Detection and Classification of Multiple Person Interaction

Scott Blunsden

Doctor of Philosophy

Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh

2009

(Graduation date: 2009)

Abstract

This thesis investigates the classification of the behaviour of multiple persons when viewed from a video camera.Work upon a constrained case of multiple person interaction in the form of team games is investigated. A comparison between attempting to model individual features using a (hierarchical dynamic model) and modelling the team as a whole (using a support vector machine) is given. It is shown that for team games such as handball it is preferable to model the whole team. In such instances correct classification performance of over 80% are attained. A more general case of interaction is then considered. Classification of interacting people in a surveillance situation over several datasets is then investigated. We introduce a new feature set and compare several methods with the previous best published method (Oliver 2000) and demonstrate an improvement in performance. Classification rates of over 95% on real video data sequences are demonstrated. An investigation into how the length of time a sequence is observed is then performed. This results in an improved classifier (of over 2%) which uses a class dependent window size. The question of detecting pre/post and actual fighting situations is then addressed. A hierarchical AdaBoost classifier is used to demonstrate the ability to classify such situations. It is demonstrated that such an approach can classify 91% of fighting situations correctly.

Acknowledgements

I wish to thank my supervisor Robert (Bob) Fisher for his endless the support, patience and guidance throughout the course of my PhD. I also would like to thank my fellow students and researchers particularly those in the vision lab at Edinburgh. Their discussions and influence made the whole experience engaging and much more bearable. In particular I would like to thank; Ernesto Andrade, Rowland Sillito, Toby Collins, Toby Breckon, Mohammed Bennamoun, Lily Xiang Li and Tim Lukins. I am grateful to all the wonderful friends I have made whilst studying in Edinburgh. Thanks must also go to Naomi who kept me going when things got tough and brightened every day. Finally I would like to thank my family. Their support and encouragement made this thesis possible.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Scott Blunsden*)

To Mum, Dad, Caroline, Layla and Naomi.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Within this thesis an unanswered research question within the field of machine vision is identified and an approach to address the problem is suggested. Specifically the open question of how to determine and classify how people are interacting within a video sequence is addressed. This chapter gives an introduction to the problem.

The multitude of CCTV cameras available generates a huge amount of information for anything but an infeasible number of human operators to monitor. Most of the information coming from such cameras depicts mundane everyday activities and situations which do not display interesting characteristics from the point of view of human operators. Within this vast amount of information there are rare situations which are of interest. An example of such a situation may be an attempt to break into a building or other such criminal activity. Many situations such as these are inevitably missed as there are simply not enough operators to interpret all of the information generated from the cameras. Those few frames where something of interest is happening are lost in the hundreds of thousands of frames where nothing abnormal is occurring. In such cases it is not uncommon for incidents to be reported and then a recording of the incident found based upon the report.

A role of computer vision within this context is to provide an automated interpretation of video footage, thus presenting camera operators situations which are deemed to be of interest in some way. This process remains an attractive proposition as there are just not enough operators to manually interpret all the video information arriving. Even if such a situation existed it is likely that manually interpreting many hours of mundane footage would introduce mistakes into the process due to the sheer volume of information. One role of computer vision in surveillance applications is therefore to provide this automated interpretation of large voluminous data and edit out those

sequences which it deems as being uninteresting. There are many possible ways in which a scene could be termed interesting [Dee and Hogg, 2004b], within this thesis the concentration will be upon human activity and how people interact.

1.1 Motivation

Previous work has established that most humans are very good at picking up both the subtle nuances and more dramatic aspects of behaviour and using them to interpret what a person is feeling or what their immediate intentions are [Baldwin and Baird, 2001]. The dynamics of group behaviour are also similarly interpreted by people. For example, one can immediately tell the difference between a group greeting one another and the more likely scenario of simply passing by one another. Likewise, the ability to spot when an individual is acting suspiciously is a skill that many security staff possess. It has been shown by Troscianko et al. [Troscianko et al., 2004] that it is possible for both expert and non expert humans to distinguish between criminal and non-criminal group behaviour when observing CCTV footage over a variety of situations prior to the event occurring.

Within this thesis we investigate the identification and classification of multiple person interactions. This thesis seeks to investigate ways in which multiple person interactions can be identified within a video sequence. The aim is to automatically classify interacting behaviour through a range of scenarios and situations. The overarching goal of this thesis is to investigate whether it is possible to classify multiply interacting people in a range of situations as observed from a video camera.

This is currently an unresolved problem in computer vision although there has been some previous investigation of the issue [Oliver, 2000, Hongeng and Nevatia, 2001]. We will seek to expand and improve upon previous work. A more comprehensive review of previous work is presented in chapter 2.

Understanding of interactions involving many people will help in the automatic interpretation of video sequences. By interpreting situations involving multiple people it would be possible to automatically get a computer system to identify specific or interesting situations. Potentially dangerous or violent situations, such as fighting, could then be automatically detected. This would have significant benefits for surveillance applications.

1.2 Thesis Structure

First a literature review is presented in chapter 2. This review details previous work within the field along with an identification of gaps within the current work. Chapter 3 then presents work upon team game interactions. Team game interactions provide a simpler and more constrained problem with which we start to investigate classification of interacting individuals. The chapter presents a simpler way to identify team interactions than previously suggested in the literature.

Chapter 4 generalises the approach to a surveillance situation. Here the role of time is investigated as we show results when varying the amount of frames taken into consideration before classifying a sample. A new feature set is presented along with a new proposed model. We demonstrate its superior performance over a number of datasests. Finally the ability of a computer to detect pre and post fighting situations is investigated in chapter 5. Results are then presented along with generalisation to cases where there may be many classes of interest.

1.3 Research area and Contribution

This thesis shows it is possible to classify multiple person interactions correctly more than 85% of the time and to do it better than the previous state of the art method as demonstrated in [Oliver et al., 2000a]. We also show that it is possible for a computer to identify if fighting behaviour is about to occur.

The main contributions of this thesis are:

- Comparison and demonstration of a simpler and more accurate model for learning team activities. The presented model requires no pre-defined template and is quick to compute. It is also more accurate than the previously suggested model.
- Significant improvement in classification performance between interacting persons for specific interactions when compared to the previous best method.
- Demonstration and quantification of the performance effects of frame length in classification accuracy.
- Investigation and demonstration of detection and classification of fighting situations including pre-fighting situations.
- Demonstration of a scalable hierarchical classifier for detecting fighting behaviour.

- Large scale classification on publicly available data thus enabling others to make meaningful comparisons
- The creation of the BEHAVE dataset [Blunsden et al., 2007b]. This is a large publicly available and annotated dataset.

Chapter 2

Previous Work

The main aim of this chapter is to show previous work within the domain of human activity recognition. This thesis seeks to demonstrate how one can identify and classify human interaction. Within this chapter previous approaches to the problem of human behaviour recognition are reviewed. The review is used to identify gaps in current research which subsequent chapters address.

This review addresses the areas which are relevant for the work presented in this thesis and behaviour recognition as a whole. First previous work for representation of moving people is given in section 2.1. The issue of modeling a persons behaviour is looked at in section 2.2. Specific work related to modelling groups of people is presented in section 2.3. Finally a section on tracking technology is included (section 2.4) to show that the methods presented both in the review and the main thesis are possible with current state of the art tracking.

2.1 Representation of Moving People

This section reviews current representation technology. For the purposes of the research proposed here the interest lies in determining how a person or object of interest is to be represented. This is important as any interpretation of human behaviour (within the context of vision) will actually be an interpretation of the representation of that person.

2.1.1 Human Shape Representation

Blob Models

Blob models have been used in many previous applications [Pentland, 1976, Oliver et al., 1997, Bobick and Bolles, 1992, Kauth et al., 1995] of modeling features within an image. The blob typically refers to the shape of the foreground features which are identified by background removal or direct estimation of foreground features. Blobs are typically modeled as large areas of connected foreground components. The features of these regions are extracted and used to form an estimate of the probability density function (PDF) of the colour [Oliver et al., 2000a, Wren et al., 1997b], or intensity [Caporossi et al., 2004] of the foreground region. In Oliver *et al.* [Oliver et al., 2000a] RGB colour space is used to form a Gaussian PDF of the colour distribution of the blob (as do [Kang and Cho, 2004, Traver et al., 2004]), others such as the Pfinder system [Wren et al., 1997b] use YUV colour space and Caporossi *et al.* [Caporossi et al., 2004] use chrominance information derived from RGB colour.

Contour Models

A widely used approach for identification of humans and also more general objects is through the use of active contours [Kass et al., 1988] and active shape models [Cootes et al., 1994, 1992b,a]. Lefevre and Vincent [Lefevre and Vincent, 2004] used a snake model for the identification of players within a football game. A more advanced version of using such snake like models is presented by Baumberg [Baumberg, 1995, Baumberg and Hogg, 1994b] who developed a tracker using active shape models whereby the point distribution model of a person is automatically learned from training data. Magee also demonstrated the use of a shape model for the tracking of livestock [Magee, 2000] for the purposes of gait analysis.

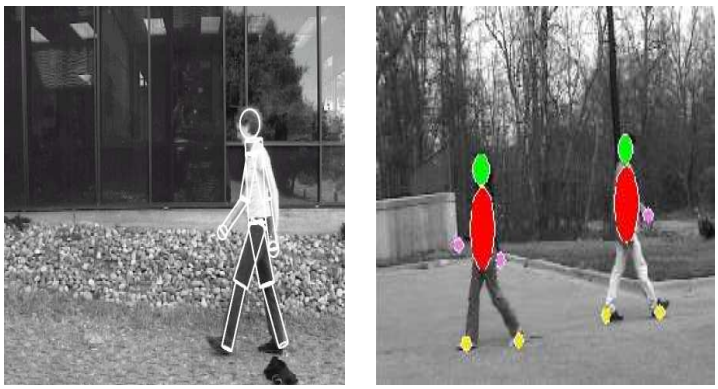
Recently the use of active appearance models (AAM) which include information about intensity information of the area enclosed by the model have been used for real time tracking. Stegman [Stegmann, 2001] tracks a range of objects using AAM based methods.

Part Models

A still richer description of the human form is given by those models which seek to localise individual parts of a person . The W4 system attempts to label parts of the

body using silhouette information [Haritaoglu et al., 2000]. Results using a simple planar body model [Haritaoglu et al., 1998] where the human form is approximated by a set of connected rectangles [Ju et al., 1996] are also presented.

More complex limb models are proposed by Sidenbladh and Black [Gavrila and Davis, 1996, Kakadiaris and Metaxas, 1996], where deformable limb models are proposed, see Figure 2.1. When building such 3D models the effects of perspective projection must be taken into account [Wachter and Nagel, 1999, Yamamoto, 1998], by doing so a more accurate mapping of the model to the data is achievable. Temporal aspects of human motion have also been used to improve the model fit to the data [Deutscher et al., 1999, Pavlovic et al., 1999, Leventon and Freeman, 1998, Sidenbladh et al., 2000] by using probabilistic or inverse kinematic models as the person moves throughout the scene. In [Leventon and Freeman, 1998] and [Brand, 1999] a 3D model is estimated from a 2D stick figure which has been extracted from the image sequence, motion was the main cue for its construction. In [Efros et al., 2003] a 2D stick figure is again fitted to the image data, however it is estimated using a similarity metric to previous examples rather than using geometric estimates.



(b) (c)

Figure 2.1: Some examples of geometric models fitted to the human form, (a) is adopted from from Sidenbladh and Black [Gavrila and Davis, 1996, Kakadiaris and Metaxas, 1996] where a fully 3D model is used and fitted to the human body. (c) is from the W4 system where a simpler approximate cardboard model is used. Only the main parts of the human are labeled using ellipses to approximate the components of a human figure

Many of these methods give a rich description of the human body which has obvious benefits for modeling the behavioral aspects of human motion. Despite the benefits there are inherent problems with trying to creating 3D models from 2D data. Such problems include but are not limited to occlusion, depth perception and kinematic sin-

gularities [Morris and Rehg, 1998]. To counter these problems steps such as stereo cameras [Ju et al., 1996], simplified backgrounds and specific clothing have been used [Gavrila and Davis, 1996, Kakadiaris and Metaxas, 1996] with some success.

Learned Feature Models

Human representations have been suggested which lie somewhere between the blob feature (see section 2.1.1) and a well defined and structured representation such as that used by the part and contour models (sections 2.1.1 and 2.1.1). A collection of feature points were proposed for tracking by Tomasi and Kanade [Tomasi and Kanade, 1991] and have been subsequently used by Hannuna [Hannuna et al., 2005] for the representation and recognition of quadruped motion. Viola and Jones [Viola and Jones, 2001] also use a point model but instead of all points being calculated based on the same feature (as in [Tomasi and Kanade, 1991]) a bank of Haar like features at various scales and configurations are used (see figure 2.21(a)). The features and positions are selected via AdaBoost [Schapire, 2002] and classified using a cascaded classifier. This method was extended to moving pedestrians where the final representation consisted of several Haar features applied to offset images of consecutive frames. Such a representation consists of many filter responses localised in a specific place in a image window. Learning features was also the focus of Papageorgiou *et al.* [Papageorgiou et al., 1998].

Another approach using function responses have been used particularly by [Felzenszwalb, 2001] who coupled PAC learning [Kearns and Vazirani, 1994] with edge responses. The Hausdorff distance (equation 2.1) metric was used to compare lines within the selected image window. Here A and B are two finite point sets.

$$h(A,B) = \max_{a \in A} \min_{b \in B} ||a - b|| \tag{2.1}$$

Image edges are also used by Wu and Nevatia [Wu and Nevatia, 2007] who use edge segments quantised by orientation (see figure 2.22) to represent small segments of a human figure. The model is then built up into parts denoting the torso, legs and head and shoulders of the person as well as considering the whole. Using structured models was also demonstrated by Lee and Nevatia [Lee and Nevatia, 2006]. Part based detection was investigated by Mikolajczyk *et al.*.[Mikolajczyk et al., 2004] whose work is similar to that of Viola and Jones as they too use a collection of outputs from filters and use many weak classifiers to produce a strong classifier to detect a human presence.

Mohan [Mohan et al., 2001] has also proposed a method of detecting objects based on learned components.

Crowded scenes have provided an interesting problem in which to apply part based models. Many of these scenes contain a high amount of occlusion and so do not provide enough of the information which can be required by global models. Leibe *et al.* [Leibe et al., 2005] use a sample patch code book with a Hough matching procedure to generate an initial hypothesis of the person. Once these matches have been established all features are used in a global shape model whereby chamfer matching [Barrow et al., 1977] is applied to the object as a whole.

2.1.1.1 Summary

Representation of the human form has been approached from two main angles, the first is to derive some statistical measure of a region which is thought to represent an object of interest. The other main approach is to try and fit a generic model to the image data. Active appearance model based techniques fall somewhere between the two approaches. Within the context of our specific project we decided that statistical based descriptions are more appropriate due to the range of situations and variable quality of the images. Such situations where there are multiple occlusions and poor quality of video would make 3D model fitting problematic and most likely impossible in the general case at present.

2.1.2 Human Motion Representation

Here the representation of motion is considered. Many approaches use the representations given in the previous section to form the basis of motion models.

One simple but powerful representation of objects is through the use of moments [Hu, 1962]. These were employed by Shutler [Shutler and Nixon, 2001] who used a temporal extension to Zernike moments [Teague, 1979] as a compact representation of human movement, which is invariant to rotation and to an extent scaling. Whilst having such an invariant representation seems appealing due to its obvious generalisation capabilities it is hard to tell what such a representation actually means and a less abstract representation is often favoured. Hoey and Little [Hoey and Little, 2000] also use a Zernike moment representation to represent flow information when classifying facial expressions.

In Davis and Bobick [Davis and Bobick, 1997] a more direct approach was taken.

They defined a motion energy image, which represents a localised measure of how the object's intensity distribution changes through time, as shown in figure 2.2. Within the W4 system [Haritaoglu et al., 2000] a similar idea (referred to as temporal textures) is employed to represent movement through time. This idea is the same as Davis and Bobick except it is extended to cope with objects moving throughout the scene by aligning the centre of the bounding boxes throughout the sequence.

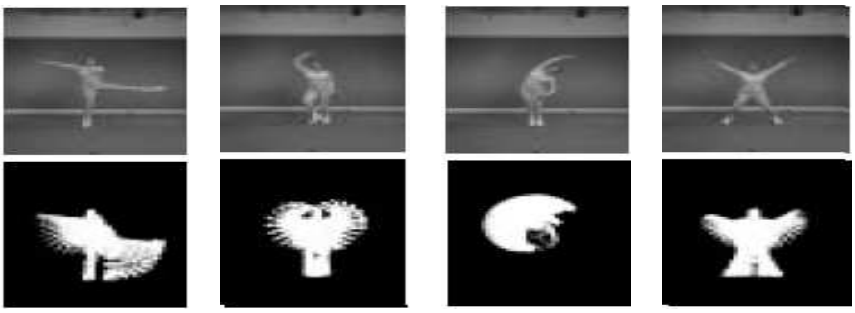


Figure 2.2: Motion energy images acquired from an exercise routine, adapted from [Davis and Bobick, 1997]

Senior [Senior, 2002] extends this approach by using the full RGB colour space and also defines a probabilistic framework for correspondence matching between frames. The other major novelty in Senior's approach is through the incorporation of occlusions into the modeling stage. If a model is partially occluded then such regions are not included in the matching process (defined by a likelihood of occlusion measure). In this way one is not trying to match a person with some statically or dynamically occluding object, which would lead to a poor overall model likelihood.

(a) (b)

Figure 2.3: (a) Human movement prototypes from which the optical flow representations (b) are calculated using motion pairs.

The use of image information in this way has also been proposed by Efros *et al.* [Efros et al., 2003] who stored a sequence of prototype human models based upon optical flow (figure 2.3).

Similar work has been produced by Yilmaz *et al.* [Yilmaz and Shah, 2005] who used a spatio temporal volume of extracted outlines of silhouettes from people. The Gaussian and the mean curvature are used to calculate points or interest in the volume. These points are then used for classification. Although the representation is robust to 3D rotations the creation of the volume itself is heavily view dependent due to its dependency upon silhouettes for creation of the volume. Scovanne [Scovanner and Ali, 2007] also investigates using space time volumes but use a SIFT[Lowe, 2004] like feature description applied to 3 dimensions. Blank et al. [Blank et al., 2005] follow a similar path by using the Poisson equation to extract space-time features such as local space-time salience, action dynamics, shape structure and orientation. Such features are used with spectral clustering [Zelnik-Manor and Perona, 2004] to classify types of action. All of the space time approaches use a similar silhouette extraction method for construction of the space time volume. Roh [Roh et al., 2008] also take a similar approach to gesture spotting by using silhouettes for spotting gestures in the domain of sports.

Another approach utilising a non silhouetted space time volume and optical flow was proposed by Ke *et al.* [Ke et al., 2005] who propose using optical flow taken over a small window size for a fixed frame length to represent actions. Classification is performed using a feature selection algorithm as proposed by Wu *et al.* [Wu et al., 2002]. The use of optical flow in [Ke et al., 2005] was shown to be superior to using intensity features such as those introduced by Laptev and Lindeberg [Laptev and Lindeberg, 2002] within the domain of action recognition. A diagram illustrating their approach is given in figure 2.4.

Niebles *et al.* Niebles et al. [2006] also propose using localised feature responses. These features are shown in equation (2.2)

$$R = (I * g * h_{ev})^2 + (I * g * h_{od})^2 \tag{2.2}$$

where $g(x;y;s)$ is the 2D Gaussian smoothing kernel, applied only along the spatial dimensions (x,y), and h_{ev} and h_{od} are a quadrature pair of 1D Gabor filters applied temporally, which are defined as $h_{ev}(t;\tau,\omega) = -cos(2\pi t\omega)e^{-t^2/\tau^2}$ and $h_{od}(t;\tau;\omega) = -sin(2\pi t\omega)e^{-t^2/t^2}$. The parameter s corresponds to the spatial scale whilst t corresponds temporal scale of the detector. These detectors are used to form a code book

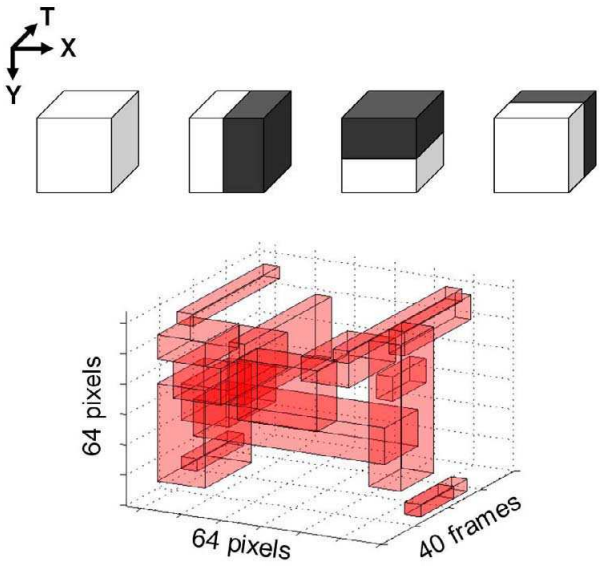


Figure 2.4: The top row illustrates the 3D volumetric features used in the classifiers. The first feature calculates the volume. The other three features calculate volumetric differences in X, Y, and time. The bottom row shows multiple features learned by the classifier to recognize the hand wave action in a detection volume. The volume only represents a small part of the entire image. Adapted from [Ke et al., 2005].

from which each sequence is labeled. This labeling is used with a pLSA [Hofmann, 1999] classifier on several datasets. The method shows good results compared to previous approaches from Dollar *et al.*[Dollar et al., 2005] and Ke *et al.* [Ke et al., 2005].

2.1.2.1 Summary

Motion representations try to describe how something is changing with respect to time. Such representations are usually coarse and attempt to describe the directionality and locality of the moving component rather than accurately preserving the underlying model.

