final assignment and the prior vote of the partial model being extended. Detecting the subsumptions among m newly extended models is $O(m^2)$; but is computationally trivial and here again m is bounded by a finite maximum depending on the number of objects in the scene.

These observations explain why for small numbers of objects, our algorithm performs very effectively. The crucial factor is our pruning policy means that the complexity becomes linear in the length of the frame sequence, even though we are finding the optimal of all possible models that are spatio-temporally consistent with that frame sequence. This pruning technique enables us to solve a highly intractable constraint problem by a method that can be seen as a kind of dynamic programming.

5.5 Implementation of the Continuity Reasoner

The continuity reasoner has been implemented using the Prolog language (SICStus Prolog [14]). This allows easy creation and manipulation of the data structures required and a natural coding of the continuity reasoning algorithm. 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. There is no reason why the algorithm could not be implemented in an 'on-line' mode, which would continually output the best hypothesis for the current state of an ongoing video sequence. And, since our present implementation processes the whole sequence in a time only slightly longer than its

[5] The actual numbers of partial models stored will typically be considerably fewer than these limiting numbers. This is because previous splitting and merger events in the history of the envelopes can lead to constraints on the minimum numbers of objects than can be present in some of the envelopes; so certain possibilities are ruled out. These cardinality constraints are automatically enforced by the model generation procedure.

actual duration, on-line processing in real time is certainly possible on present day hardware.

6 Evaluation

The system was evaluated on approximately two and a half minutes of the basketball scene illustrated in figure 3. This scene consists of four objects (three players and a ball), variable numbers of which may be in the scene at any one time. The movie contains much interaction and occlusion that a conventional object tracker would find hard to track with a single camera. The movie is 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) and took approximately 5 minutes to generate all possible spatio-temporally consistent labellings. [6] The model with the highest overal 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). Thus, over the length of the sequence a total of 612 tracked boxes were compared.

Comparing the output of the consistency reasoning algorithm with the raw output of object recogniser is somewhat problematic, because the raw output does not give any indication of the number of objects that are in each box. The statistics given are just ranked probabilites of individual object being present. However, for purposes of comparison we must treat this data as somehow identifying a definite set of labels. To do this we use a rather rough heuristic: we say that the assignment includes any label which has the highest (or joint highest) probability of all those listed in the ouput, 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 + Heuristic" column of our tables of statistics.

Another way of interpreting the raw output is to assume that the occupancy of each tracked box is somehow known by an independent procedure or oracle. The "Raw + OCC" column shows these figures. Here, when evaluating a box which we know (from human annotated ground-truth data) contains n objects, we take the n labels that are assigned the highest probabilites in the raw tracker/recogniser output. [7] Although there seems to be no obvious

[6] All experiments were carried out on a 500 MHz Penium III; so real time performance is certainly possible on currently existing hardware.

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

way this occupancy information could be obtained in practice, the statistics derived under this assumption may be useful for comparison. They show that the improvement gained by the reasoner goes well beyond that which could be obtained by simply being able to determine occupancy.

The table in figure 6 compares the accuracy of assignments obtained using the "Raw + Heuristic" and "Raw + Occ" interpretations of the tracker/recogniser output with the optimal spatio-temporally consistent box labellings given by the model generation algorithm. The "Objects detected" row gives the number of correct labels as a percentage of the total number of objects (averaged over all tracker boxes). The "Labels correct" gives the percentage of assigned labels that are correct. Notice that the detection rate for "Raw + Heuristic" is much lower than the percentage of correct labels. This is because for multiply occupied boxes it very often assigns fewer labels than the acual number of objects present. The third row shows the percentage with which boxes are assigned the correct number of objects. This figure is not very informative for "Raw + Heuristic" (which nearly always returns a single assignment) or for "Raw + Occ" (which nearly always gives the correct occupancy). However, it does show how good the spatio-temporal reasoner is as working out box occupancy. The final row gives the percentage of all compared boxes, where the label assignment exactly matched the ground truth data. This is perhaps the most intuitive and best overall performance metric.

	Raw + Heuristic	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%

Fig. 6. Accuracy statistics for all sampled boxes.

These figures show that use of the spatio-temporal consistency algorithm results in a significant improvement in the object recognition accuracy of the tracker. However, enhancement obtained by this method is fully effective in the case of multiply occupied boxes. Hence it is useful to divide up the satistics into single and multiple box cases. Of the 612 boxes compared, 377 contained a single object (i.e. a person or the ball) and 235 contained multiple objects. The single occupancy box statistics are as follows:

This table is somewhat degenerate. This is because both raw data interpretations almost invariably assign a single label to single occupancy boxes. The reasoner is considerably more accurate, although it sometimes assigns more than one label to a singel occupancy box.