The approaches reviewed so far are required to have information from either various scales [Ke et al., 2005][Hofmann, 1999][Davis and Bobick, 1997] or exhaustive viewpoints [Yilmaz and Shah, 2005, Efros et al., 2003, Intille and Bobick, 2001] in order to be applicable in the general case. Although it is claimed that some models can work for for multiple viewpoints [Yilmaz and Shah, 2005] this is typically only true of the model itself (ie the 3D volume) whereas differing camera viewpoints will cause very different models to be constructed. Several authours have attempted to create invariant representation [Gritai et al., 2004], [Rao et al., 2003] (however the

views displayed a high degree of similarity to one another) [Rao et al., 2003], or model complexity was increased significantly [Sidenbladh et al., 2000]. Such a restrictions are perhaps not as as limiting as first imagined as many surveillance camera positions provide similar viewpoints.

Motion is important in establishing what a person is doing, for example fast motions may be unusual or characterise a certain type of behaviour. However it should be noted that many of the methods are not translation independent. In order to be able to recognise and classify action from differing viewpoints it will be necessary to capture data from such viewpoints in order to model them accurately.

2.1.3 Scene Based Motion Representation

Within this section scene understanding is treated as the semantic interpretation of image sequences based upon knowledge (either learned or imposed) of a scene. Typically there is a learning phase which consists of automatically constructing a semantic model of the scene. This model represents typical behaviour within the current scene.

One such model is that of Makris and Ellis[Makris and Ellis, 2002] who track pedestrians as they move throughout the scene. These tracks are used to build probabilistic spline representations which capture common pathways throughout the scene, along with common variations of the pathways. Nair and Clark [Nair and Clark, 2002] also represent common trajectories throughout the scene, however they use a HMM representation. Again trajectory information plays a key role in Morris and Hogg [Morris and Hogg, 2000] who used trajectory information to classify unusual behaviour. However this time the trajectory information is defined in relation to a set of landmark points rather than an absolute position.

A similar approach is taken within the domain of traffic modeling [Grimson et al., 1998, Galata et al., 2002, Magee, 2004, Ivanov and Bobick] where common traffic flow throughout the scene is learned. Significant deviations from the common flows are classified as representing anomalous behaviour.

An alternative approach is to discover those parts of the scene which display some regularity with respect to how it is used and to concentrate upon such areas. Needham [Needham, 2003] builds a scene model for a football game which represents the most common areas on the playing field which players inhabit. Common positions and directionality of players within a game is learned. Brand and Kettnaker [Brand and Kettnaker, 2000] take a similar localisation approach, where regions within the scene

(a) (b) (c)

Figure 2.5: (a) Shows two learned route models superimposed upon the scene. The central line of the trajectory is the expected route, the surrounding lines are the expected variance from the route. (b) and (c) show evaluated route which an individual has taken throughout the scene. (b) shows a route which is evaluated as typical, where as (c) shows a route which is classified as being atypical. The red circle illustrates where the route differs significantly from those captured by the model. Adapted from [Makris and Ellis, 2002]

are learned along with typical trajectories through that region. To classify movement regions are modeled using a modified entropic HMM [Dietterich, 2002].

In Dee and Hogg [Dee and Hogg, 2004a,b] a measure of "interestingness" is proposed. Within this paper the validity of defining *atypical* behaviour as *interesting* behaviour, is questioned. In this case interest is defined as significant deviation from a goal or sub goal. These goals and sub goals characterise what could be expected as a person moves throughout the scene rather than just representing what people typically do.

2.1.3.1 Summary

By representing the scene in this way these models are only valid for the current scene. If the cameras attention is moved elsewhere then the scene semantics must be re-learned. This is of course possible but it will take time to re-learn the scene semantics and so they are not applicable if the camera is constantly moved (unless this is compensated for) or if one wishes to identify common behaviours in a scene independent way.

2.2 Behaviour Modeling

One of the major aims of a computer vision surveillance system is to allow the automatic semantic interpretation of the video sequences. Previous sections presented ideas

which allow representations of important image features throughout a sequence. This section presents a review of the interpretation of those features within the sequence. It is divided into two main sections, pre-defined behaviour and learned behaviours.

One can interpret pre-defined behaviours as those which an expert has defined situations of interest from well known patterns and behaviours, such as tactics in a football game. These situations are then to be automatically identified by the computer when they appear in the video sequence.

The definition and representation of such models has been made explicit by the designers of the system, although the behaviours which they represent may of course be very general. When the behaviours are learned, it is meant that their representation is automatically encoded within the parameters and structure of a model. In this case a set of manually selected video sequences showing the behaviour of interest is usually given to the computer and the representation of such a situation is created by the system.

2.2.1 Manually Defined Behaviours

Within the general A.I. community there has been much previous work upon trying to form an automatic understanding of many situations and behaviours. One of the earliest attempts to provide an interpretation of situations is with Minsky's frames [Minsky, 1975] which presented a framework for interpreting situations that an artificial agent may encounter. Other related approaches, such as scripts [Schank and Abelson, 1977] and schema's [Bobrow, 1977] have been proposed which seek to describe various situations in a compact linguistic format.

Such a high level approach is pursued by Crowley and Reignier [Crowley and Reignier, 2003] who describe a generic software processing model for the interpretation of context when observing human activities. Some parts of this theoretical framework have been implemented in the "Context Toolkit" as proposed by Salber *et al.*[Salber et al., 1999].

Much research has been centred upon the idea of representing an observed scenario through a compact description. In particular much work has been focused upon the automatic recognition of activities based upon a pre-specified template of specific behaviour. Before such templates are be applied there is usually a tracking stage where persons or objects of interest are applied. There are frequently zones or areas of interest which are marked on screen and combined with the tracker it is possible to tell

(a) (b)

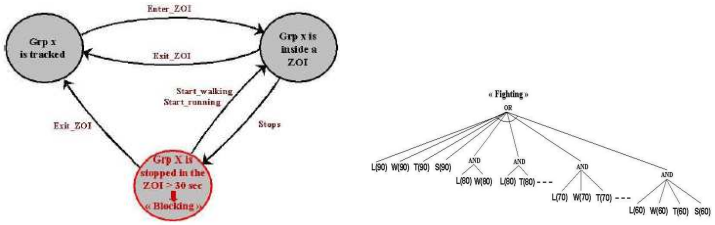
Figure 2.6: Zones of interest defined within the image. For image (a) the zone of interest is defined around the barriers where a three dimensional representation of the barrier is given (in yellow). Here a person is jumping over the barrier which is something the system looks for, and on this occasion is flagged. Whilst in (b) the space in front of the exit (shown in red) is defined as a zone of interest. Here the system correctly detects that there is indeed a potential blocking event being caused although it is not clear who they are blocking (as there are no passengers leaving the station at this time). Adapted from [Cupillard et al., 2004].

where a person is and whether they are in a specific region of the scene. Figure 2.6 gives an example of how a zone of interest is created and used.

In [Rota and Thonnat, 2000] Rota and Thonnat propose a declarative representation of activities defined as a set of spatio-temporal and logic constraints for recognising activities. In Gerber, Nagel and Schreiber [Gerber and Schreiber, 2000] fuzzy logic was used to recognise pre-specified activities within the scene over a short time period. Van Vu [Van Vu et al., 2003] also define behaviour in a similar way by using pre-defined scenario model where the actors, sub-scenarios and constraints are used to form templates. Within the work they propose a method where only a limited subset of the possible transitions from one state to another are considered.

The ADVISOR system [Naylor and Attwood, 2003] is a large scale visual surveillance system which uses a pre-specified scenario to identify behaviours of interest. The behaviours are not defined linguistically, as in the case of [Van Vu et al., 2003] but are realised as a finite state machine (FSM) [Gibson, 1999]. Several behaviours are demonstrated within the system, such as fighting, vandalism, overcrowding and barrier jumping [Cupillard et al., 2004] (figure 2.6and 2.7). Datta *et al.* [Datta et al., 2002] also specifically identify criminal or hazardous behaviour within the observed

sequence although they use a less flexible template approach.



(a) (b)

Figure 2.7: The FSM represents those specific states which occur within the activity of interest and the allowable transitions between such states. This example is used for the detection of pedestrians blocking an area which has been defined as being a zone of interest within the visual scene. (a) Finite State Machine representing blocking of a zone of interest, the states are supplied by interpreting the information given as output by the vision component of the system. (b) tree, figures from [Cupillard et al., 2004]

Using a FSM representation (as in [Cupillard et al., 2004]) it is possible to define simple behaviour templates which can consist of individuals or groups displaying a relatively simple sequence of state transitions. For situations involving many interacting individuals a tree representation is used to represent the behaviour of interest. This tree is composed of sub-sequences within the overall behaviour. This method is based upon the previous work of [Van Vu et al., 2003] who described a method to match pre-defined temporal scenario descriptions to observed features in near linear time complexity. To recognise the situation where people are fighting several individual sub-sequences are defined (such as lying on floor, high group velocity) in a tree structure (see figure 2.7 (b)). When there are more of these sub-activities recognised the time threshold for the duration of events decreases. When more sub-activities are concurrently detected they do not have to have as long a duration to be identified as performing a specific behaviour.

A pre-defined representation is also used by Datta [Datta et al., 2002] who specifically detects instances of fighting between two people. They use a predefined constrained sequence of events between two interacting individuals from an extracted video stream. Events detected are the raising of an arm, the recoil motion of a person's head and the proximity of one person to another. Such a recoil measure requires the localisation of a person's head within the scene, whereas Cupillard *et al.* [Cupillard et al., 2004] use coarser measures such as velocity change and splitting of tracked groups. Pre-defined behaviour is also the focus of Ryoo and Aggarwal work in [Ryoo

and Aggarwal, 2006a] and [Ryoo and Aggarwal, 2006b] where temporal information is included within a scripting framework for the detection of two person interactions. This method was later extended to also include interactions with objects [Ryoo and Aggarwal, 2007]. The work of Ryoo and Aggarwal required the accurate partitioning of a human to determine some of the lower level behaviour which needed to be detected in order for the overall sequence to be correctly recognised.

Methods requiring a detailed knowledge of the human body have been shown to be useful when classifying interactions. The work of Park, first with Aggerwal [Park and Aggarwal, 2004] then later with Trivedi [Park and Trivedi, 2007] proposes a system for classifying two person interactions. Each body part is recognised with both a ellipse and a convex hull, with the ellipse model giving a greater degree of detail. These parts are estimated using a Bayesian network (as shown in figure 2.8). The estimated parts are then used as input into a dynamic Bayesian network for classification of the actual interaction.

Intille and Bobick [Intille and Bobick, 1999] also represent scenarios with a temporal element. Their representation is goal directed where a pre-defined and pre-parameterised Bayesian network is updated to infer the likelihood of a particular scenario occurring based upon the observable evidence. Each node has a simple yes/no state along with a label indicating whether the state has actually been observed or if it was thought to have been observed. Each of these networks are applied to the object(s) in question and provides a likelihood that the observed evidence supports the proposition that the object in question is attempting to reach this goal.

Perse *et al.* [Perse et al., 2006] take a similar template based approach but this time in the domain of basketball, squash and tennis. Such templates are pre-defined by an expert (normally a sports coach) and refer to a specific pattern of people, movement and ball passing over a temporal window.

In more recent work [Perse et al., 2007] such templates have been extended to incorporate a Bayesian framework. The authors also used a feature detector to evaluate things such as fit to line and screening (where one player blocks another). An example of such an activity network is given in figure 2.9. The figure represents a “double screen play“ where two players screen two of the opposition players whilst the ball is passed. The bottom layer represents the temporal relations between the elements and the arrows represent a dependency. A model’s probability is calculated using an inference algorithm and the class of the model giving highest probability is taken to be class of the current action.

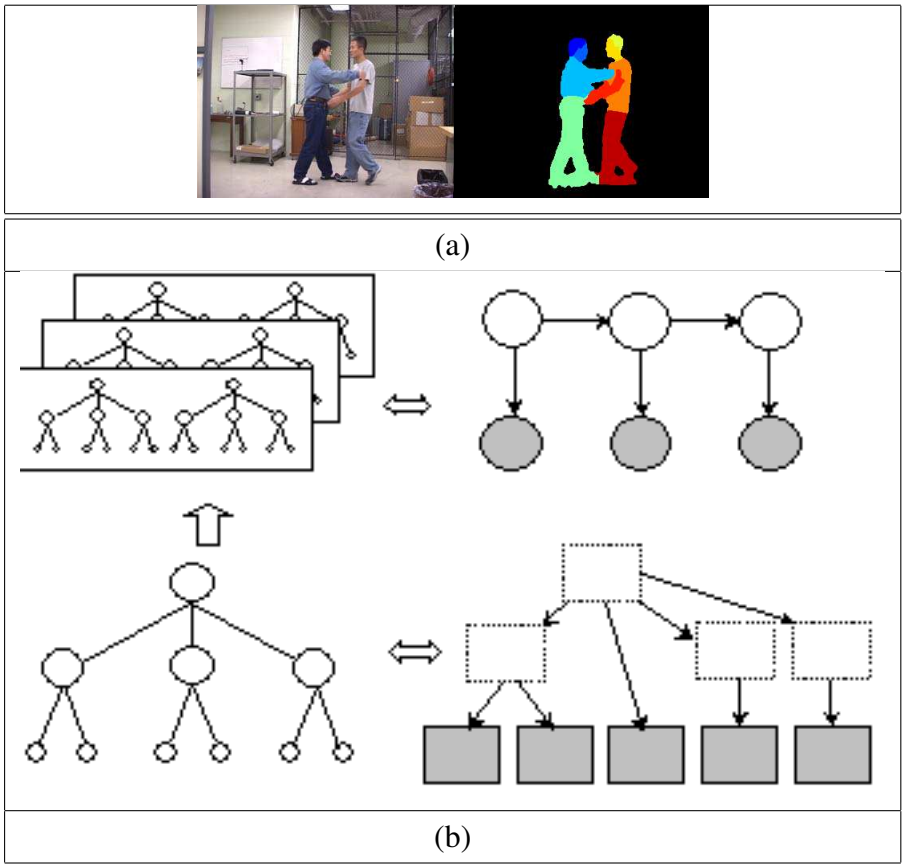


Figure 2.8: (a) Pointing sequence the left image shows the original captured video whilst the image on the right shows the segmented image. Different colours show how the body parts of the two individuals have been segmented. (b) The lower-left panel shows a tree structure that represents individual body part regions of a person. The lower-right panel shows a Bayesian network to estimate the body poses in each frame. The upper-left panel shows a sequence represented by the concatenation of two tree structures for two interacting persons at each frame. The upper right panel shows a dynamic Bayesian network, which recognizes the two-person interactions. Adapted from [Park and Aggarwal, 2004].

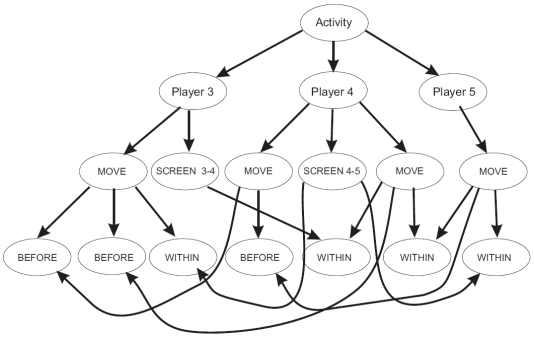


Figure 2.9: Example for the Bayesian network for the Double screen activity. Nodes represent the variables and the arrows represent probabilistic relations between variables. Adapted from [Perse et al., 2007]

In a related approach Shi *et al.* [Shi et al., 2004] propose a sequential representation of temporal actions. The network is represented as a Bayesian network with parent nodes representing a previous step in the sequence (see figure 2.10). Each node has an associated duration model realised as a Gaussian distribution.

The duration model represents the likelihood of the node being active once it has been activated (ie once the state has been reached). Due to the potentially large state space a condensation based probability propagation method is presented (d-condensation). The hypotheses which are considered by the d-condensation algorithm are constrained to only come from the set of reachable states (from the current state(s)). This step helps to avoid problems where many hypotheses will have similar probabilities, as has been previously commented upon [Arulampalam et al., 2002]. This model can be thought of as a sequential, probabilistic FSM with an explicit duration model.

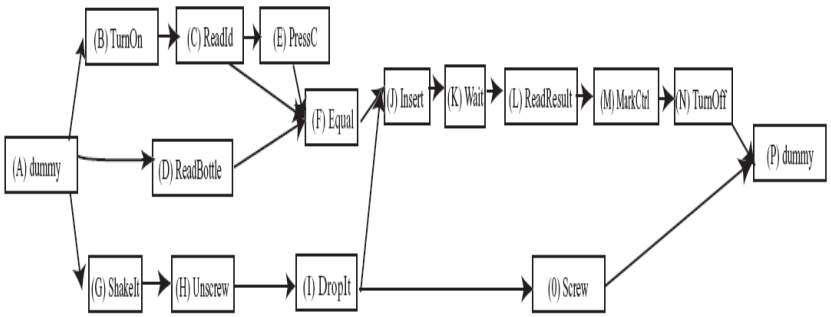


Figure 2.10: Propositional network for performing a glucose calibration task. Each state is represented as a box with the directional arrows showing the allowable subsequent states from that state. Each state has a duration model associated with it. Adapted from [Shi et al., 2004]

2.2.2 Learned Behaviour

In contrast to hand constructed models for classifying an action sequence, learned methods seek to automatically build such a representation. Here the action sequences are learned and represented automatically from data rather than being defined by an external entity.

Automatic learning of behaviour between two people was proposed by Galata, Johnson and Hogg [Galata et al., 2001, 2002], who build variable length models of human behaviour. In order to derive a tractable representation of the state space generated by a person's actions a prototyping approach [Johnson and Hogg, 1996] is used. Action sequences are represented as a sequence of finite vector prototypes which can be automatically learned from training data. Vector quantisation has also been employed by [Brand and Kettnaker, 2000, Oliver et al., 2000a, Kadir et al., 2004] for the purposes of generating a discrete representation of a continuous state space.

A variable length Markov model (VLMM) [Ron et al., 1996, Guyon and Pereira, 1996] was used by Galata *et al.* [Galata et al., 2002, 2001] to represent and predict observed behaviour. A string of tokens, w, representing previous states, are used to predict the next token a' according to the estimate $\hat{P}(a'|w)$ (where the hat $\hat{P}$ represents an estimate) of $P(a'|w)$. The Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951] (shown in equation 2.3) is used to measure the difference in distributions to determine if a longer memory should be used to potentially improve the estimate. If this measure exceeds a manually set threshold (ε) then the longer memory (aw) is used, otherwise the shorter memory is considered sufficient.

$$KL(aw,w) = \hat{P}(aw)\sum_{a'}\hat{P}(a'|aw)\log\frac{\hat{P}(a'|aw)}{\hat{P}(a'|w)} \tag{2.3}$$

Here $KL(aw,w)$ refers to the KL divergence (also known as cross entropy), a is the current token, a' is the next token and w is the memory, aw refers to the longer memory.

The modeling of the actual behaviour of a person is done by constructing a set of key prototypes. Such prototypes are established by clustering the extracted human point models, as shown in Figure 2.11(b). Cluster centres are then used to form action sequence templates. This stage is necessary so that a variable length Markov model (VLMM) can represent actions in a sufficiently abstract way to recognise high level behaviour. These templates form the alphabet which is used by the VLMM. This is illustrated in Figure 2.11.

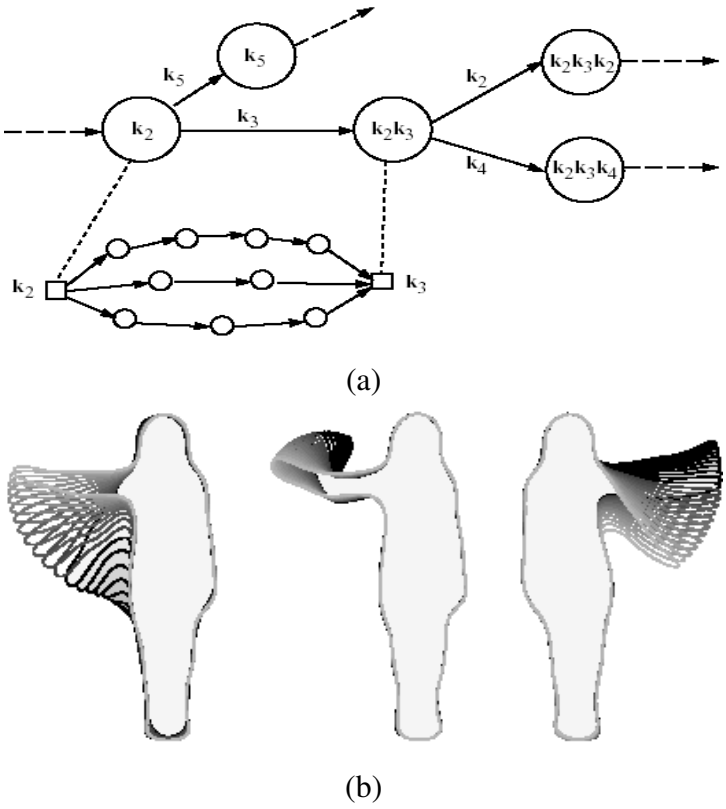


Figure 2.11: (a) Illustration of the hierarchical model used [Johnson et al., 1998] showing the states of the PFSA corresponding to the memories of the VLMM. Three alternative templates for a movement between k2 and k3 are also given. (b) Predictive behaviour for the learned model.

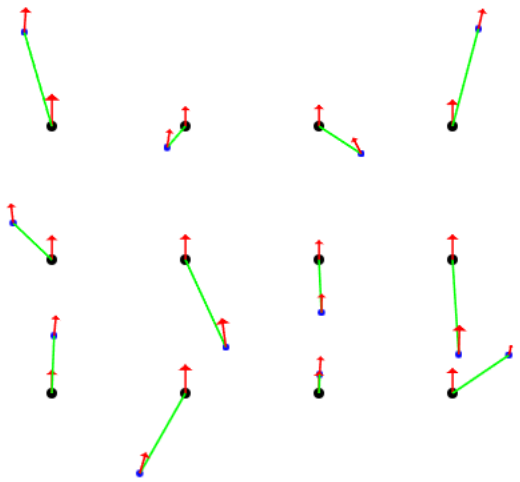


Figure 2.12: Learned primitive interactions – traffic domain example. The two dots represent pairs of close vehicles (larger dot being the reference vehicle). The arrows show their direction of movement. These patterns represent typical "midpoints" as result of clustering the input data into B different conceptual "regions". Adapted from [Galata et al., 2002]

Johnson *et al.* [Johnson et al., 1998] use a VLMM to generate virtual interactions. Computer generated models are learned which can automatically recognise types of behaviour exhibited by a person and respond appropriately. This is demonstrated in the form of a handshake sequence between a person and a synthetic agent. VLMM have also been used for the automatic modeling of traffic behaviour [Johnson et al., 1998]. The VLMM was also used to model interactions between vehicles on a section of road. A discrete set of primitives is learned, representing the interactions of the two vehicles with the representative set obtained through clustering (see Figure 2.12). The temporal dependencies in the behaviour of the interacting vehicles are then learned using the VLMM model in a similar way to [Johnson et al., 1998].

The idea of using feature space prototypes was investigated in Blunsden *et al.* [Blunsden et al., 2007a] who use a sliding window in feature space to compare with the correct time window. The Hausdorff distance is used for comparison of the two feature trajectories over a fixed time window.

Hidden Markov models [Rabiner, 1990] have been used in modelling temporal problems within the vision community by several authours [Oliver et al., 1998, Brand and Kettnaker, 2000]. The observation at time t is defined as y_t. The parameters of the model are given by $\lambda = (\Pi, \mathbf{A}, \mathbf{B})$ and are determined by the expectation maximisation (EM) algorithm as described by [Dellaert, 2002]. The prior distribution $\prod$ represents

the prior probability of each state $\pi_i = P(x = i)$. The hidden state to hidden state transition matrix is given by **A**. This represents the probability $P(x_t = i|x_{t-1} = j)$. Each entry in the matrix **B** represents the emission probability $P(y|x = j)$ of making observation y_t when in state j. To model continuous input a continuous model such as a mixture of Gaussians, [Duda et al., 2000] can be used.

Representation through a sufficient level of abstractness is investigated in Bui, Venkatesh and West [Bui et al., 2002] who proposed a multiple layered abstract hidden Markov model (AHMM). They are not the first people to propose using various HMM inspired representations to represent the level of abstractness. Oliver, Horvitz and Garg [Oliver et al., 2002] used a layered bank of HMMs to encode increasingly abstract concepts when modeling human activity within an office situation. Bui, Venkatesh and West's approach consists of using various HMM models in a connected layered approach (cf Oliver *et al's* approach was not connected) where higher layers represent a more abstract (coarse) description of the concept. The architecture of such a model is given in figure 2.13.

This architecture is modeled so as to represent a process which can be decomposed into various sub-goals. This has an advantage over using a standard HMM representation where invalid transitions are permitted. If one were to imagine a simple printing task, there are sub goals which need to be completed, such as, write a document, send document to printer and collect document from printer. Knowledge that the task 'send document' has been completed would imply the next step is to attempt collect it which would guide future behaviour. This goal directed behaviour is a similar idea to that proposed by Intille and Bobick [Intille and Bobick, 1999], as described in the previous section.

This idea is further expanded in [Nguyen et al., 2003] where a memory component is added to the model. This step allows the model to store a sub-goal sequence which it was previously following. In doing so more complex behaviours can be represented by, for example reverting back to an old sequence, or knowing that the current sub-goal sequence has been tried so try something else. Having a memory component thus allows the representation of (slightly) more complex behaviours.

The ideas proposed by Galata *et al.* [Galata et al., 2001, 2002, Johnson et al., 1998] and Nguyen, Bui, Venkatesh and West [Bui et al., 2002, Nguyen et al., 2003] have concentrated upon modeling the behaviour of a single person. In Galata *et al.* there is interaction with another human but this was represented as a learned response to a person's actions. In Oliver *et al.* [Oliver et al., 2000a, 2002, Oliver, 2000] the

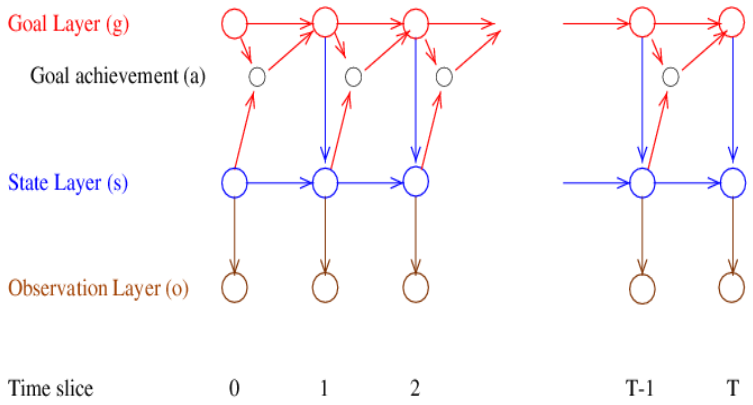


Figure 2.13: The abstract hidden Markov model (AHMM). Higher up in the chains represent more abstract concepts. Within this particular model the highest level is the goal, in this example the goal is to print a document. Lower levels represent more specific items such as what the subject is currently doing, eg sitting, walking, at computer. Note also the specific inclusion of the goal achievement node forming a buffer between the state and goal layers.

interaction between people is specifically represented and modeled. Within this work both a HMM [Rabiner, 1990] and a coupled HMM (CHMM)[Brand, 1996a] are used to represent the interactions between individuals. Here only the CHMM is reviewed as it produced far superior results in conducted experiments.

The CHMM model, as presented by Oliver *et al.* [Oliver et al., 2000a] and (separately) Brand [Brand, 1996b] consist of two interacting hidden Markov models where the hidden node of each chain is connected to the hidden node of the other chain(s) at the next time-step. The three chain version of this model is illustrated in figure 2.14. For cases where there are more than two interacting chains approximation techniques are needed [Jordan et al., 1999]. When there are only two interacting chains exact inference may be computed [Brand, 1996a, Saul and Jordan, 1995].

Oliver *et al.* [Oliver et al., 2000a] use a set of video sequences, which exhibit pre-defined behaviours are used as input to the model. From this visual input the magnitude of velocity of each individual ($v_i = \sqrt{\dot{x}_i^2 + \dot{y}_i^2}, i = \{1,2\}$), the change in distance between two people ($\dot{d}_{1,2}$) and the degree of alignment between two people ($sign(v_1.v_2)$) are used as features. Image features were extracted using background subtraction and representing people as simple blob models. It should be noted that the input data is identical for each chain except for the velocity which is unique to each individual.

Howard and Jebara [Howard and Jebara, 2004] use a hierarchical tree architecture to aggregate several person's trajectories (again within the domain of American football) into a Markov process describing the team activity. Although they only present limited results the approach presents a plausible interpretation of the structure within the game. Each player is modeled, then the team, then the game as a whole. Others such as Zhang *et al.* [Zhang et al., 2000] have also used an aggregating approach for group action clustering in meetings. However in this instance a hierarchical architecture is not as explicit.

One of the shortcomings of a hidden Markov model approach is the lack of an explicit duration model, this can be problematic especially with regard to its geometric distribution. Such a shortcoming has been highlighted by Tweed *et al.* [Tweed et al., 2005] who proposed using a hidden semi Markov model to model the duration of events. Du *et al.* [Du et al., 2006] also address the issue of modeling the varying length of time an activity can take. They apply a hierarchical duration-state dynamic Bayesian network (HDS-DBN) to the problem of two person interactions. The features are divided into local and global features before being used within the HDS-DBN model.

2.2.3 Modeling Interaction Dynamics

Within this section the case where there may be multiple interactions is considered. There are significant computational challenges when an arbitrary number of links or interactions are introduced, each of which may or may not shape the future actions of other processes. Here a review of some of the challenges and potential solutions when using such representation are identified.

When modeling an arbitrary number of potentially interacting people a C chain CHMM is the natural extension (where C is the number of chains in the model), such a model is shown in figure 2.14 (right) along with a standard HMM (left) for comparison. Xiang and Gong [Gong and Xiang, 2003b,a] have followed this approach, where each Markov chain within the model represents an event type (eg truck moving, cargo unloading). The specific domain of interest is in modeling cargo loading/unloading situations at airports. A model set consisting of all possible couplings ($O(C^2)$) between the chains is created.

Each potential model has its parameters trained by an Expectation Maximisation (EM) [Dellaert, 2002, Minka, 1998] algorithm. Models are then evaluated using a

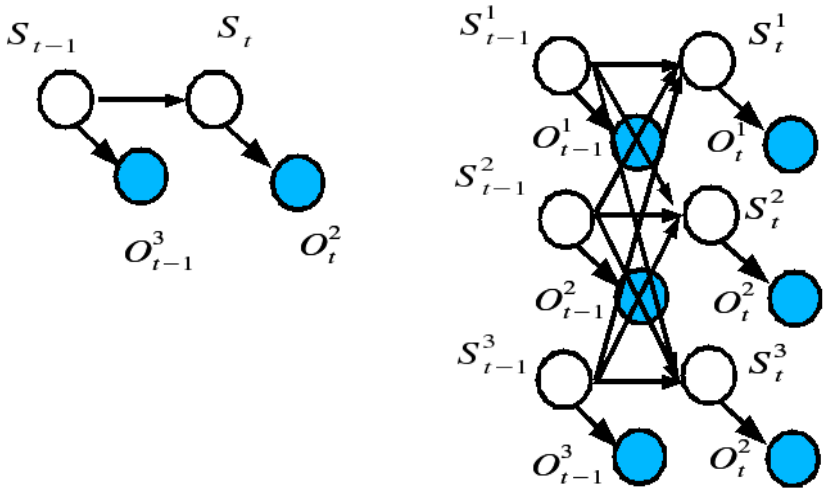


Figure 2.14: The HMM (left) and the CHMM (right), visible nodes are shaded, hidden nodes are not. Note that each chain within a CHMM is in itself a fully specified HMM model with its own observations and state transition parameters. How these parameters are updated distinguish such coupled models from being a collection of individual HMM's

information theoretic metric, in this case the Bayesian Information Criterion [Schwarz, 1978] or Bayesian Dirichlet score (BDe) [Heckerman et al., 1995] to measure the fit of the data upon the trained model. A penalising term to discourage overly complex models is also inherent within both metrics.

The model with the best score is chosen to represent the situation, this method is illustrated in figure 2.16. Using such an evaluation scheme the chosen model structure successfully classified more situations correctly compared to a fully coupled CHMM model. This situation can be explained by the learned model structure accurately representing the state transition diagram of the sequence (shown in figure 2.15 (b)). This structure forms a constraint over the representable form of the posterior distribution so that it is closely related to the actual state sequence. When using a fully coupled CHMM it is likely that the inherent freedom in a fully coupled CHMM means that spurious features such as noise are learned and represented. This can be confirmed by looking at the presented error rates where a fully coupled CHMM has a recognition rate of around 60% compared to 90% for a sparsely connected model.

Whilst using such an approach would allow the modeling of an arbitrary number of interactions in a single framework, (which was desirable for the purposes it was used in) there are a number of problems inherent in such an approach. When learning a complete model in this way interactions are characterized by the presence or absence

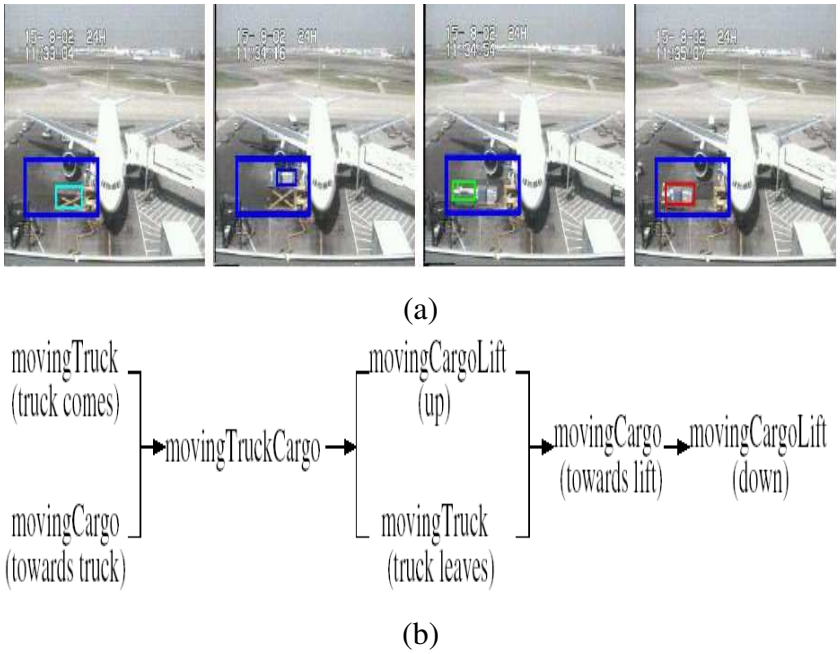


Figure 2.15: (a) The airport loading task which Xiang and Gong have model. Above the sequence shows the visual input to the classifier. (b) The actual task being modeled represented as a state transition diagram. This diagram was manually constructed. Figures are adapted from [Gong and Xiang, 2003b].

of links from one process to another. Additionally the actual influence that one chain has upon another is embedded in the parameters of the state transition matrices of each chain. There is no explicit representation of interaction other than a presence or absence of links within the model.

This hard yes/no threshold does not integrate well with the Bayesian approach where the ability to deal with uncertainty remains one of its key contributions. The other major drawback is that each model has to be evaluated separately meaning that there are (assuming all possible couplings) $O(C^2)$ potential couplings to train and evaluate. The authors also do not provide details of the learning algorithm used to train such models.

Using a standard exact EM algorithm the learning algorithm for CHMM's requires the estimation of $O(TN^C))$ parameters per chain (as given in [Brand, 1996b]). However Brand [Brand, 1996b] has shown that it is possible to approximate this by dynamic programming using a algorithm of $O(T(CN^2))$. Unfortunately this is only realistically possible for a small (he states 4) number of chains due to the exponentially smaller state space being sampled as more chains are added. Other approaches such as decimation [Saul and Jordan, 1995] have been suggested, although such a technique is

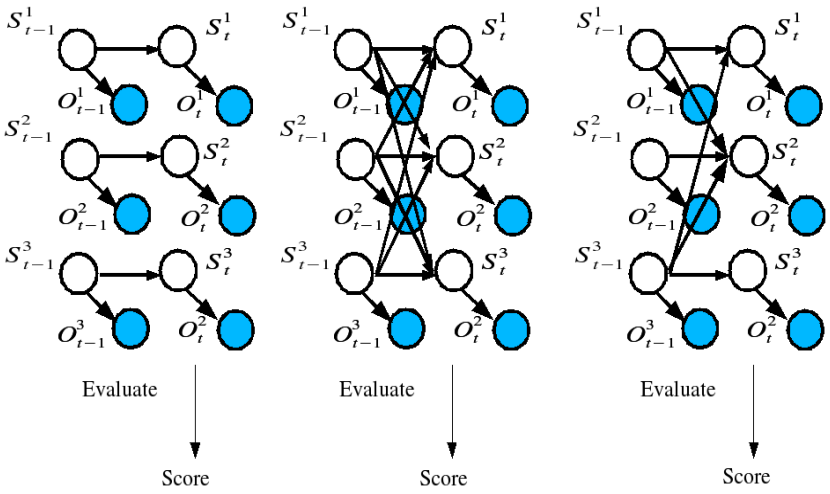


Figure 2.16: Evaluation scheme used in [Gong and Xiang, 2003b,a]. Each model is trained with the data and then a score is generated from an information measure (see text) for each model model architecture. The best scoring model is the one which is chosen to represent the situation.

only possible for a limited class of graph structures. Another approach taken by some authors is to use Monte Carlo methods [MacKay, 1998] to sample the state space and generate good estimates of the model's statistics.

Kwon and Murphy [Kwon and Murphy, 2000] use domain knowledge to restrict couplings so that only a subset of possible couplings are made. Others, such as Tian *et al.* [Tian et al., 2003], Choudhury *et al.* [Choudhury and Pentland, 2004, Choudhury et al., 2003, Basu and Choudhury, 2004, Basu et al., 2001] and Zhong and Ghosh [Zhong and Ghosh, 2001a,b] approximate the joint distribution when computing the posterior distribution. Saul and Jordon proposed a tractable approximation for the task of modeling an n-order Markov model [Saul and Jordan, 1999]. This approximation can be modified slightly so that it represents an C-chain coupled model. This approximation is shown in equation 2.4. It can be seen that the posterior can now be calculated as a weighted sum rather than requiring the full joint distribution.

$$P(S_t^c|S_{t-1}^1,..,S_{t-1}^C) = \sum_{c'=1}^{C} \psi_{c,c'} P(S_t^c|S_{t-1}^{c'}) \tag{2.4}$$

The coupling strength from chain c' to c is given by $\psi_{c,c'}$. There is a constraint upon the influence parameter in that $\sum_{c'=1}^{C} \psi_{c,c'} = 1$.

Such an approach is taken in Choudhury *et al.* [Choudhury and Pentland, 2004, Choudhury et al., 2003, Basu and Choudhury, 2004, Basu et al., 2001] where each

individual is represented as a separate HMM model. This method is an extension of that proposed by Asavathiratham [Asavathiratham, 1996] for representing influences between interacting Markov chains. Here HMM models are used in place of Markov chains and the influence parameter is learned rather than a-priori fixed. A debating game is used for testing the model where speech cues provide the input to the model.

Both the HMM parameters and the overall influence are automatically updated during the learning stage. Due to the use of a HMM the hidden state sequences are revealed using a Viterbi decoding [Forney, 1973] and then the re-estimation of the influence between people is computed. This step is necessary so that the model represents a coupling at the hidden layer rather than trying to model a coupling at the observation layer. They effectively make the "hidden" parameters "visible". Other parameters of the HMM are updated using the Baum-Welch algorithm [Baum, 1969].

One of the disadvantages of such an approach is that a Viterbi decoding can introduce inaccuracies into the learning phase and that it is replacing a probabilistic measure of the hidden state with a definite state, which is not always appropriate. In Zhong and Ghosh [Zhong and Ghosh, 2001a,b] the interaction is modeled directly at the hidden layer using a modified forward backward algorithm. As the HMM's are coupled together the forward and backward parameters are also modified to represent this situation (equations 2.5-2.7).

$$\alpha_1^c(j) = \pi_j^c b_j^c(x_1^c), t = 1, 1 \leq c \leq C \tag{2.5}$$

$$\alpha_t^c(j) = b_j^c(x_t^c) \sum_{c'=1}^{C} d_{c,c'} \sum_{i=1}^{N_c} \alpha_{t-1}^{c'}(i) a_{i,j}^{c,c'}, 2 \leq t \leq T, 1 \leq c \leq C \tag{2.6}$$

$$P(O|\lambda) = \prod_{c=1}^{C} \left[\sum \alpha_T^c(j)\right] \tag{2.7}$$

The extended forward backward variable $\alpha_t^c(j)$ is defined for state j for the c^{th} chain at the t^{th} timestep. The prior distribution is given by π_j^c with the hidden to observed probability probability being given by b_j^c. The number of chains is represented by N_c. The indicator function $d_{c,c'}$ is set to one if there is a link between chain c and c' or zero otherwise. Tractable learning within such a model is performed using a EM algorithm which is discussed in [Zhong and Ghosh, 2001a]. The domain of airline loading/unloading has also been researched by Vaswani *et al.* [Vaswani et al., 2003] who used a particle filter to model the normal behaviour of passengers walking to and from a plane.

In Tian *et al.* [Tian et al., 2003] a gradient based version of this algorithm is presented. It is suggested that such a gradient based method will give smoother on-line learning in the coupled case in much the same way that Baldi and Chauvin [Baldi and Chauvin, 1994] showed for the standard HMM. EM learning can give large jumps in parameter space, whilst gradient based equivalents have been shown to give smoother changes in the learning parameters. Here the influence model is applied to discover influences within customer data for a marketing application. Results are presented and prove comparable to those of the self mapping described by Zhong and Ghosh [Zhong and Ghosh, 2001a] on large scale datasets. There are no results presented which give proof of the gradient approach having smoother on-line learning in this model. However the rationale for this assumption is based upon the work of Baldi and Chauvin's [Baldi and Chauvin, 1994] findings for a single HMM.

Other authors have attempted to extend the HMM to incorporate multiple interacting agents. Lui and Chua [Liu and Chua, 2006] modified a multiple input HMM by assigning a role parameter to each input. The role a person plays, such as a victim or perpetrator of a mugging is assigned while the inference stage is occurring. This makes it possible to handle three person interactions where the roles are different within that interaction. Unfortunately it requires scenarios where exactly three people are interacting as both training and testing sequences.

2.2.3.1 Structure Learning

An alternative approach taken is to learn the complete model structure from the data alone. Several authors have proposed such approaches[Friedman et al., 1998, Friedman and Goldszmidt, 1996, Koivisto and Sood, 2004, Heckerman et al., 1995]. Although this work has not been formally applied to vision research it has been applied in other domains (specifically finance and simulation). Friedman introduced the structural expectation maximisation (SEM) algorithm [Friedman et al., 1998, Friedman and Goldszmidt, 1996] for learning Bayesian network structure. The SEM algorithm interleaves the structure discovery and parameter estimation step of model learning. The automatic determination of structure allows one to see the relationships between the variables in the data. In the case of human interaction modeling this would represent the interactions between people. However such modes currently represent static Bayesian networks and so may not best represent continuously changing data. One possible use for structure discovery would be for learning an action sequence in much the same way as Cupillard *et al.* [Cupillard et al., 2004] used FSM's to represent action

sequences.