	Raw + Heuristic	Raw + Occ	Reasoner
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%

Fig. 7. Accuracy statistics for boxes containing one object (in ground truth)

The multiple box statistcs give a much more informative comparison. It will be seen in particular that the exact match score for the spatio-temporal consistency algorithm is over 60%; whereas, even when magically given the ground occupancy, the raw output of the recogniser rarely gives a completely correct labelling. Without being given the occupancy our heuristic interpretation didn't give a single completely correct assignment for any multiply occupied box.

	Raw tracker	Raw + Occ	Reasoner
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%

Fig. 8. Accuracy statistics for boxes with more than one object (in ground truth)

7 Discussion and Future Work

The results presented in the previous section show the substantial improvement in labelling performance that can be achieved by applying our two simple spatio-temporal consistency rules and using voting over spatio-temporally equivalent regions. The framework presented in this paper is a simple, yet efficient, method of applying these constraints. The performance increase is especially evident in the labelling of multiple occupancy blobs. This is as to be expected as the clasifiers were only trained on well separated objects and there is often significant occlusion. The constraints detect the inconsistencies caused by such errors and so can fix qualitatively localised errors.

A possble extension to the system is to the use of multiple cameras. Each (rectangular) blob represents a quadrilateral area when projected to a groundplane. This area must contain the set of objects relating to the set of labels assigned to the blob. If there are multiple cameras the intersections of these quadrilaterals must contain the intersection of the two label sets. This is a highly efficient way of performing multiple camera integration and occlusion reasoning.

The underpinning theoretical framework could also be usefully generalised in various ways. In particular we would like to remove the assumption that we are dealing with a fixed finite number of lables each of which refers to a unique object. More generally we would like a system which handled classifiers that could apply to multiple objects. For example, to identify team membership in a sports game, or types of animal amongst a mixed herd. In such a setting we need to drop the 'exclusivity' constraint employed by our system; and so allow the same label to be simultaneously applied to several envelopes (in fact we would need also to allow multiple instances of a label to be associated with a single envelope). This would obviously weaken the constraints that our system uses and might make it less accurate. However, the continuity constraint would still be in force, and would still provide a significant mechanism for enhancing the labelling.

The restriction to the case of a fixed number of objects could also be removed by modification of our algorithm. Instead of considering all assignments from a fixed set of objects, we could initially construct only models with a minimal number of objects — i.e. one object in each initial envelope. If an an envelope splits, we will then be forced to recognise that it contains at least two objects. We then revise our models by considering all possible classifications of the new object, propagating these back along possible paths of the envelope sequence of the models and calculating new votes. This idea introduces some subtleties because to obtain an efficient algorithm we will also need to apply pruning to the back propagation. However, there do not seem to be any obstacles to achieving this.

A final and more radical enhancement would be to follow up the suggestion made in section 2, which was to implement some kind of feedback between the continuity reasoner and the tracking and classifying modules.

In the case of tracking, we have seen that certain tracking errors can be circumvented by the construction of spatio-temporally consistent models. For instance, we can enforce the requirement that objects cannot spontaneously appear or disappear. In the current implementation we just take the tracker's output and attempty to clean up any discontinuities; but it could clearly be useful to signal these back to the tracker so it could modify its processing. For example, the reasoner could flag a tracked box as being almost certainly a phantom, or it could help it keep tabs on objects that remain stationary for long periods of time.

It is less obvious how to achieving beedback between the consistency checking algorithm and the classifier; however, this may also be a very fruitful line of research. The observation that a certain combination visual properties forms a coherent entity that moves continuously in space is good evidence that those properties serve to identify a particular object. Thus, spatio-temporal continuity can be used as a criteron for validating a classifier: good object classifiers ought to pick out spatio-temporally continuous regions in a video sequence. Hence, one can envisage that spatio-temporal reasoning may play a very important role in the automated learing of object classifiers.

8 Acknowledgements

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

References

[1] D. Koller, J. Weber, J. Malik, Robust multiple car tracking with occlusion reaasoning, in: Proc. European Conference on Computer Vision, 1994, pp. 189–196.

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

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

[4] P.Remagnino, A. Baumberg, T. Grove, D. Hogg, T. Tan, A. Worrall, K. Baker, An integrated traffic and pedestrian model-based vision system, in: Proc. British Machine Vision Conference, 1997, pp. 380–389.

[5] R. Rosales, S. Sclaroff, Improved tracking of multiple humans with trajectory prediction and occlusion modeling, in: Proc. IEEE Workshop on the Interpretation of Visual Motion, 1998.

[6] S. Dockstader, A. Tekalp, Multiple camera fusion for multi-object tracking, in: Proc. IEEE Workshop on Multi-Object Tracking, 2001.

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

[8] J. Sherrah, S. Gong, Resolving visual uncertainty and occlusion through probabilistic reasoning, in: Proc. British Machine Vision Conference, 2000, pp. 252–261.

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

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

[11] R. Howarth, H. Buxton, Conceptual descriptions from monitoring and watching image sequences, Image and Vision Computing 18 (2000) 105–135.

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

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

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