In related work Kubica *et al* [Kubica et al., 2003a,b] seek to establish most likely groupings based upon link data. Such an approach allows multiple interacting individuals to be grouped together for further analysis. By identifying groups present within an image (his was not done in this case) the amount of data which requires processing could be reduced, as only properties of the group need be considered. However there has been no specific application of such an approach to vision problems.

2.2.3.2 Applicability to our Problem

Of these structure models a CHMM structure seems the most appropriate for the task of modeling interacting people. By using a causal model where influence is directly represented is appropriate for modeling interactions. Other approaches to structure learning such as SEM and group learning assume that the structure is fixed throughout all data samples. This could be modified to incorporate a time dimension but would increase the computation cost. Using a CHMM approach models the situation to be represented in the most natural way.

2.3 Groups

Methods such as those outlined above in section 2.2.3 require the group size to be determined apriori. Within the context of a general surveillance system this is an unrealistic assumption. In [Khan and Shah, 2005] Khan and Shah propose a way of recognising groups of interacting people based upon the rigidity of their formation. Individuals within the group are tracked by finding their heads and mid points (thus giving 2 points per tracked person). These tracks are used to form a 3D shape describing the structure of the group. A rigid shape such as this is present in structured groups such as people marching in a parade or convoys. As such the method works well when identifying rigid group activities and is able to distinguish when a non-rigid event is occurring.

Khan and Shah's method can deal with regular and rigid group patterns where the distance is constant and regular (subject to some small variation). However this is a restricted class of group behaviour. Within surveillance video a much richer class of group interactions is required to be modeled and identified.

Hong and Nevatia [Hongeng and Nevatia, 2001] build up a representation from a collection of lower level features rather than enforcing a global model. The event

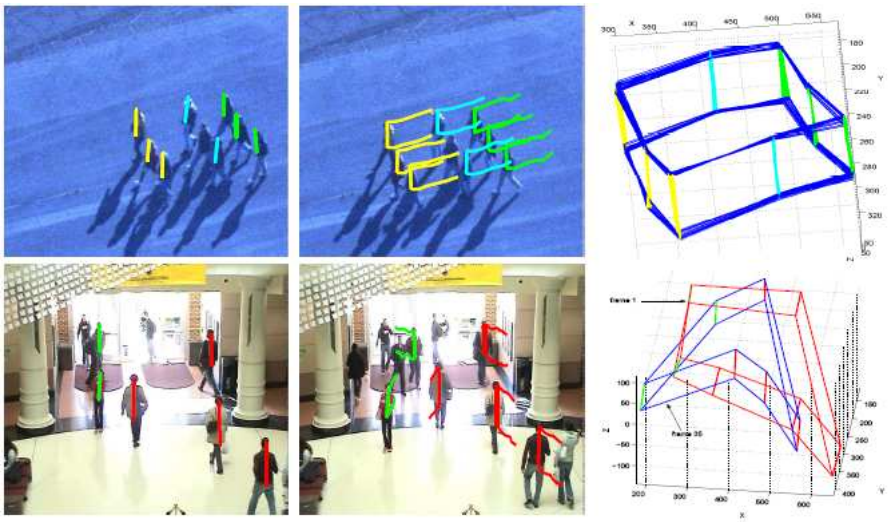


Figure 2.17: Rigid events (top row), give a distinctive rigid shape that is consistent throughout the observed frames. The lower row shows a random crowd. Adapted from [Khan and Shah, 2005]

detection is completed by a hierarchy of features derived from a blob tracker. This hierarchy can be seen in figure 2.18.

Events are represented by finite state automata such as those shown at the top of the hierarchy of figure 2.18. A probabilistic evaluation of such events is used, mainly due to the uncertainty within the system. The probability that the complete automation state sequence of a multiple state complex event (MS) occurs within the most likely state transition timing given the sequence observations O :

$$P(MS*|O) = \max_{\forall(t_{1,..,}t_N)} P\left(S_{1(t_1,t_2-1)}S_{2(t_2,t_3-1)}\cdots S_{N(t_N,t_N-1)}|O\right) \tag{2.8}$$

Within equation (2.8) the term t_i refers to the time that the transition to state i from state $i-1$ occurs and $S_{i(t_i,t_{i+1}-1)}$ means that scenario i occurs occurs between t_i and $t_{i+1}-1$.

A specific ontology was derived by Hakeen and Shah [Hakeem and Shah, 2004] for the classification of meetings. This was later extended in [Hakeem and Shah, 2007] to learn, the structure of group interactions. The actual detection of lower level events such as meets, moves, stops etc is not detailed (only that there is some function which can detect this). In order to learn the event structure a sub event dependency graph (shown in figure 2.19) is created which depicts the conditional dependency between sub events. These sub events are then clustered using spectral clustering under the assumption that events of correlated sub events have high intra clustering weights and

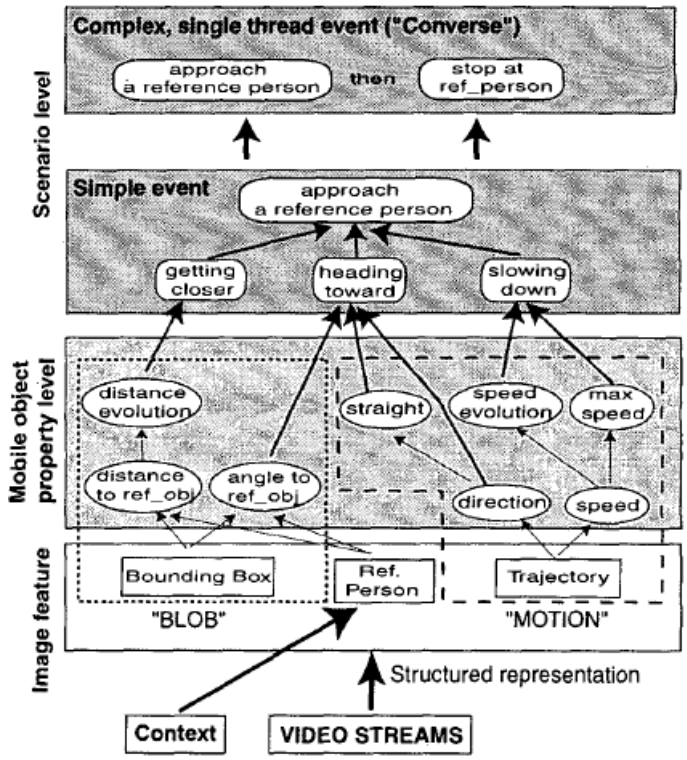


Figure 2.18: Hierarchy of a 'converse' event. Simpler features are combined to form higher level features which enable recognition of the whole event. Adapted from [Hongeng and Nevatia, 2001].

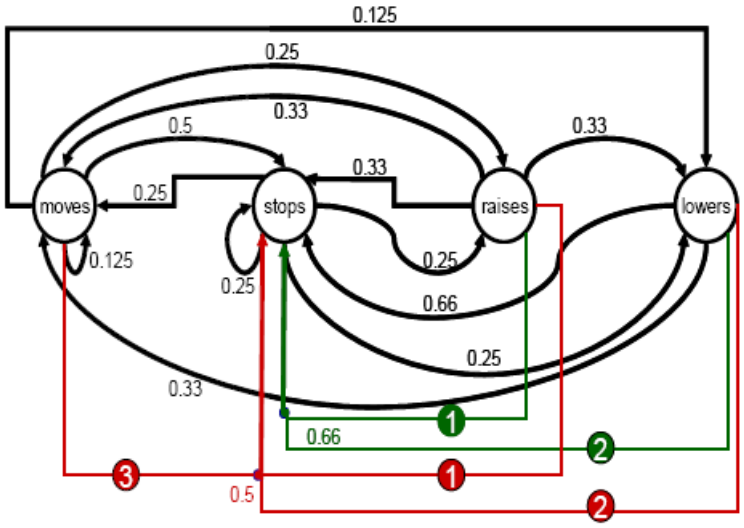


Figure 2.19: Partial event correlation graph for the sample video of voting events. The sub-events are the vertexes, and the conditional probabilities between sub-events are represented by the weights on the hyperarcs. Note that a single example of hyperarcs with cardinality of 3 and 4 are shown respectively in green and red, so as to keep the figure comprehensible. Also, the circled number on the hyperarc represents the order index in Pi, e.g. the B-arc of cardinality 4 represents P(stops|moves,lowers,raises). Adapted from [Hakeem and Shah, 2004]

low inter cluster weights. The method effectively builds up a representation of a more complex event using common sub events and the relations between them.

Hosie *et al.* [Hosie et al., 1998] also take a bottom up approach and use pair primitives (such as converge/diverge) between two people to build up a set of rules which define an interacting group. These rules are pre-scripted and define what a group is. The system can cope with people leaving and joining a group with no set limit on group size. Unfortunately the data upon which the system was tested is limited although the rules may well of applied for a larger test corpus.

2.4 Tracking

Here a section on tracking is included with the intended purpose of showing that tracking technology is available and it is reasonable to assume that it can provide tracks which will enable behaviour detection to take place. Tracking is not the main objective of this thesis, however it is important to show that it is sufficiently developed to

enable behaviour recognition to take place. The purpose of a tracker is to locate an object of interest in every frame and keep a consistent label for the object throughout the video sequence. There may be many objects within a scene and it is a tracker's job to accurately record the positions of all such objects. Trackers may be responsible for identification of the objects as well as keeping track of them. Here we concentrate on tracking using a single video camera covering a scene. Other approaches to tracking involve using multiple cameras [Batista, 2004, Fleuret et al., 2007] or specialist 3D scanning equipment [Cui et al., 2006].

2.4.1 Object Identification

Background Modeling

Many tracking techniques rely on the removal of the background in order to identify the object of interest. The aim is to accurately model a background so that it can be eliminated from the image leaving only those objects of interest in the image. The question of the correct or optimal colour space to perform such background operations has been investigated by several authors. One of the most commonly used colour spaces is RGB (red, green, blue). It has been empirically shown by Paschos [Paschos, 2001] that the differences in the RGB colour space do not correspond to differences as perceived by humans. Another aspect of the RGB colour space is that it is highly correlated. Colour spaces such as LUV (Luminance, with uv being the chromaticity coordinates of a specified white point) are perceptually uniform color spaces, while HSV (Hue, Saturation, Value) is an approximately uniform color space. A problem with these colour spaces is that they are more sensitive to noise than RGB [Song et al., 1996], for this reason there is no clear best colour space in which to model the background.

Image differencing

One of the simplest ways to remove the background is to have a background image, which contains no objects of interest. The difference between this image and each subsequent image is then calculated at every pixel location. If the difference is greater than a threshold then mark the pixel as being non background. Early examples of this were conducted by Jain and Nagel [Jain and Nagel, 1979] who studied the difference between two temporally adjacent frames. Wren *et al.* [Wren et al., 1997a] went on to propose modeling of each image pixel $I(x,y)$ using a single Gaussian (a 3D Gaussian for Y,U,V colour spaces). The parameters of the Gaussian, namely the mean $\mu(x,y)$ and

covariance Σ are learned from several frames worth of data. Pixels which are deemed to deviate from this distribution are labeled as foreground objects. The issue of how to set this deviation or threshold has been addressed by Ching [Ching, 1994] and Rosin and Ellis[Rosin and Ellis, 1995]. Threshold variation can help when intensity variations are observed and are non-uniform across the image (if done per pixel/region).

One of the main issues with threshold adaption is that image intensity can change for some or all parts of the image throughout time. In addition to this the pixel itself may change. One commonly cited example is that of a curtain constantly being blown across a window causing a bimodal distribution on the pixels colour values. In an attempt to model such occurrences Stauffer and Grimson [Grimson and Stauffer, 1999] use a mixture of Gaussian to model each pixels colour and use dynamic updating of the means and covariance of the mixture model to reflect changes in the visual scene.

Taking inspiration from the relative success of the mixture model approach kernel methods such as those of Elgammal *et al.* [Elgammal et al., 2001] and (separately) Han *et al.* [Han et al., 2004] have been shown to be effective at background modeling. Both the kernel and Gaussian methods require a set (normally contiguous but this is not a requirement) of images in order to build robust background models. One problem that remains throughout all methods is how fast to update the background model. If it is too slow then changing lighting conditions can have a detrimental effect on the estimation of what is considered to be foreground. Conversely if it's too fast then foreground objects can be rapidly "absorbed" into the background model leading to few correct detections. Varying threshold methods are again useful in such situations.

Oliver *et al.* [Oliver et al., 2000a] propose a holistic approach using an eigenspace decomposition. A number of training frames (k) are placed into a matrix B formed by concatenating m rows in each frame to form a background matrix $\mathbf{B}$ of size $k \times l$ (where $l = (n \times m)$). Eigenvalue decomposition is applied to the covariance matrix of $\mathbf{B}$ and the η most significant eigenvectors are then used to represent the background. Foreground images are found by projecting the current image with the η most significant eigenvectors and finding the differences between the original and the reconstructed image.

Segmentation

Clustering methods and related segmentation methods have also been used to model the background distribution. These seek to group together similar parts of the background in an attempt to better model the components of the scene. An advantage over

per-pixel methods is the reduced computational resources required. Authors such as Indupalli *et al.* [Indupalli et al., 2006] and Picardi and Jan [Piccardi and Jan, 2004] have employed clustering methods when modeling the background. In particular Picardi and Jan applied the mean shift method when modeling the background. The mean shift method [Comanicu and Meer, 2002, Rodriguez and Suarez, 2006] uses an initialised (usually by hand) image region and the model is generated based on this region. Typically a probability density function (approximated by a histogram) of the region's intensity values is calculated. The difference between the target and the current region is then calculated and the position is iteratively updated based upon the gradient. Comaniciu *et al.* [Comaniciu et al., 2003] successfully use this method to segment a foreground object and its position throughout a sequence. Collins [Collins, 2003] extends this method to take into account the changing scale of an object through a sequence leading to performance increases when the object's scale varies significantly. Occlusions are the main problem with such methods where the temporary loss of an object behind some scene feature can cause tracking problems. Robertson [Robertson, 2006] suggested a way to overcome such limitations in the standard algorithm.

Other techniques used for segmentation have also been applied to the problem of background removal. One particularly notable segmentation technique which has been applied to background segmentation is that of graph cuts [Boykov and Jolly, 2001]. Graph cuts are an optimisation method where (in this instance) the boundary energy between foreground and background is minimised. This minimal boundary is called the cut and determines what is background and what is foreground. Both interactive [Blake et al., 2004, Rother et al., 2004] and non-interactive solutions [Ahn et al., 2006, Russell and Gong, 2006b] have been proposed. Notably graph cuts methods have enjoyed particular success where the tracked object is occluded by non stationary objects [Russell and Gong, 2006a]. Some results of background segmentation using graph cuts is shown in figure 2.20.

Contour Models

Snakes, such as those introduced by Kass *et al.* [Kass et al., 1988] have also been applied to segmentation problems. A snake is typically initialised so that it completely encloses the object of interest and then an energy function as given in equation (2.9) is minimised.

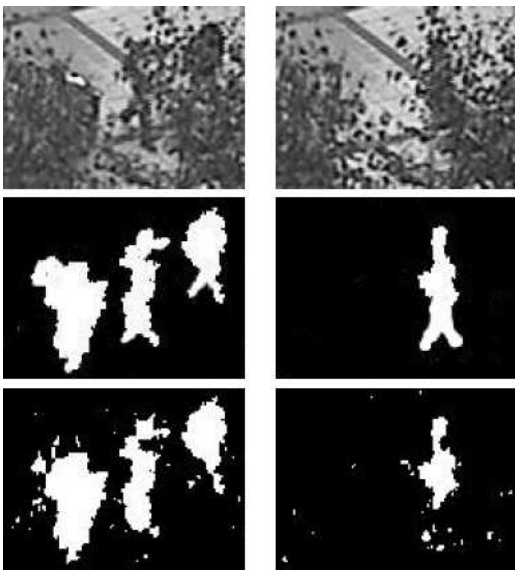


Figure 2.20: Segmentation of a person from behind the moving branches of a tree. The segmentation gives (bottom row) gives a good approximation of the ground truth (middle) in a difficult (even for a human) segmentation problem. Adapted from [Russell and Gong, 2006a]

$$E(\Gamma) = \int_0^1 E_{int}(\mathbf{v}) + E_{im}(\mathbf{v}) + E_{ext}(\mathbf{v})ds \tag{2.9}$$

In this equation the energy term $E(\Gamma)$ with Γ is the contour (as defined by a set of points) s is the arc length of the contour. $E_{int}(\mathbf{v})$ contains the regularisation constraints whilst $E_{im}(\mathbf{v})$ is the gradient energy of the image $\mathbf{v}$. Any additional constraints are contained in $E_{ext}(\mathbf{v})$. This iterative energy minimisation has been solved by gradient descent optimisation [Kass et al., 1988] and also more recently Xie and Mirmehdi [Xie and Mirmehdi, 2007] applied level sets to the minimisation. There is a wide variety of energy functions which have been used by various authors Zhu and Yuille [Zhu and Yuille, 1996] proposed using region information instead of the image gradient. Paragios and Deriche [Paragios and Deriche, 2002] propose using a convex combination of the gradient and region-based energies.

One of the main drawbacks of the snake method is its sensitivity to initialisation and its non-explicit use of a model meaning that the object you are wishing to identify can be ambiguously or ill defined resulting in the model getting trapped in other regions of the image. Pryor *et al.* [Pryor et al., 2007] have used such models within a larger tracking system.

A more constrained version of a snake like model has been used by Cootes and

Taylor [Cootes et al., 1992b,a, 1994] who quantified the statistical variation on a set of landmark points. Recent additions to the model incorporate texture [Stegmann, 2001] information as well. The contour model is then fitted to a new example by iterative optimisation of both the point position and texture similarity. Such models have been used by Magee [Magee, 2000] for the tracking of livestock while human tracking was the focus for Galata *et al.* [Galata et al., 2001] and Baumberg and Hogg [Baumberg and Hogg, 1994a]. The main drawbacks with such methods are the pre-labeling of the training data to get a consistent set of points coupled with the complex motion of human movement, especially arms. Another problem which requires special treatment is the onset of occlusions which can detrimentally affect the model matching stage. There have been attempts to automatically select the landmark features [Saragih and Goecke, 2006, Walker et al., 1999] but as yet there is no universally agreed way of doing so.

Feature Selection

One of the drawbacks of statistical shape models is the requirement for prior and consistent labeling of points to be available. Although several ways of addressing this shortcoming have been proposed [Saragih and Goecke, 2006, Walker et al., 1999] they have only been applied to very specialised and sanitised training data. Viola, Jones and Snow [Viola et al., 2003] propose a method of tracking humans based on Haar like filters. Such filters are applied to difference images of a walking pedestrian as shown in figure 2.21. These features can be calculated quickly by using an integral image representation [Viola and Jones, 2001].

Such features are then used with an AdaBoost [Schapire, 2002] learning and classification procedure. The advantage of such an approach is that it automatically uses both motion and appearance to track pedestrians. However the method requires a large training set which may be time consuming to acquire. Although real time performance can be achieved (see [Comport et al., 2005] for a comparison of other real time methods) peak accuracy of 90% correct detection is not quite good enough for commercial application at present.

A boosted classifier approach was also favoured by Wu and Nevatia [Wu and Nevatia, 2007] who used edgelet features in a nested boosted classifier as shown in figure 2.22. In addition to finding the whole full body in one go Wu and Nevatia break up the pedestrian into 3 additional parts (head and shoulders, torso and legs). These parts are also learned using edgelet features via AdaBoost. The advantage of breaking de-

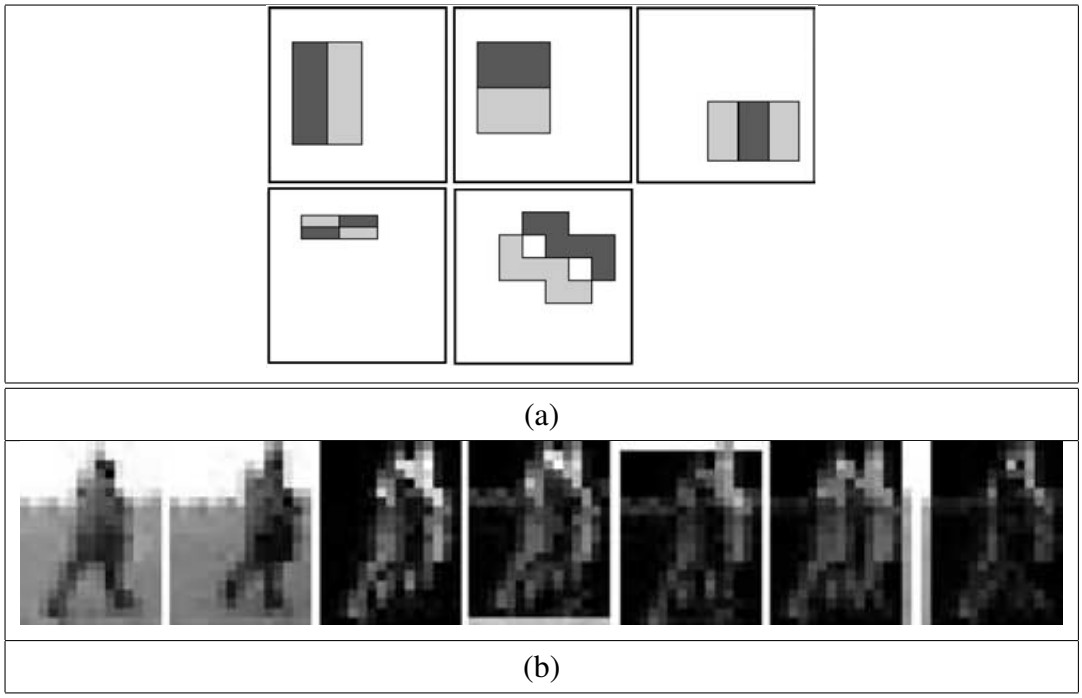


Figure 2.21: (a) Haar like features which are selected by the tracker. These features are placed at every pixel location. The sum of the pixels which lie in the darker region is subtracted from the sum of those in the lighter area. (b) From left to right. Original image at t and at t+1. The absolute difference between the two images is next. The following images show the difference between an image at t and t+1 but shifted by (respectively) up, down, left and right one pixel. Adapted from [Viola et al., 2003].

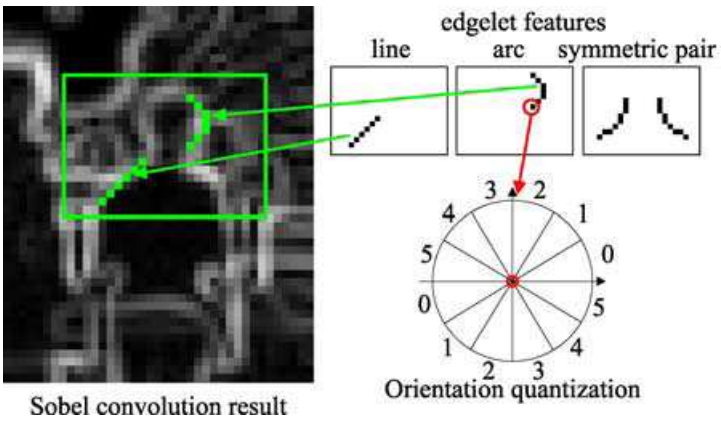


Figure 2.22: Edgelet features are extracted from a edge image. They are further quantised based upon their orientation. Adapted from [Wu and Nevatia, 2007]

tection up into separate parts is realised when attempting to identify and track heavily occluded pedestrians. Senior [Senior, 2002, Senior et al., 2001] also attempted to deal with occlusions by combining a person colour model with an occlusion model. The occlusion model helps to keep track of people when it deems parts of them to be occluded by reducing the weighting placed upon these pixels when evaluating the likelihood of a tracked person (and also conversely increase the likelihood on good pixels). Other methods for taking into account occlusions (although on this occasion static) are given in [Greenhill et al., 2004] where semantic knowledge of the scene is embedded in the tracking process.

Zhao and Nevadia [Zhao and Nevatia, 2003] again use a parts based model using edge features. However they use a hypothesised pose model and attempt to fit it to the data using incremental computation of the likelihood of such a model fitting the data. Sidenbladh and Black [Sidenbladh and Black, 2001] also use a Bayesian approach to learn the features for a hypothesised human model.

Dynamical Models

Many of the ideas shown in the preceding sections require a method to enforce temporal continuity of an identification or to direct the model where to look for the target in relation to the last frame.

The Kalman filter [Kalman, 1960] has been used within many tracking systems. The moving object's location is represented at time t by $\mathbf{x}_t$. The change in state is modeled by the dynamic equation given in equation (2.10).

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t \tag{2.10}$$

Here the state (normally the location in tracking applications) at time t is dependent upon the previous state at $t-1$. The state transition model $\mathbf{F}$ is applied to the state at the previous timestep ($\mathbf{x}_{t-1}$). The control input model $\mathbf{B}_t$ at time t is applied to the control parameters $\mathbf{u}_t$. The final term $\mathbf{w}_t$ represents additional white noise which is assumed to be from a zero mean multivariate Gaussian distribution with covariance $\mathbf{Q}_t$, giving $\mathbf{w}_t \backsim \mathcal{N}(0, \mathbf{Q}_t)$, where $\mathcal{N}$ is the normal (Gaussian) distribution. At time t an observation ($\mathbf{z}_t$) of the true state $\mathbf{x}_t$ is made according to equation 2.11.

$$\mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t \tag{2.11}$$

The observation model which maps the the true state space into the observed space is given by $\mathbf{H}_t$. The term $\mathbf{v}_t$ is modeled by a zero mean Gaussian distribution with covariance $\mathbf{R}_k$. The initial state and the noise vectors at each state are assumed to be mutually independent of one another.

The current state of a Kalman filter can be represented by two variables, $\hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t}}$ is the estimate of the state $\mathbf{x}$ at time t whilst the error covariance is given by $\mathbf{P}_{t|t}$. The error covariance is a measure of the estimation of the accuracy of the state estimate. The predictions for the estimated state and the error covariance are given in equations (2.12) and (2.13) respectively.

$$\hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}} = \mathbf{F}_t\mathbf{x}_{t-1|t-1} + \mathbf{B}_t\mathbf{u}_t \tag{2.12}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^T + \mathbf{Q}_t \tag{2.13}$$

The filter contains two main steps, the prediction step, as noted above, and an update step where the parameters of the model are updated.

The updated state $\hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t}}$ and covariance $\mathbf{P}_{t|t}$ estimate are given as (with I being the identity matrix) :

$$\hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t}} = \hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}} + \mathbf{K}_t\tilde{\mathbf{y}}_t$$

$$\mathbf{P}_{\mathbf{t}|\mathbf{t}} = (I - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_{t|t-1}$$

With $\mathbf{K}_t$, $\mathbf{S}_t$and $\tilde{\mathbf{y}}_t$ being defined as :

$$\tilde{\mathbf{y}}_t = \mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}} \quad (2.14)$$

$$\mathbf{S}_t = \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T\mathbf{R}_t \quad (2.15)$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T\mathbf{S}_t^{-1} \quad (2.16)$$

In the equations above the measurement residual is given by equation (2.14) whilst the covariance of this residual is given in equation (2.15). The Kalman gain is shown in equation (2.16).

Despite the Kalman filters wide applicability to vision problems and its ability to provide an optimal linear estimate it has shortcomings within tracking applications due to the non Gaussian distributed nature of tracked objects. Robertson [Robertson, 2006] showed that the filter has shortcomings when modeling behaviour which switches states rapidly. For instance when someone stops and turns round the model is unable to adapt quickly enough to follow the movement.

In an attempt to overcome problems of the filter's assumption of a single Gaussian distribution Isard and Blake [Isard and Blake, 1996, 1998] introduced a particle filter (which they termed condensation) which can model multimodal distributions. The conditional state density $p(X_t|Z_t)$ at time t is represented by a set of particles $\left\{s_t^{(n)} : n = 1,..,N\right\}$ with associated weights $\left\{\pi_t^{(n)} : n = 1,..,N\right\}$ denoting the sampling probability of the particle. Weights are used to denote the importance of the sample [Isard and Blake, 1998]. For each pair $\left(\pi_t^{(n)}, s_t^{(n)}\right)$ a cumulative weight $c_t^{(n)}$ is stored where $c_t^{(N)} = 1$. At each timestep new samples are drawn from the previous times samples. Importance sampling was used by Isard and Blake [Isard and Blake, 1998] but other sampling methods have been proposed by MacKay [MacKay, 2003] and Doucet *et al.* [Doucet et al., 2001]. Importance sampling is described below:

- **Selection**. Select N random samples $\hat{s_t^{(n)}}$ from the set of all samples $\mathbf{S}_{t-1}$. The chance that a sample is selected is weighted by it importance (ie higher weighted samples have more chance of being selected.)

- **Prediction**. For each of the selected samples $\hat{s}_t^{(n)}$, generate a new sample $s_t^{(n)} = f\left(\hat{s}_t^{(n)}, W_t^{(n)}\right)$. The function f is a non negative function, often it represents a motion model describing the movement (or lack thereof) of the tracked object or expected features of the object. $W_t^{(n)}$ is a zero mean Gaussian error.

- **Correction**. Weights $\pi_t^{(n)}$ corresponding to the new samples $s_t^{(n)}$ are computed

using the measurements z_t by $\pi_t^{(n)} = p(z_t|x_t = s_t^{(n)})$, where $p(.)$can be modeled as a Gaussian density.

For tracking purposes these samples have encoded colour [Rowe et al., 2007], point features [Fan et al., 2006], contours [Isard and Blake, 1996] and kinematic models [Sidenbladh et al., 2000].

Other multiple hypothesis models include the joint probabilistic data association filter, used by Nguyen *et al.* [Nguyen et al., 2006]. Models with inbuilt logical constraints to guide tracking have been used by [Bennett et al., 2004].

2.5 Conclusion

Interaction such as Oliver used very simple interactions, The features also had a high degree of redundancy so it is questionable what the extra coupling of the model achieves. Others such as Khan enforce a rigid structure upon the interaction types which (as demonstrated in Khan's paper) is not applicable to the general case. Many others have successfully employed a pre defined script which has proved to be both successful in terms of classifying situations and also has been implemented in commercial systems (for example Object Video). However there is little work on identifying interacting individuals in more complex surveillance situations for larger scale datasets.

The ability to learn a representation is useful in that it would allow one to train a system by simply showing examples of the desired behaviour instead of relying on an expert to produce an exhaustive list of how a behaviour should be defined. Of course pre-defined behaviour evaluation has proved to be successful but within this thesis we seek to advance state of the art in detecting interaction between individuals without the aid of a pre-defined script. Previous work has demonstrated this ability on limited and small datasets, here we shall make use of real data and seek to improve upon previous methods, most notably those of Oliver *et al. [Oliver et al., 2000a].*

There is also a gap in the current work with regard to detecting dangerous situations before they happen. Specifically the ability to detect behaviour which can lead to a fight has not been previously investigated by computer vision researchers. This question has been investigated by physiologists such as Troscianko [Troscianko et al., 2004] who demonstrated that it was possible for humans to perform this task.

The ability of a computer to track a person was discussed and demonstrated that it is feasible in section 2.4. The ability to track an individual is central to many approaches

and this ability is assumed throughout this thesis.

Chapter 3

Detection of Coordinated Motion

Parts of this chapter have appeared in Blunsden et al. Recognition of coordinated multi agent activities, the individual vs the group, 2006 [Blunsden et al., 2006]

3.1 Introduction

This chapter deals with the problem of identifying coordinated, well structured and restricted team actions. Here we concentrate on the domain of sports. In particular the focus is on Football and European handball. Coordinated motion is defined as that displayed in organised games and which typically have a well defined structure. The investigation of group activities within a comparatively restricted domain of sports activities will form a starting point from which the more general case of unstructured interaction will be investigated.

This chapter deals with those team interactions which have an agreed classification[1] but where there is not necessarily a rigid structure. For example here we concentrate upon the games of football and handball. Structures in these games are more loosely defined than those of for example, American football where the action often follows a set plan. The aim of this chapter is to investigate how to model such team activities. For restricted cases such as team sports the domain provides a good testing ground within a well restricted and controlled setting. Here we will perform a comparison between individually modelling each player within the game and taking team information as a whole to model the game state. In the following sections both methods will be explained and their results presented on two different datasets. First a brief review of

[1] By agreed classification it is meant that an expert coach will have labeled the particular instance of that behaviour.

previous work will be given.

The investigation within this chapter will focus on classifying loosely structured team interactions. By this it is meant that there are agreed terms for what is happening within a game but there is not necessarily a template or state model which can accurately describe the action. A contrast to this is the work of Intillie and Bobick [Intille and Bobick, 1999] and also (separately) Perse *et al.* [Perse et al., 2007, 2006] where pre-specified team actions were available. This is covered in more detail in section 3.4.

Within the domain of sports analysis some authors have sought to classify team behaviour from broadcast footage, particular examples being Assglag *et al.* [Jurgen Assfalg, 2003] and Ekin *et al.* [Ekin et al., 2003] who incorporate shot information into the classification framework. Whilst this is necessary when attempting to automate the highlight generation process such methods have an implied domain expert (the cameraman/editor) watching the sporting event meaning that the data already contains knowledge of the game state. A particular example of this could be fast camera motion when a player is going to score a goal, or slow motion replays to illustrate interesting events. Here we concentrate upon understanding what is going on within the game without the potential bias of an expert viewer.

3.2 The Problem

Within this chapter we want to address the problem of classifying team activity. Team activity will mean the particular style or structure that the team is adopting when playing the game. These styles will be identifiable by an expert in the game. Classification of team game activities would be useful for automatically finding examples of a particular play or style from many games. It would be useful for coaches and also potentially fans of team games who would be able to specify what it is they want to watch.

3.3 Contribution

This chapter presents two competing approaches to classification of team game activities. We do not require a pre-specified template to identify activities and instead use examples to learn activities. We present a faster and more accurate method using whole team activities as contrasted with the previous state of the art as given by Howard and Jebara [Howard and Jebara, 2004]. We also highlight issues with per player modelling and limited video data.

3.4 Previous Work

In fitting with the theme of modelling the individual vs modelling the whole this section will broadly divide previous research into two sections. First research where individuals are modelled and then aggregated to give a overall team state is described. Then a review of previous research using features derived from of the whole teams features is given.

3.4.1 Individual modelling

The work most closely resembling the domain of interest here is that of Intillie and Bobick [Intille and Bobick, 1999]. The goal was to classify pre-defined plays within the game of American football. Intillie and Bobick used predefined Bayesian networks to evaluate the likelihood of an observed play. Based on visual evidence a goal network was used to determine the belief that certain actions had been committed. This provides input into a multiagent network combining many agent's visual goal networks to compute the likelihood that a particular play was being observed. Perse *et al.* [Perse et al., 2007, 2006] take a similar template based approach. Both of these approaches use a template based description of a particular scenario. Within the context of the game such sequences of moves are well defined and are rehearsed so it is possible to recognise a specific sequence of moves that the team is attempting to execute.

In addition to the static Bayesian network architectures used by [Intille and Bobick, 1999] dynamic Bayesian networks have also been applied to multiagent problems. In [Oliver et al., 2000b] Oliver *et al.* used coupled hidden Markov models to model two person interactions. Again separate models are trained for each class of interaction. However there was no explicit apriori domain knowledge within the models. The likelihoods of observing a given interaction was determined from examples. Coupled models have also been employed by Xiang and Gong [Gong and Xiang, 2003b] who modeled vehicle interactions on an airport loading bay. All possible couplings of the model were enumerated and evaluated using the Bayesian information criterion [Schwarz, 1978] to determine the best connection architecture.

More recently Howard and Jebara [Howard and Jebara, 2004] use a tree architecture to aggregate several persons trajectories (again within the domain of American football) into a hierarchical Markov process describing the team activity. Although they only present limited results the approach presents a plausible interpretation of the structure within the game. Each player is modeled, then the team, then the game as

a whole. Others such as Zhang *et al.* [Zhang et al., 2000] have also used an aggregating approach for group action clustering in meetings. However, in this instance, a hierarchical architecture is not as explicit.

3.4.2 Team modelling

Features derived from the overall team have been previously investigated by Taki *et al.* [Taki et al., 1998] and Fujimura and Sugihara [Fujimura and Sugihara, 2005]. Both approaches analyse how the team as a whole is doing within the context of a football game. Fujimura and Sugihara introduce the idea of a Voronoi diagram tessellated on the basis of how quickly a player can get to a particular location. This region is referred to as a dominant region for the player. Certain regions are given more weight (eg regions close to goal). Separately Needham [Needham, 2003] modeled the positions of players within a football game but did not publish any attempt to analyse the results or categorise team performance.

Central to the team modelling is the idea of a dominant region. The dominant region first starts with the calculation of a player's sphere of influence. This is defined by the region that a particular player can reach before any other player. The individual k at time t is represented as $p_k^t \in \mathbf{P}^t$, where $\mathbf{P} = \{p_1, p_2, .., p_n\}$is the set of all player's positions. The dominant region of this player is given in equation 3.1 as:

$$D(p_k^t) = \left\{x | t_s(x, p_k^t) \leq t_s\left(x, p_m^t\right), \text{ for } m \neq k\right\} \tag{3.1}$$

The dominant region $D(p_k^t)$ is in this case bounded by the game area (the football field) with the position x referring to the two dimensional (in this case) world coordinate. The shortest time t_s is the time necessary for an individual k to move from their current position p_k^t to the point indicated by x. The function $D(p_k^t)$ defines a region where the player p_k^t can occupy before any other player (ie they can reach that point before any other player). To determine this the current motion vector of player p_k^t is combined with the positional information to determine the time required to reach the point. The acceleration and speed of a player are approximated based on the average individual. A graphical representation of this can be shown in figure 3.1.

The calculation of the dominant region is then used in further processing when evaluating the team as a whole. The team is evaluated on space making, ball passing and the amount of pressure they put upon the opposing team. To evaluate the ability of the team to make space the entire team's dominant region throughout time is measured.

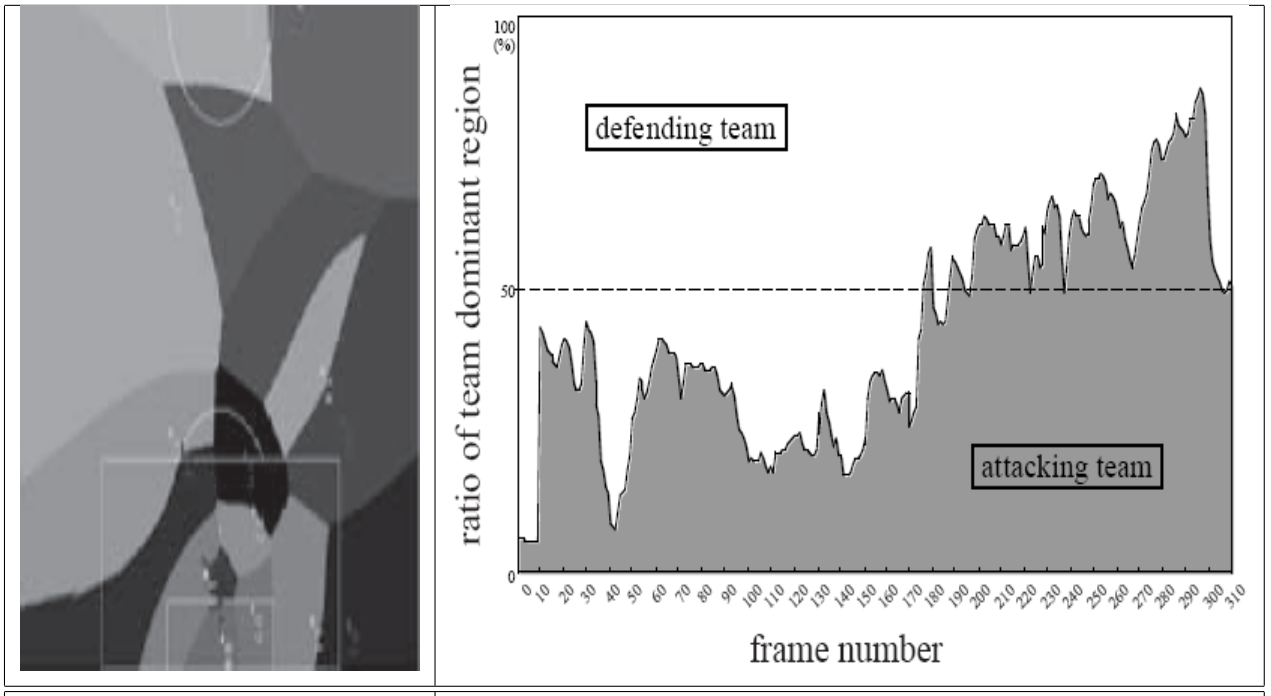


Figure 3.1: (a) Example of several players dominant regions overlaid upon one half of a football field. (b) The variation through time of the ratio of the attacking team. In this example the attacking teams cooperative movement is superior from around frame 180 onwards. During the latter half of this sequence to dominant team scored a goal. Adapted from [Taki et al., 1998]

The amount of pressure is determined by spacial factors such as density of players, sphere of influence, distance from goal and distance from the ball.

3.4.2.1 Summary

Previous work has made use of template based approaches [Intille and Bobick, 1999], [Perse et al., 2007, 2006] with considerable success. However here we are interested in investigating how a method can learn when only presented with positive examples. As well as investigating a different area from previous work such an approach could make it possible for non-experts to define how they would like to see the game classified. Such an approach would be useful for watching highlights of games as determined by the viewers themselves.

We are also interested in comparing the approach of modelling each individual vs modelling the whole team. We make use of Howard and Jebara's [Howard and Jebara, 2004] method for modelling individual players. This method is one of the most recent developments in team classification. They also state that "*A naive approach to modeling our data is to stack or concatenate each player's time series into a single multivariate series*". In this chapter we test the validity of this statement.

In order to make it as fair a comparison as possible we make use of features which can be commonly used for both individual and team modelling. For this reason we do not follow Taki *et al.* [Taki et al., 1998] or Fujimura and Sugihara's [Fujimura and Sugihara, 2005] approaches to calculating a feature set who use many whole team features. For example it would be unclear how to divide up the mean team position between individual players. They also use such metrics for evaluating how the team is doing, not necessarily what it is doing.

3.5 Classification Approach

This section details the approach to the classification problem. First the extracted features are introduced in section 3.5.1. The classification algorithms are then detailed in section 3.5.3.

3.5.1 Extracted Features

In addition to the original $x_{n,t}$ and $y_{n,t}$ court position provided in the data set (for the n^{th} player at the t^{th} timestep), three additional features were also calculated. A speed

feature ($s_{n,t}$) was calculated for every player of every frame. The speed ($s_{n,t}$) of a player was calculated over the past w frames (equation 3.4). Such a feature helps to distinguish between fast and slow breaks. Here a temporal window size (w) is used due to the high sample rate (25 fps) resulting in very small movements between consecutive frames. If the change in position was calculated based solely upon the previous frame then frequently that change would be either zero or very close to zero. Coupled with the error in the associated tracking process most of this movement would be the result of noise.

The directional change of a player is also encoded in the feature vector with the direction of change in x ($c^x_{n,t}$ equation 3.2) and y ($c^y_{n,t}$ equation 3.3) being calculated, again over the past w frames. The window size (w) for calculating the change in position was empirically set to 25 frames (1 second) throughout all experiments. To prevent features from one complete activity sequence overlapping with those from another class (ie immediately after transitions) the first w frames from the sequence are removed. In addition to removing any overlap between features this also helps with eliminating the transitional period where one activity turns into another.

$$c^x_{n,t} = sign(x_{n,t-w} - x_{n,t}) \tag{3.2}$$

$$c^y_{n,t} = sign(y_{n,t-w} - y_{n,t}) \tag{3.3}$$

$$s_{n,t} = \|(x_{n,t-w}, y_{n,t-w}) - (x_{n,t}, y_{n,t})\| \tag{3.4}$$

This gives a final feature vector for the n^{th} player at the t^{th} timestep as given by equation (3.5).

$$\mathbf{p}_{n,t} = \left[x_{n,t}, y_{n,t}, c^x_{n,t}, c^y_{n,t}, s_{n,t}\right]^T \tag{3.5}$$

3.5.2 Classification

Each test/training sample was taken from the original data annotations and consists of a sequence of contiguous frames containing the calculated features as described in section 3.5.1. These sequences are taken directly from the annotated sequences. Within this classification scheme the minimum sample length varied due to the length of time an activity was being performed. For the training phase only the training samples were used and for the testing phase only the unseen test samples were used. The training

and test samples did not overlap. The individual and group classification methods are now detailed and results are presented in section 3.6.

3.5.3 Methods

3.5.3.1 Individual modelling approach

Within this section a method for representation and classification of team action is given. Here each individual is modeled as having their own independent Markovian dynamics with each individual then aggregated into an overall team model. Each individual model is then used as input into a higher level process which models the current activity of the team. The approach implemented here is that of Howard and Jebara [Howard and Jebara, 2004]. This approach is chosen to contrast to the global view where all the players are considered as forming the input signal (section 3.5.3.5). The dynamical systems tree (DST) was chosen as it does not require an explicit domain model (cf. Intille and Bobick [Intille and Bobick, 1999]) and is capable of generating a class label for a given sequence (cf. Xiang and Gong [Gong and Xiang, 2003b]). This hierarchical model also represents the problem well in that we hypothesize that each player forms input into a higher level process representing the team activity. The dynamical systems tree model is explained in section 3.5.3.2.

3.5.3.2 Dynamical Systems Tree

The dynamical systems tree was introduced by Howard and Jebara [Howard and Jebara, 2004]. The model consists of several independent switched linear dynamical systems (SLDS). Within the context of the complete tree these processes are referred to as leaf-processes. Each SLDS subsumes both a hidden Markov model and a standard (ie non switched) linear dynamical system. For this reason the explanation of the DST shall deal with SLDS, however it is general to all three methods.

Within the SLDS the transitions between hidden states are Gaussian. The and emissions are continuous hidden states again with a Gaussian distribution.

In order to couple the independent SLDSs together a Markovian aggregating process is used. Each of these aggregating processes have the leaf processes as their children. In the general model an aggregating process can have another aggregating process as its parent, however here only one aggregating process is used. The process's role is to combine all the player's processes into an overall process which describes the team's activities. The structure of the model is such that each player is modeled with

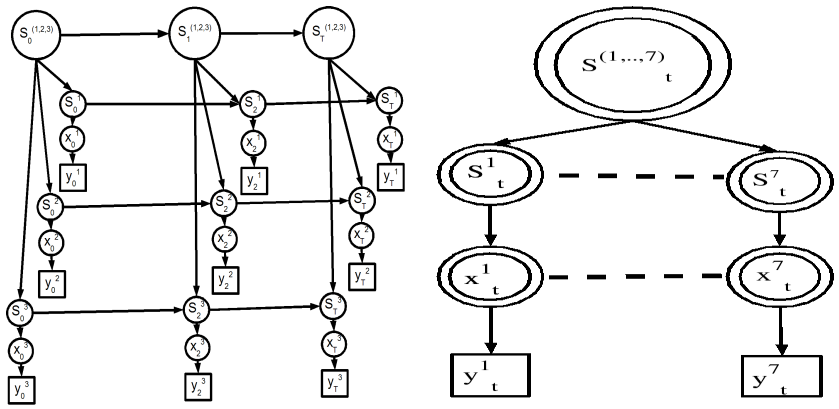


Figure 3.2: The structure of the dynamical systems tree. Unfolded on the left whilst the diagram on the right shows a compact representation of the model. A circle within a circle represents repetition throughout time.

their own SLDS while the overall team is modeled with a single aggregating process. This structure is given in figure 3.2

3.5.3.3 Probability distribution

The DST consists of one or more aggregating process and leaf processes arranged in a hierarchical manner. An aggregating process is a Markov chain which has at most one parent process. For the problem represented here there is one aggregating process which represents the group activity. The states of the aggregating (a) process states are represented by $s^a = \{s_0^a, .., s_T^a\}$, where T represents the maximum possible time. The activity aggregating process, without any parents, has the conditional distribution:

$$p(s^a) = p(s_0^a) \prod_{t=1}^{T} p(s_t^a | s_{t-1}^a) \tag{3.6}$$

Each player is modeled as a switching linear dynamical system (SLDS) and is referred to as a leaf process within this framework. A leaf process has a parent aggregating process and has the following conditional distribution:

$$\begin{aligned} p(\mathbf{s}^i, \mathbf{x}^i, \mathbf{y}^i | \mathbf{s}^{\pi(i)}) = {} & p(s_0^i | s^{\pi(i)}) p(x_0^i | s_0^i) p(y_0^i | x_0^i) \\ & \times \prod_{t=1}^{T} p(s_t^i | s_{t-1}^i, s_t^{\pi(i)}) p(x_t^i | x_{t-1}^i, s_t^i) p(y_t^i | x_t^i) \end{aligned} \tag{3.7}$$

In this conditional distribution $\mathbf{s}^i = \{s_0^i, .., s_T^i\}$ represents the i^{th} leaf process's discrete Markovian hidden state. $\mathbf{x}^i = \{x_0^i, .., x_T^i\}$ is the continuous Markovian hidden

state, with $\mathbf{y}_i = \{y_0^i, .., y_T^i\}$ being the observations of the i^{th} leaf process. The leaf processes parent process is given by $\pi(i)$ with discrete hidden states $\mathbf{s}^{\pi(i)} = \{s_0^{\pi(i)}, .., s_T^{\pi(i)}\}$.

This gives the conditional distribution over all variables in the DST with $\mathcal{A}$ aggregating processes and $\mathcal{L}$ leaf process's as:

$$P(\mathcal{S}, \mathcal{X}, \mathcal{Y}) = \prod_{a \in \mathcal{A}} p(\mathbf{s}^a) \prod_{i \in \mathcal{L}} p(\mathbf{s}^i, \mathbf{x}^i, \mathbf{y}^i | \mathbf{s}^{\pi(i)}) \tag{3.8}$$

where $\mathcal{S}, \mathcal{X}, \mathcal{Y}$ represents the discrete hidden, continuous hidden and emission variables, respectively.

The estimation of the parameters of the DST model are given in the appendix section B.1.

3.5.3.4 Is the model appropriate to the task?

The DST model as used in the remainder of this chapter is shown in figure 3.2. Each player is modelled as a first order switching linear dynamical system (as shown in figure 3.2). The hidden state of each of the player models are aggregated by the aggregator state. The role of the aggregator state is to model the state of the team as a whole based upon each player's state. The main assumption behind this model is that the team's state can be modelled as a function of the players states.

This model is appropriate to the task as each player within the team is individually fulfilling some function but this function is contributing towards the overall teams behaviour (as modelled by the aggregation variable). It is hoped that by modelling the overall state of the team by aggregating each individual through time an accurate model can be formed.

Due to the high complexity of the model an approximation of the probability distribution is used. The uncoupling of absolute links between higher and lower levels for each player is performed to allow the algorithm to become tractable. More details of this procedure are provided in the appendix (B.1) and also in Howard and Jebara [2004].

The main advantage of modelling individual players and combining their individual states into a single function is the high similarity between the model and the game being modelled. Compare this to a coupled hidden Markov model which could be set up to link all players to one another. Assuming this could be done (it would be infeasible at present for games with more than 8-10 players due to memory requirements without some form of approximation) there is no explicit game state in such a model,

which is the very thing we are trying to model. The DST provides an explicit "team state" in the form of the aggregator variable.

3.5.3.5 Group modelling

In contrast to modelling each individual player the input vector is now made up of *all* the players in the team. That is the input signal per frame is given by equation (3.9). For the example of handball where there are 7 players each with 5 attributes this forms a 35 dimensional input vector per frame.

$$\mathbf{q}_t = \left[\mathbf{p_{1,t}}, .., \mathbf{p_{N,t}}\right] \tag{3.9}$$

Here $\mathbf{p}_{n,t}$ is as defined in equation (3.5) and N is the size of the team (where N=7 for handball and N=11 for football). A support vector machine (SVM) classifier [Scholkopf, 2000] is then trained upon this data. Partitioning of training and testing data followed the same convention as in the individual case. The SVM classifier is briefly reviewed here before describing its application to the sports dataset. The SVM classifier was chosen after examining the eigenvector projections of the data (figure 3.9). The data is well separated in the space per class but not clustered around any obvious points. The decision boundaries between the classes are also non-linear, for this reason a SVM classification scheme was deemed an appropriate classifier to use.

3.5.3.6 SVM

A SVM classifier seeks to find a separating (hyper) plane such that the distance between the plane and each data point is maximised. This distance is given by the margin which is defined as being the minimum perpendicular distance from the plane to a data point. A point is then classified by the discriminant function:

$$f(\mathbf{x}) = sgn(\sum_{i}^{N} y_i \lambda_i . k(\mathbf{x}, \mathbf{x}_i) + b) \tag{3.10}$$

Within this equation $\mathbf{x}_i$ is the i^{th} datapoint with $y_i \in \{+1, -1\}$ being the corresponding class label and λ_i the Lagrangian term associated with the i^{th} data point. When the data point ($\mathbf{x}_i$) has a non-zero Lagrangian (λ_i) value it is considered a support vector (ie contributing to the hyperplane separating the classes). The term b is a constant scalar representing the offset from the origin. The kernel function is represented by k. Throughout these experiments a Gaussian kernel function is used. The problem is to maximise :

$$W(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \lambda_i \lambda_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{3.11}$$

subject to $0 \leq \lambda_i \leq C, i = 1, ..., N$ and $\sum_{i=1}^{N} \lambda_i y_i = 0$

The constant C places an upper bound on the Lagrange parameters, thereby limiting the influence of individual patterns. In experiments presented here C was set to 10 as good classification performance was obtained with this value. Although using pairwise classification strategies have been shown to be preferable in some situations [Duan and Keerthi, 2003] the one against all method proved to work well for this problem and so is preferred.

The SVM classifier is trained using individual points from $\mathbf{q}_t$ (equation (3.9)). Each frame in the training set was assigned an activity label corresponding to the class of the complete sequence. This effectively removed the temporal ordering of the data (which is an area of future research). The publicly available Torch libraries [Collobert et al., 2002] were used to compute the results.

To classify a test sequence each point in that sequence is individually labeled by the SVM classifier. This produces a vector of tokens for each (variable length) sample ($\mathcal{L} = \left[l_1 = c_1, .., l_t = c_1 .., l_{samplen} = c_i\right]$). Here l_t represents the label at time t and is an element from the set of all possible classes as determined by the SVM classifier. To classify the entire sample from the labeled vector the most frequent label in this labelling is taken to be the label of the whole sample.

3.6 Experiments

This section details the experiments performed to evaluate the methods described in the previous sections.

3.6.1 Experimental Setup

The experiments were set up to compare the accuracy of the individual modelling approach from section 3.5.3.1 and the group approach as described in section 3.5.3.5. The experiments were set up to classify whole sequences of actions which have been labeled by an expert. The data is described in section 3.7 and consists of positions of all players performing a specified game action.

In order to test the methods described a training and test set were created. In order to fairly compare both the group and individual approaches to classification we shall

randomly permute samples to be included in the training and testing sample.

A question we address in this chapter is how does the amount of time you view a particular sequence affect your ability to classify it? To address this question the input required to classify a single frame is changed from one consisting of 1 frame in length to 150 frames. The maximum size of window was set at 150 as above this the number of available sequences reduces dramatically and there are no examples of some classes beyond this length. Most sequences range between 50 and 200 frames in length.

The two methods were compared across a range of window sizes. Such window sizes corresponded to the information which was available in order to make a decision on the classification of that frame. A frame is classified using information from previous frames to decide upon the class of the current frame. For example to classify a frame using a window size of 3 the test data required will consist of :

$$\mathbf{q}_t = \begin{bmatrix} \mathbf{p_{1,t}}, .., \mathbf{p_{N,t}} \\ \mathbf{p_{1,t-1}}, .., \mathbf{p_{N,t-1}} \\ \mathbf{p_{1,t-2}}, .., \mathbf{p_{N,t-2}} \end{bmatrix} \tag{3.12}$$

This process is shown in figure 3.3.

3.7 Results on Data Sets

Here results for two different datasets are introduced. First synthetic data is used in order to show that the previously described methods are feasible. We then move on to substantially more complex dataset and attempt classification of data obtained from a game of handball.

3.7.1 Synthetic Data

For initial testing of the suggested method a synthetic dataset was used. The synthetic data was gathered from the TricTrac dataset [multitel, 2006] which provides a multi camera viewpoints for showing 3 separate scenarios within a football match. The first shows a team attacking with a goal being scored. The second shows one team attacking and shooting the ball at the goal without a goal being scored and the final case shows the team successfully defending an attack with no shot or goal resulting.

Because the data was generated synthetically the tracking positions are available and are highly accurate. The position of all 11 players from one team are given in world

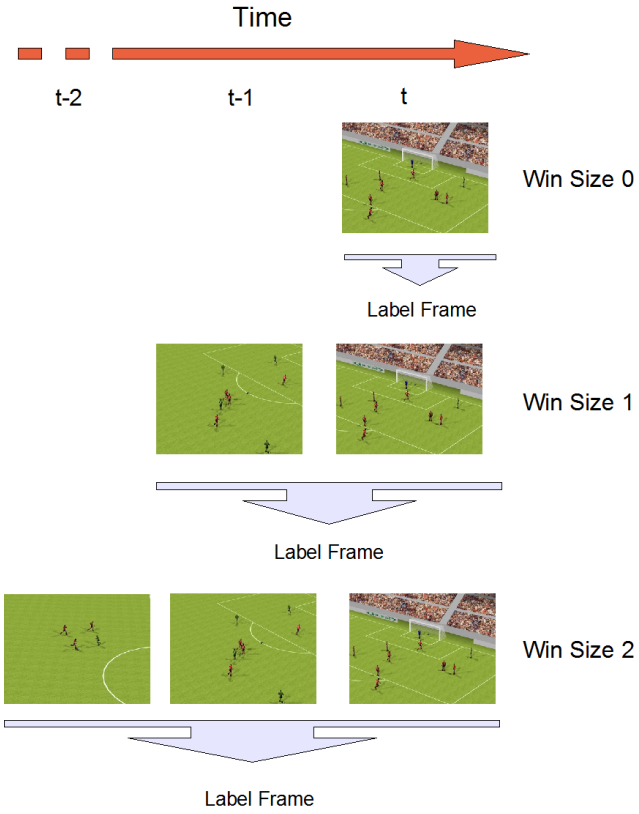
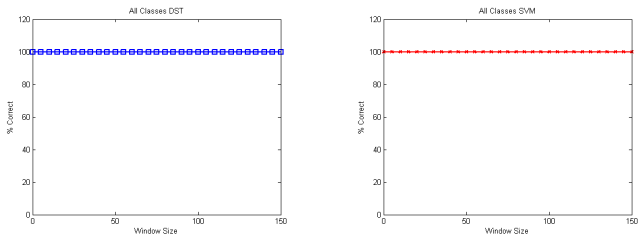


Figure 3.3: Data required to label a frame.

Figure 3.4: Original video data from the TriacTrac dataset. This sequence is from scenario 1 where a goal is scored.



(a) (b)

Figure 3.5: Results using the TricTrac dataset over varying window sizes. (a) Results using the dynamical systems tree to classify the TriacTrac dataset over varying window sizes. (b) Results using support vector machines to classify the TriacTrac dataset over varying window sizes.

co-ordinates (even though the videos show multiple views). The world co-ordinates are used as input throughout the experiments.

The data comprises of a total of 10,162 individual frames. For each frame the world co-ordinates of each individual player are given. The length of each sequence ranged from 682 frames to 971 frames. Results are presented using a varying window size.

3.7.2 Results

This dataset does not seem to provide enough of a challenge for a classifier and indeed the situations it depicts are all very similar (although they vary in length). This factor results in perfect classification by both methods. These results are shown in figure 3.5 for both the SVM and DST methods. The reason that it is included in this thesis is to illustrate that the methods are feasible to classify the differing situations and that results on synthetic examples should be backed up with real examples. This is the focus of the next section where real video data from a real game is used to test the methods in a more realistic and challenging scenario.

3.7.3 CVBASE Dataset

The data set used in this section is from the publicly available CVBASE dataset [CVb, 2006]. Only the handball sequences are used in the methods and results presented here. The handball dataset consists of 3 separate video cameras recording a 10 minute long game. Court coordinates of each of the players for one team throughout this sequence are available (first 1,000 frames are shown in figure 3.6 (a)). The activity the whole

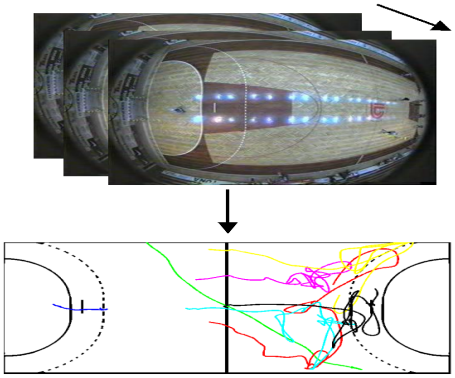
team is engaged in and the starting and end times of the activity is also annotated. In total there are 15,000 annotated frames each giving the positions of all 7 players (unless the player was temporarily out of shot). From these frames there are around 60 complete sequences ranging from a few frames in length to many hundreds.

The timeout class from the original dataset was removed as this is not regarded as a coordinated activity. In addition some classes have been merged together. This step was primarily taken due to the high similarity between the activities themselves. The classes offense against set-up defense, setting up an offense' (nfpn) and 'offense against set-up defense, ending an offense' (nfan) were merged. These classes are highly similar in description and in appearance and require extra information about the game (ie the ability to detect an attempt on goal) to distinguish them. The classes 'defense, basic defense, against preparation of an offense' (obz) and 'defense, basic defense against ending offense'(obg) were merged for similar reasons. The classes 'defense returning' (ovpp) and 'defense slowly returning' (ovpc) were merged as the actual speed difference distinguishing each class is not well defined. These activities may have looked different if information on the other team was also available, particularly in distinguishing between trying to stop a fast break. This gives 5 final classes as shown in figure 3.6 (b).

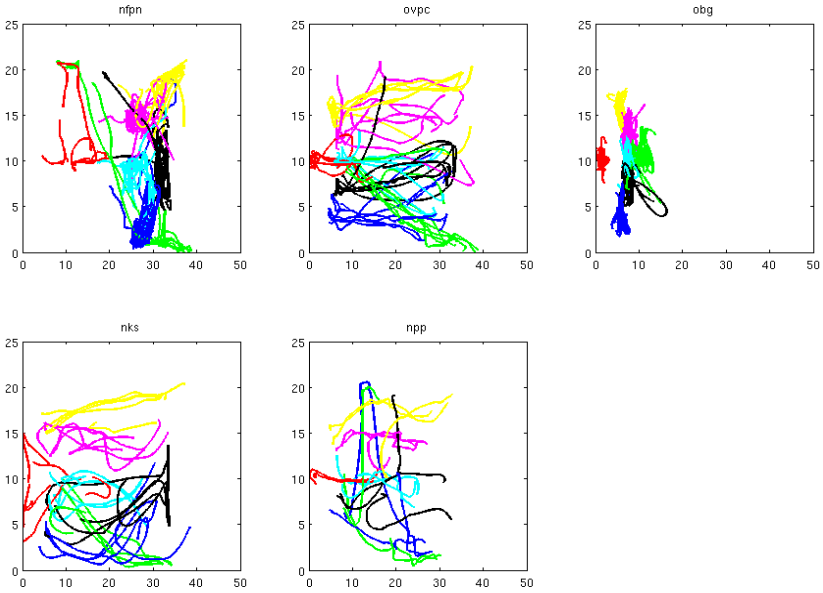
3.7.4 Results on Sequences

Here results for both the DST and SVM methods are given. The overall results are given in figure 3.7. Throughout the experiments the window size was varied from 1 to 150 frames. For this particular case the overall results improve slightly when using a SVM classifier with an increased window size. The results when using the DST method are more erratic. Although the classification rates were obtained over multiple runs (as was the case for all experiments) it is visible that there is greater variation in the average performance of the DST classifier. However the variance for smaller window sizes is actually less than that of the DST. This would suggest that the learned model is more generaliseable. Once the window size used has exceeded 100 frames performance begins to drop. The results using a DST are not as consistent as those when using a SVM. This could be down to the random initialization required for the DST coupled with the larger number of parameters which are required.

The per-class results are given in figure 3.8. Per-class results demonstrate the variability in performance that varying window sizes can make. For many classes and in-



(a)



(b)

Figure 3.6: (a) The first 1000 frames from the video sequence and the associated player positions as given in court coordinates. (b) Positional information of each player (different colour) plotted in court co-ordinates. The classes are: nfpn - offence against set-up defense , ovpc - defense, returning, obg - defense, basic defense, nks - offense, fast break, npp - offense, slowly going into offense.

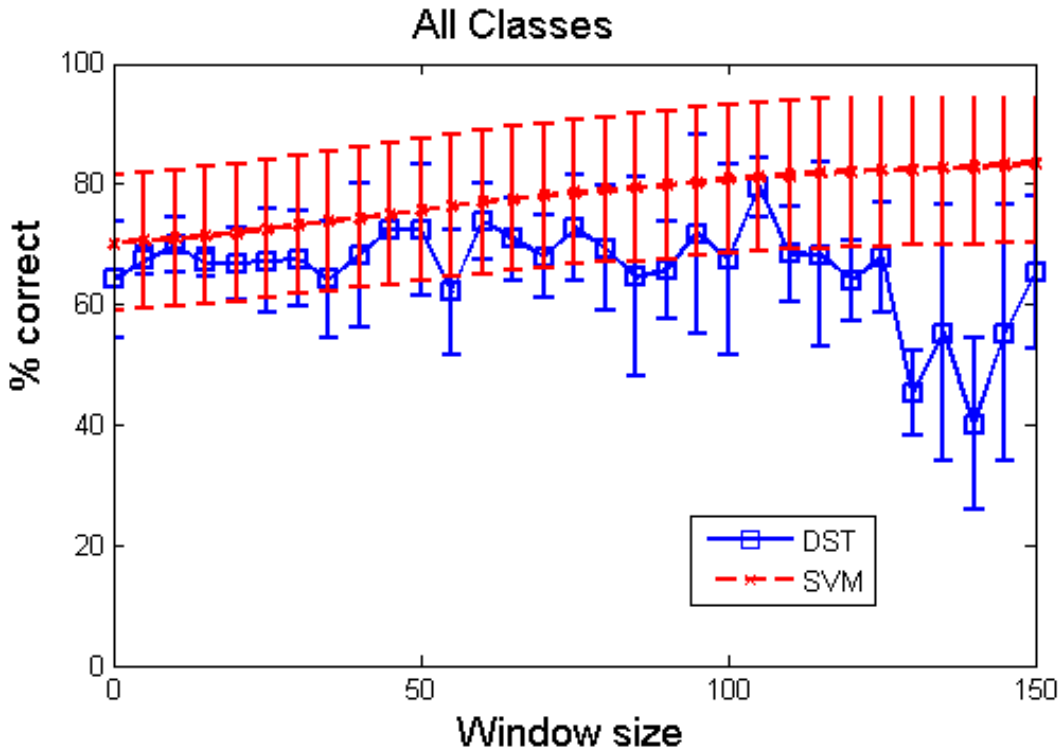
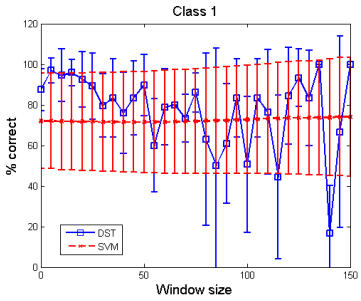


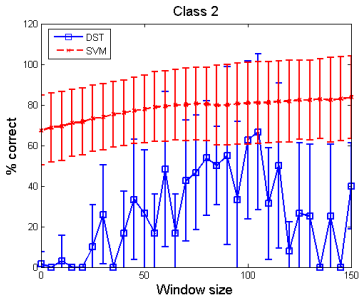
Figure 3.7: Overall results from all classes

deed overall the DST classifier gives a less consistent classification performance then the SVM method. This is mainly down to the increase in parameters and the random initialisation used by the EM update process. Results when classifying class 1 display this contrast between the two classifiers. For class 2 the SVM consistently outperforms the DST. There is a significant drop in performance by the DST after 100 frames whereby the accuracy drops significantly. For the SVM there is a slow improvement in classification accuracy until around a window size of 75 frames at which point the improvement stops.

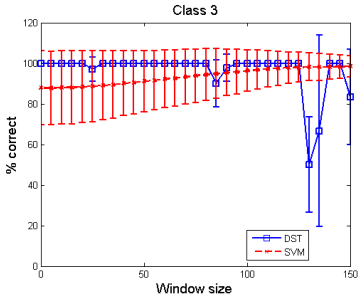
When looking at the results for class 3 it is shown that for almost all window sizes the DST method slightly outperforms the SVM method. As window size increases this gap becomes smaller. The results for class 4 are perhaps the most interesting and show how the window size can relate to the underlying data. There is drastic improvement in classification accuracy between zero and 100 frames where classification peaks. After this point there is a decline in performance. This is perhaps because of the class itself. The class involves a fast break meaning with the characterisation of this class involving a quick movement. If the window size is too small it gets confused with other defending classes or too big then the main action is lost. Again a rapid rise in



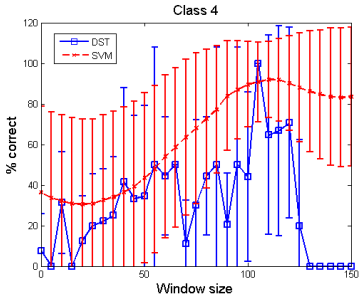
Class 1 - offense against set-up defense



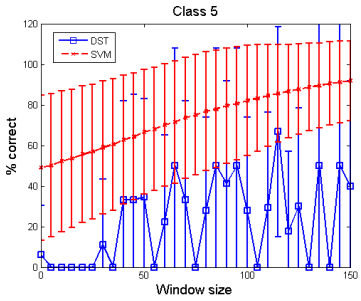
Class 2 - defense, returning.



Class 3 - defense, basic defense



Class 4 - offense, fast break



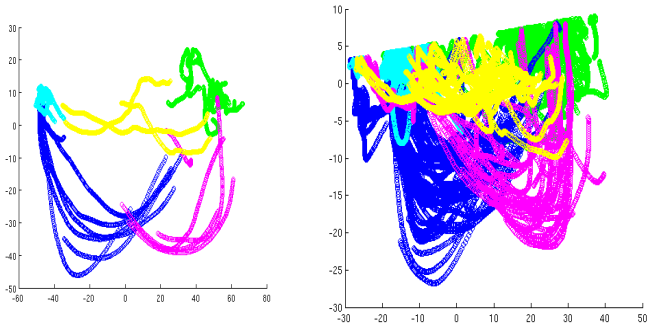
Class 5 - offense, slowly going into offense

Figure 3.8: Per-class results for the CVBASE dataset.

classification accuracy is observed where the shorter window sizes are confused with those of other classes such as defending due to the offense taking a long time to build up. Finally Class 5 displays a rapid rise in classification accuracy where the shorter window sizes are confused with those of other classes. Such an example would be the confusion with defending due to the offense taking a long time to build up.

It is worthwhile noting that those sequences on which the DST method performed poorly are those which had fewer examples. Classes 2,4 and 5 had an average of 750, 330 and 450 training samples respectively. This may indicate that the DST method with its greater number of parameters requires more training data than the SVM method to accurately learn the input data.

The results show that for many cases it is easier to classify what is going on when



Team Individual

Figure 3.9: PCA projection showing training data projected into two dimensions using two eigenvectors with the two largest eigenvalues. 1) Green - offense against set-up defense, 2) Blue defense, returning. 3) Light Blue - defense, basic defense. 4) Purple - offense, fast break 5) Yellow - offense, slowly going into offense.

the team as a whole is considered rather than trying to model individuals within that team. The PCA projections of both the team and the individuals data (corresponding to input data for the SVM and DST methods respectively) and plot them as given in figure 3.9. When looking at the plot taking the team as a whole (left plot) the classes are clearly more separated than when looking at individuals (right plot). This can account for the better overall performance of the SVM method which used this data.

One reason that PCA data was not used within the classification methods themselves is that it would not be clear how much of an effect the PCA was having on the data compared to the classification methods themselves. Due to the way that the methods require input data (individual vs concatenated) it would not be possible to test whether it was the model or the PCA projections which were causing the differences in classification. However it is a useful tool for plotting the data. If one were to use PCA projected data there would also be the additional parameter of how many eigenvectors to use in the reconstruction.

3.7.4.1 Continuous Classification

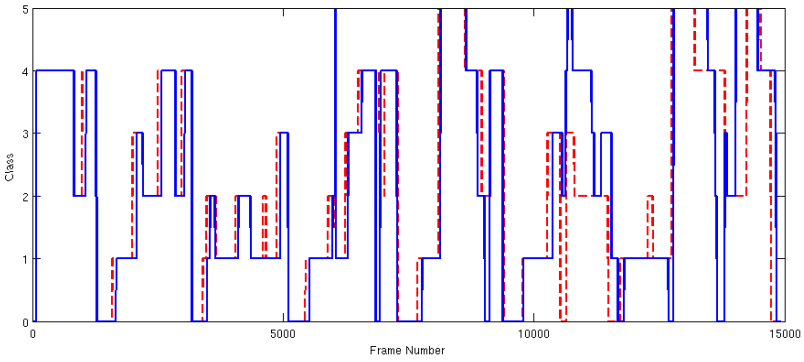
In addition to the classification of pre-segmented sequences a trained model was run over the whole video sequence (all 15,000 frames). Models were trained on a randomly selected sample of complete activity sequences. The number of training sequences accounted for 50% of the total video length. This random sampling approach was taken in preference to simply dividing the complete video in half as the distribution

of activities is not uniform and some only occur in the second half of the game. The models were trained using the procedures described in the preceding sections. As this process results in a continuous labelling of the complete sequence the time out class was also used for training. For labelling an individual frame the previous n frames were used as input into the classifier. Throughout the experiments the window size was set to 100. This produces a labelling for every frame in the sequence (after 100 frames have passed). The results are shown in figure 3.10. It is clearly visible that there is some delay in making a correct classification when the activity changes in both the SVM-group and the DST classifiers. The overall correct classification rate when using the SVM-group classifier is 72.3%, with the DST giving a performance of 61.9%. When looking at the plot of correct and predicted there is a high correlation, with most of the mis-classifications being due to delaying the decision or between frames 10,000-12,000. The team activity changes fairly rapidly in quick succession during this time period, which may explain the poor performance. It is also visible that there is a lag between the current action and what the algorithm thinks is going on. This is due to the decision being based upon previous information due to the window size.

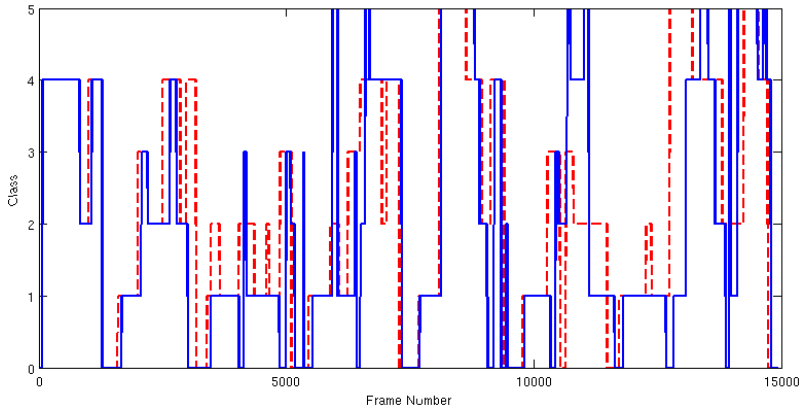
3.8 Conclusion

This chapter has shown that in certain cases where team interaction is constrained, such as in the team sports domain, better classification performance can be obtained by considering the team as a whole. There are many situations where this approach may not find favour. Such an example would be when the team is not a consistent size or if the interactions are not as regular in nature as those encountered in team games. By considering team interactions as a whole rather than modelling each individual it can be possible to reduce model complexity (number of parameters). This can be very important in many domains where it is often hard and laborious to obtain datasets. We also want to be able to quickly understand and classify data despite a relatively small training set. The ability of a computer system to classify team game interactions would be useful for smart video applications such as selecting only the interesting highlights from a game for later viewing.

Within this chapter we have compared two models which take a individual and a team based approach to classifying the teams state. The models are also different in a more fundamental way with the DST being a generative model whilst the SVM is a discriminative model. In this particular case the discriminative model performs



(a)



(b)

Figure 3.10: Classifications for every frame in the handball sequence. Dotted line denotes correct (ground truth) class. Continuous line denotes label determined by the classifier. (a) labelling as given by the SVM-group classifier. (b) labelling as given by the DST classifier.

better than the generative one. However one potential drawback to using a discriminative model is that it is not possible to generate samples from the model and so check that such samples agree with the input data. Such tests can be done to allow the validation of the model and to help tune the parameters and state space by using model selection algorithms [Gong and Xiang, 2003b]. The discriminative SVM cannot produce additional samples as it models the boundary between the classes not the classes themselves. The SVM is also a simpler model as only the boundary is required for classification where as the more complex discriminative model is not able to form an accurate model. For classification problems such as this there is no requirement to produce samples from the model, however one could imagine situations (such as model validation) where such a property would be desirable.

Future chapters will seek to extend the classification of interactions to more general and less structured cases.

Acknowledgement

Thanks must go to Andrew Howard who kindly made his code for computing the DST available.

Chapter 4

Detection and Classification of Interacting Persons

4.1 Introduction

This chapter presents a way to classify interactions between people. Examples of the interactions we investigate are people meeting one another, walking together and fighting amoung others. A new feature set is proposed along with a corresponding classification method. Results are presented which show the new method performing significantly better than the previous state of the art method as proposed by [Oliver et al., 2000b].

4.2 Previous Work

There has been much previous work upon identifying what activity individual people are engaged in. [Davis and Bobick, 2001] used a moment based representation based on extracted silhouettes and [Efros et al., 2003] modelled human activity by generating optical flow descriptions of a person's action. Descriptions were generated by first hand-tracking an individual, re-scaling to a standard size and then taking the optical flow of a persons actions over several frames. A database of these descriptions was created and matched to novel situations. This method was extended by [Robertson, 2006] who also included location information to help give contextual information to a scene. Location information is of assistance when trying to determine if someone is loitering or merely waiting at a road crossing. Following on from flow based features [Dollar et al., 2005] extracted spatio-temporal features to identify sequences of actions.

[Ribeiro and Santos-Victor, 2005] took a different approach to classify an individual's actions in that they used multiple features calculated from tracking (such as speed, eigenvectors of flow) and selected those features which best classified the persons actions using a classification tree with each branch using at most 3 features to classify the example.

The classification of interacting individuals was studied by [Oliver et al., 2000a] who used tracking to extract the speed, alignment and derivative of the distance between two individuals. This information was then used to classify sequences using a coupled hidden Markov model (CHMM). [Liu and Chua, 2006] expanded the two person classification to three person sequences using a hidden Markov model (HMM) with an explicit role attribute. Information derived from tracking was used to provide features such as the relative angle between two persons to classify complete sequences. Xiang and Gong again used a CHMM to model interactions between vehicles on an aircraft runway. This features are calculated by detecting significantly changed pixels over several frames. The correct model for representing the sequence is determined by the connections between the separate models. Goodness of fit is calculated by the Bayesian information criterion. Using this method a model representing the sequences actions is determined.

Multi-person interactions within a rigid formation was also the goal of [Khan and Shah, 2005]who used a geometric model to detect rigid formations between people, such an example would be a marching band. [Intille and Bobick, 2001] used a pre-defined Bayesian network to describe planned motions during American football games. Others such as [Perse et al., 2007] also use a pre-specified template to evaluate the current action being performed by many individuals. Pre-specified templates have been used by [Van Vu et al., 2003, Hongeng and Nevatia, 2001] within the context of surveillance applications.

4.3 Contribution

The main contributions of this chapter are to advance state of the art by improving classification of unstructured interactions over the previous best published method ([Oliver, 2000, Oliver et al., 2000a]). As this is unstructured we also do not require a template in order to evaluate what is going on as in [Van Vu et al., 2003, Hongeng and Nevatia, 2001] and [Perse et al., 2007].

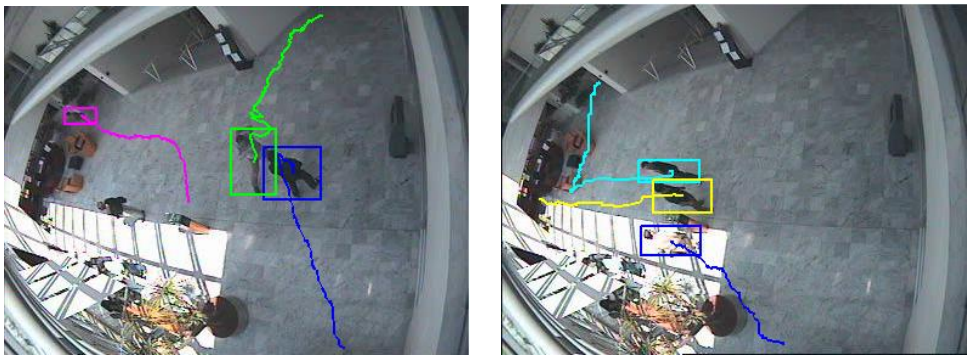


Figure 4.1: Bounding box tracking. Coloured lines show the previous position of the centre of the tracked object.

4.4 Features

Video data is rich in information with the resolution of modern surveillance cameras capable of delivering megapixel resolution at a sustained framerate of greater than 10fps. Such data is overwhelming and mostly unnecessary for classification of interactions. As a first step, tracking of the individuals is employed. There is a rich body of work upon tracking of people and objects within the literature. This work is discussed in the literature review (see chapter 1). For all experiments performed in this thesis the bounding box method of identifying a person was used. The positional information was calculated based upon the centre of this box. This process is illustrated in figure (4.1). Such tracking information is typical of the output of many tracking procedures and it will be assumed that such a tracker is available throughout all experiments carried out in this chapter.

It was suggested by [Gigerenzer et al., 1999] that there are seven features which were required for interactions to be recognized between two tracks. In their experiments two participants were required to act out scenarios such as chasing, courting and fighting by moving a cursor around a computer screen. The two participants were in different locations so the interaction remained anonymous.

The seven suggested features are :

1. mean absolute velocity across both agents
2. mean absolute vorticity across both agents
3. relative distance between agents
4. relative velocity of the two agents

5. relative vorticity of the two agents

6. relative heading of the two agents

7. relative angle between one agent's heading and the others current position

Classification of these scenarios by German adults resulted in a 75% accuracy with the significant confusion arising when attempting to distinguish between playing and fighting. Although the work of [Gigerenzer et al., 1999] provides some rational for choice of features, other authors have used alternative features. [Oliver et al., 1998, 2000a] use speed, alignment, relative distance and derivative of the relative distance for classifying pairwise interactions using a coupled hidden Markov model. Others such as [Liu and Chua, 2006] use the relative angles and ratios of dot products. Other authors such as [Robertson and Reid, 2005] and [Gong and Xiang, 2003a] also incorporate positional information. Within this work the absolute positional information is not used. Such information because an interaction should be recognised regardless of its position.

4.4.1 Movement Based Features

Movement plays an important role in recognising interactions. The speed of an individual is calculated as shown in equation (4.1). The double vertical bar ($\|.\|$) represents a vector $L2$ norm as given by $\|\mathbf{x}\| = sqrt(\mathbf{x}_1^2 + \mathbf{x}_2^2 +, \ldots, +\mathbf{x}_n^2)$, where $\mathbf{x}_n$ refers to the n^{th} component of the vector $\mathbf{x}$.

$$s_i^t = \frac{\|\mathbf{p}_i^t - \mathbf{p}_i^{t-w}\|}{w} \tag{4.1}$$

Here $\mathbf{p}_i^t$ refers to the position of the tracked object at time t for object i. Within this work only two dimensional co-ordinates are used $\mathbf{p}_i^t = [x_i^t, y_i^t]$ due to the tracking information only being available in two dimensions. The w temporal offset is introduced due to the high frame rates which typify many modern video cameras. High frame rates of around 25fps can mean that taking the last frame (w=1) results in very small movement between subsequent frames. Combined with the effect of noise and imprecision in the tracking process these small values are mostly dominated by noise. The absolute difference in speed ($\varepsilon_{[i,j]}^t$) between two tracks is also calculated ($\left|s_i^t - s_j^t\right|$).

Vorticity (v_t^i) is measured as a deviation from a line. A window of points is used $\mathbf{P}_i^t = [\mathbf{p}_i^{t-w}, .., \mathbf{p}_i^t]$, the window is the same size as that used in equation (4.1). The principle direction of the points are found by fitting a line to the set of points. To

do so the eigenvector corresponding to the largest eigenvalue of the covariance of the window of points was found. The distances from each point in the window to the line are calculated. These distances are then summed and normalised by window length.

The amount of movement over a time period w is also calculated.

4.4.2 Alignment Based Features

The alignment of two tracks can give valuable information as to how they are interacting. The degree of alignment is common to [Gigerenzer et al., 1999], [Oliver et al., 1998, 2000a] and [Liu and Chua, 2006] who all make use of such information when classifying trajectory information.

To calculate the dot product the heading ($\mathbf{h}$) of the object is taken as in equation (4.2) and the dot product was calculated from the directions of tracks i and j.

$$\hat{\mathbf{h}}_i^t = \frac{\mathbf{p}_i^t - \mathbf{p}_i^{t-w}}{\|\mathbf{p}_i^t - \mathbf{p}_i^{t-w}\|} \tag{4.2}$$

$$a_{[i,j]}^t = \hat{\mathbf{h}}_i^t \cdot \hat{\mathbf{h}}_j^t \tag{4.3}$$

In addition to alignment the potential intersection ($\gamma_t^{i,j}$) of two trajectories is also calculated. Such a features are suggested in [Gigerenzer et al., 1999] and [Liu and Chua, 2006]. We first test for an intersection of the headings. This is achieved as shown in algorithm 1.

Within algorithm 1 we use an unnormalized version of the heading. Within this algorithm the $\times$ symbol represents the cross product. The cross product is defined between two vectors ($(x_1, y_1) \times (x_2, y_2)$) as given in equation (4.4). As before $\mathbf{p}_j^t$ refers to the point in two dimensions of track j at time t.

The line is defined by equation (4.2) with $\mathbf{h}$ being the heading of the object, $\mathbf{s}$ being the cross product with μ being the mean position of the points.

$$(x_1, y_1) \times (x_2, y_2) = (x_1 \times y_2) - (x_2 \times y_1) \tag{4.4}$$

As the eigenvectors direction may not correspond to the direction of the trajectory is moving in (as it is fit to a set of points) it is further necessary to take into account the direction $\mathbf{d}_{[i,j]}^t$ to make sure that both trajectories are heading in the direction of the intersection.

Algorithm 1 Algorithm to determine the meeting of two trajectories

$\mathbf{d}^t_{[i,j]} = \mathbf{p}^t_j - \mathbf{p}^t_i$ // distance between targets j and i at time t
$c^t_{[i,j]} = \mathbf{h}^t_i \times \mathbf{h}^t_j$ // cross product of the headings of i and j at time t
if $c^t_{[i,j]} != 0$ // non zero cross product
  $\delta^t_{[i,j]} = \frac{\mathbf{d}^t_{[i,j]} \times \mathbf{h}^t_j}{c^t_{[i,j]}}$
  $\iota^t_{[i,j]} = \mathbf{p}^t_i + \delta^t_{[i,j]}.\mathbf{h}^t_i$ - they intersect at this point
else if $c^t_{[i,j]} = 0$ // parallel but may be in opposite direction - ie walking towards each other
  if $\|\mathbf{d^t_{[i,j]}}\| > \mathbf{0}$ and $\mathbf{h}^t_i = -\mathbf{h}^t_j$ and $\|\mathbf{h}^t_i\| > 0$ and $\|\mathbf{h}^t_j\| > 0$
    $r^t_i = \frac{||\mathbf{h}^t_i||}{||\mathbf{h}^t_i|| + ||\mathbf{h}^t_j||}$
    $\delta^t_{[i,j]} = \frac{\mathbf{d^t_{[i,j]}} \times r^t_i}{c^t_{[i,j]}}$
    $\iota^t_{[i,j]} = \mathbf{p}^t_i + \delta^t_{[i,j]}.h^t_i$- they intersect at this point
  else
    $\iota^t_{[i,j]}$ =do not meet
end

4.4.3 Distance Based Features

Distance is a good measure for many types of interaction, for example meeting is not possible without being in close physical proximity. First an Euclidean distance measure is used as given in equation 4.5.

$$d^t_{[i,j]} = \|\mathbf{p}^t_i - \mathbf{p}^t_j\| \tag{4.5}$$

The derivative of the distance was also calculated. This is the difference in distance at contiguous time steps. It is calculated as shown in equation 4.6 below.

$$\dot{d}^t_{[i,j]} = d^t_{[i,j]} - d^{t-1}_{[i,j]} \tag{4.6}$$

An instantaneous measure such as the distance and the derivative of the distance can both be prone to short term tracking errors. In an effort to remove this effect a window size containing w points (as in $\mathbf{P}^t_i$ in section 4.4.1) was averaged. The distance was calculated for every point (as in equation 4.5) in this window.

$$\hat{d}^t_{[i,j]} = \frac{\sum_{t-w}^{t} d^t_{[i,j]}}{\|t - (t-w)\|} \tag{4.7}$$

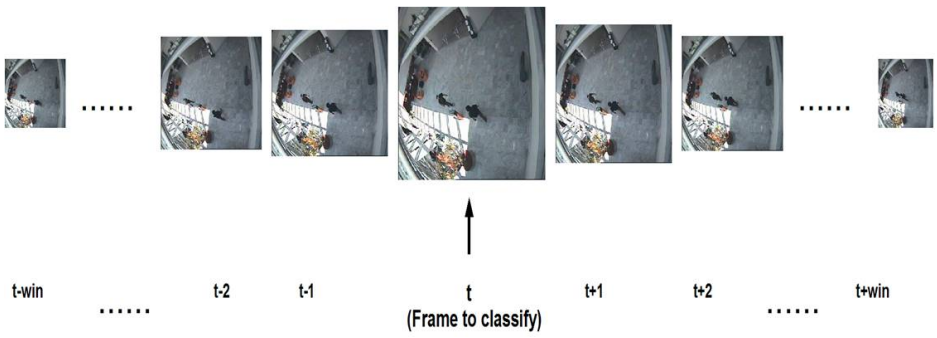


Figure 4.2: The frame to classify (t) uses information from $\pm w$ frames around the current frame in order to classify the frame.

4.4.4 Final Feature Vector

The final feature vector for each pair of people is given in equation (4.8).

$$\mathbf{r}_t^{i,j} = \left[s_i^t, s_j^t, \dot{s}_t^i, \dot{s}_t^j, \varepsilon_{[i,j]}^t, a_{[i,j]}^t, d_{[i,j]}^t, \dot{d}_{[i,j]}^t, \hat{d}_{[i,j]}^t, \nu_t^i, \gamma_t^{i,j} \right] \tag{4.8}$$

The vector between persons i and j at time t is made up of the speed of each person (s_i^t, s_j^t) and along with the corresponding change in speed $\dot{s}_t^i, \dot{s}_t^j$. The alignment, distance and change in distance at a particular point in time is given by $a_{[i,j]}^t$, $d_{[i,j]}^t$and $\dot{d}_{[i,j]}^t$ respectively. The normalised distance is given by $|d|_{[i,j]}^t$ and the vorticity of a trajectory is given by ν_t^i. The possible intersection of two trajectories is given by $\gamma_t^{i,j}$. The final vector contains 10 features. The data was normalised to have zero mean and unit standard deviation.

4.5 Observation Window Size

Throughout these experiments we investigated the role of varying the number of video frames used before making a decision as to what is happening within the frame. Figure 4.2 below shows how this is achieved. Throughout this work we used information from before and after the current frame in order to classify it. This helps with the lag problem as experienced in Chapter 2 (Team Interaction). The window size variation throughout this work is equivalent to a few seconds delay. This was not foreseen as a problem if such an approach was taken in a real surveillance application. The fact that there would be a lag in classification if making use of only previous information seems an appropriate trade-off for an increase in accuracy.

4.6 Classification Methods

This section introduces the classifiers which are used throughout subsequent experiments. We make use of a simple linear discriminant classifier (LDA) which is non-probabilistic and provides a baseline for performance. This is introduced in section 4.6.1. We then detail hidden Markov models (HMM) in section 4.6.2 which are used widely throughout the literature. We then introduce a newer model, the conditional random field (CRF) in section 4.6.3. Finally the previous best method as suggested by [Oliver, 2000] is briefly reviewed in section 4.6.4.

4.6.1 Linear Discriminant Analysis

Linear discriminant analysis (LDA) seeks to maximise the objective function given in equation 4.9. Here $\mathbf{S}_W$ is the within class scatter matrix with $\mathbf{S}_B$ being the between class scatter matrix. Both the between class and within class scatter matrices are defined in equations 4.10 and 4.11 respectively.

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{4.9}$$

For the scatter matrices (given below) N is the total number of samples with N_c being the number of examples of class c. The mean of class c is denoted as μ_c (as shown in equation 4.12) with the variance ($\bar{x}$) calculated as in equation (4.13).

$$\mathbf{S}_B = \frac{1}{N_c} \sum_{c}^{N_c} (\mu_c - \bar{x})(\mu_c - \bar{x})^T \tag{4.10}$$

$$\mathbf{S}_W = \frac{1}{N} \sum_{c}^{N_c} \frac{1}{N_c} \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{4.11}$$

$$\mu_c = \frac{1}{N_c} \sum_{i \in c} x_i \tag{4.12}$$

$$\bar{x} = \frac{1}{N} \sum_{c} N_c \mu_c \tag{4.13}$$

The objective function (equation 4.9) is often referred to as the signal to noise ratio. What we are trying to achieve is a projection ($\mathbf{w}$) which maximises the distance of the class means relative to the (sum of) variances of a particular class. To generate a solution it is noted that equation 4.9 has a property whereby it is invariant with respect to scaling of the $\mathbf{w}$vectors (eg $\mathbf{w} \rightarrow \alpha\mathbf{w}$, where α is some arbitrary scaling). Therefore

$\mathbf{w}$ can be chosen such that $\mathbf{w}^T\mathbf{S}_W\mathbf{w} = 1$. It is also common (and indeed the case here) to perform a whitening step (zero mean and unit variance) on the data prior to input into this method. Therefore LDA is:

$$\begin{aligned} min_w \quad & -\frac{1}{2}\mathbf{w}^T\mathbf{S}_B\mathbf{w} \\ \text{subject to} \quad & \mathbf{w}^T\mathbf{S}_W\mathbf{w} = 1 \end{aligned}$$

This gives the Lagrangian

$$\mathcal{L}_p = -\frac{1}{2}\mathbf{w}^T\mathbf{S}_B\mathbf{w} + \frac{1}{2}\lambda(\mathbf{w}\mathbf{S}_w\mathbf{w} - 1) \tag{4.14}$$

Using the Karush-Kuhn-Tucker condition the equality which needs to hold at the solution is:

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w} \tag{4.15}$$

As the matrix $\mathbf{S}_W^{-1}\mathbf{S}_B$ is not symmetric then equation 4.15 is in the form of a generalised eigenvalue problem. Using the fact that $\mathbf{S}_B$ is symmetric and positive definite it is possible to write $\mathbf{S}_B = \mathbf{S}_B^{\frac{1}{2}}\mathbf{S}_B^{\frac{1}{2}}$ where $\mathbf{S}_B^{\frac{1}{2}}$ is defined as in equation 4.16.

$$\mathbf{S}_B^{\frac{1}{2}} = U\Lambda^{\frac{1}{2}}U^{-1} \tag{4.16}$$

In equation (4.16) U are the eigenvectors of matrix $\mathbf{S}_B$ and Λ are eigenvalues obtained by a singular value decomposition of the matrix $\mathbf{S}_B = U\Lambda U$. If $\mathbf{v}$ is defined as $\mathbf{v} = \mathbf{S}_B^{\frac{1}{2}}\mathbf{w}$ then it follows that,

$$\mathbf{S}_B^{\frac{1}{2}}\mathbf{S}_W^{-1}\mathbf{S}_B^{\frac{1}{2}}\mathbf{v} = \lambda\mathbf{v} \tag{4.17}$$

This problem is now a regular eigenvalue problem for a symmetric positive definitive matrix ($\mathbf{S}_B^{\frac{1}{2}}\mathbf{S}_W^{-1}\mathbf{S}_B^{\frac{1}{2}}$) and solutions can be found for λ_k and $\mathbf{v}_k$ that give $w_k = \mathbf{S}_B^{\frac{1}{2}}\mathbf{v}_k$. Projections are found for each class (ie class vs all others projection) and the mean and variance (C) of the class projection are found. In order to classify a novel point the new point is projected with $\mathbf{S}_W^{-1}$. The class label is determined by taking the smallest Mahalonobis distance between the calculated class model's mean and variance and the new test point.

4.6.2 Hidden Markov Models

The model is parameterised by the prior distribution Π with each element π_i representing $\pi_i = p(x = i)$ across all hidden states $i \in [1,..,N]$. The (stationary) state transition matrix $\mathbf{A}$ is used to represent the probability of a transition from one state (i) to another (j) through time. An entry within the symmetric matrix $\mathbf{A}$ is referenced by $a_{i,j} = p(x_t = i|x_{t-1} = j)$. Within this work we are concerned with continuous inputs $(\mathbf{r}_t)$ which can be accommodated within the model by using a Gaussian mixture model to model the distribution $p(\mathbf{r}_t|x_t = j)$.

$$b_j(\mathbf{r}_t) = \sum_{m=1}^{M} c_{j,m} \mathcal{N}\left(\mathbf{r}_t, \mu_{j,m}, \Sigma_{j,m}\right) \tag{4.18}$$

Here the observed data **R**is the vector being modelled, $c_{j,m}$ is the mixture coefficient for the m^{th} mixture in state j. $\mathcal{N}$ is a Gaussian distribution with mean vector $\mu_{j,m}$ and covariance matrix $\Sigma_{j,m}$ for the m^{th} mixture component in state j. A restriction on the mixture coefficient is that it must satisfy the following constraints:

$$\sum_{m=1}^{M} c_{j,m} = 1,\ 1 \leq j \leq N \tag{4.19}$$

$$c_{j,m} \geq 0, \quad 1 \leq j \leq N, \quad 1 \leq m \leq M \tag{4.20}$$

The parameters of the HMM can be represented as $\lambda = (\Pi, \mathbf{A}, \theta)$ where θ represents the parameters for the mixture model $\theta = (\mathbf{C}, \mu, \Sigma)$.

4.6.2.1 Inference

The goal of inference is to compute the probability of the observation sequence **R** given the parameters of the model $p(\mathbf{R}|\lambda)$. Here the forward backward procedure (also known as the Baum-Welch algorithm [Baum et al., 1970]) is presented. It is also possible to solve this problem using the well known junction tree algorithm [Huang and Darwiche, 1996]. However, here the forward backward algorithm is used for speed and simplicity.

The forward variable is defined as:

$$\alpha_t(i) = p(\mathbf{r}_1, \mathbf{r}_2, .., \mathbf{r}_t, q_t = S_j|\lambda) \tag{4.21}$$

with q_t being the (hidden) state at time t. To generate the probability the solution to $\alpha_t(i)$can be calculated incrementally as show in algorithm 2.

Algorithm 2 Forward parameter recursive algorithm

Initialisation

$\alpha_1(i) = \pi_i b_j(\mathbf{r}_1)$

Induction

$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i)\, a_{i,j}\right) b_j(\mathbf{r}_{t+1})$

Termination

$p(\mathbf{R}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Algorithm 3 Backward parameter recursive algorithm

Initialisation

$\beta_T(i) = 1$

Induction

$\beta_t(i) = \sum_{j=1}^{N} a_{i,j} b_j(\mathbf{r}_{t+1}) \beta_{t+1}(j)$

The backward parameter β is also used within inference and parameter updating and represents the conditional probability given below in equation 4.22.

$$\beta_t(i) = P(\mathbf{r}_{t+1}, \mathbf{r}_{t+2}, .., \mathbf{r}_T | q_t = S_i, \lambda) \tag{4.22}$$

As in the case of the forward parameter a recursive updating procedure is used as given in algorithm 3.

The forward and backward variables can be used to calculate the probability of being in state i at time t by equation (4.23):

$$\gamma_t(i) = \frac{\alpha_t(i)\,\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)} \tag{4.23}$$

We make use of this equation when updating the parameters of the model in the following subsection.

4.6.2.2 Parameter Updates

This section details how to updates the parameters of the HMM. All updates are performed within an expectation maximisation (EM) framework.

The prior probability is estimated as:

$$\hat{\pi}_i = \gamma_1(i) \tag{4.24}$$

with $\gamma_1(i)$ being defined as in 4.23, the hat represents the fact that this is an estimate of the true prior distribution. The state transition matrix is updated according to the

following:

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{4.25}$$

The term $\xi_t(i,j)$ represents the probability of being in state i at time t and in state j at time $t+1$ given the observations and the model parameters. This is shown in equation 4.26.

$$\xi_t(i,j) = p\left(q_t = S_i, q_{t+1} = S_j | \mathbf{R}, \lambda\right) \tag{4.26}$$

Using the previous definitions of the forward $\alpha_t(i)$ and backward $\beta_t(i)$ variables the term $\xi_t(i,j)$ can be written as:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{i,j} b_i(\mathbf{r}_{t+1}) \beta_{t+1}(j)}{\sum_{r=1}^{N} \sum_{s=1}^{N} \alpha_t(r) a_{r,s} b_r(\mathbf{r}_{t+1}) \beta_{t+1}(s)} \tag{4.27}$$

For the estimation of the parameters of the mixture models $\theta = (\mathbf{C}, \mu, \Sigma)$ the following formulas are used:

$$\hat{c}_{j,k} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{n=1}^{M} \gamma_t(j,n)} \tag{4.28}$$

$$\hat{\mu}_{j,k} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot \mathbf{r}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{4.29}$$

$$\hat{\Sigma}_{j,k} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot \left(\mathbf{r}_t - \mu_{j,k}\right)\left(\mathbf{r}_t - \mu_{j,k}\right)^T}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{4.30}$$

For each class, examples from that class are presented to a HMM and the parameters are set as described above. There is one model for each class. In order to classify a new example ($\mathbf{R}$) the likelihood of that example ($p(\mathbf{R}|\lambda)$) is computed for each trained model (see section 4.6.2.1). The model producing the highest likelihood ($p(\mathbf{R}|\lambda)$) is then taken to be the class of the new example. Within this experiment we used a model with 4 hidden states and 3 mixture components per state. These values were determined empirically.

4.6.3 Conditional Random Field

In this section the workings of a conditional random field (CRF) are explained and then the specific formulation as applied in this thesis is given. The structure of the CRF weuse is shown in Figure 4.3. The CRF can be configured to resemble HMM like

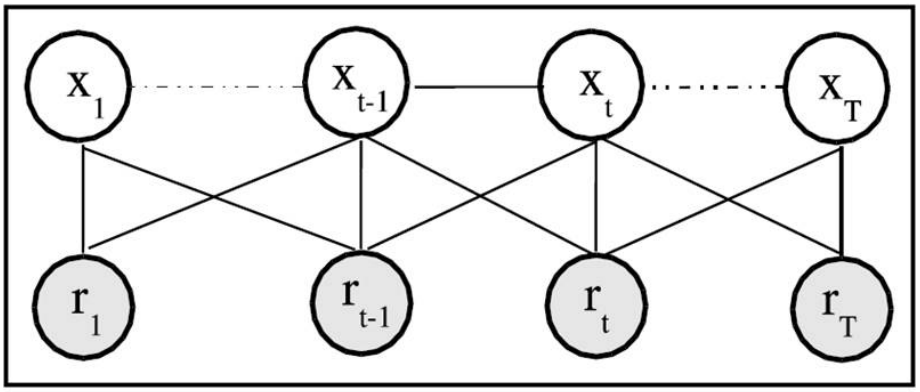


Figure 4.3: CRF model. The observations (r) are shown for each timestep. The class label x is also shown.

models. However they can be more expressive in that arbitrary dependencies on the observations are allowed. Using the feature functions of the CRF allows any part of the input sequence to be used at any time during the inference stage. It is also possible that different state's (classes) can have differing feature functions (though we do not make use of this here). The feature functions describe the structure of the model.

The CRF is also a discrimanative model where as the HMM is a generative one. A potential advantage of the discriminative CRF over generative models is that they have been shown to be more robust to violations of independence assumptions [Lafferty et al., 2001].

The discrete temporal state at a particular timestep t is given by x_t which takes a value from the set of all class labels $x \in \mathcal{X} = \{1,2,..,C\}$. Here C is the number of class labels and $t = 1,..,T$,with T being the maximum length of the sequence. Observations at time t are denoted as $\mathbf{r}_t$ with the joint observations given as $\mathbf{R}_t = (\mathbf{r}_1,..,\mathbf{r}_t)$. Likewise the join state is given by $\mathbf{X}_t = (x_1,..,x_t)$. For notational compactness we shall often refer to $\mathbf{X}_t$ as $\mathbf{X}$ and $\mathbf{R}_t$ as $\mathbf{R}$ in accordance with other authors ([Sminchisescu et al., 2005, Wallach, 2004]).

The distribution over joint labels $\mathbf{X}$ given observations $\mathbf{R}$ and parameters θ are specified below :

$$p_\theta(\mathbf{X}|\mathbf{R}) = \frac{1}{Z_\theta(\mathbf{R})} \prod_{c \in C(\mathbf{X},\mathbf{R})} \phi_\theta^c(\mathbf{X}_c, \mathbf{R}_c) \tag{4.31}$$

In the above equation (4.31) ϕ_θ^c is the positive valued potential function of clique c and $Z_\theta(\mathbf{R})$ is the observation dependent normalisation (sometimes referred to as a partition function) as given in 4.32. $C(\mathbf{X},\mathbf{R})$ is the set of maximal cliques.

$$Z_\theta(\mathbf{R}) = \sum_{\mathbf{X}} \prod_{c \in C(\mathbf{X},\mathbf{R})} \phi_\theta^c(\mathbf{X}_c, \mathbf{R}_c) \tag{4.32}$$

Here a first order linear chain is used (as shown in figure 4.3). The cliques are pairs of neighbouring states (x_t, x_{t+1}), whilst the connectivity among observations is unrestricted and are known and fixed in the graph. A CRF model with T timesteps (as we use here) can be rewritten in terms of exponential feature functions F_θ computed in terms of weighted sums over the features of the cliques. This exponential formulation is given in equations 4.33 and it's normalising term4.34.

$$p_\theta(\mathbf{X}|\mathbf{R}) = \frac{1}{Z_\theta(\mathbf{R})} \exp\left(\sum_{t=1}^{T} F_\theta(x_t, x_{t-1}, \mathbf{R})\right) \tag{4.33}$$

$$Z_\theta(\mathbf{R}) = \sum_{\mathbf{X}} \exp\left(\sum_{t=1}^{T} F_\theta(x_t, x_{t-1}, \mathbf{R})\right) \tag{4.34}$$

The conditional log likelihood of the CRF is given as in equation 4.35. Assuming that the training data is fully labelled $\left\{\mathbf{X}^d, \mathbf{R}^d\right\}_{d=1,..,D}$ the parameters of the model are obtained by optimisation of this function.

$$\mathcal{L}_\theta = \sum_{d=1}^{D} \log p_\theta\left(\mathbf{X}^d|\mathbf{R}^d\right) = \sum_{d=1}^{D}\left(\sum_{i=t}^{T} F_\theta(x_t^d, x_{t-1}^d, \mathbf{R}^d) - \log Z_\theta\left(\mathbf{R}^d\right)\right) \tag{4.35}$$

In order to make parameter optimisation more stable the problem often makes use of a regularised term. This means the problem becomes one of optimising the penalised likelihood ($\mathcal{L}_\theta + \mathcal{R}_\theta$) which consists of both the likelihood $\mathcal{L}_\theta$ (as in 4.35) and the regularising term $\mathcal{R}_\theta$. The regularising term used within this work is chosen to be $\mathcal{R}_\theta = -\|\theta\|^2$ which corresponds to soft feature selection. The derivative of the likelihood w.r.t. the parameters (θ) of the model as given in equation 4.36:

$$\frac{d\mathcal{L}_\theta}{d\theta} = \sum_{d=1}^{D}\left(\sum_{i=t}^{T} \frac{dF_\theta(x_t^d, x_{t-1}^d, \mathbf{R}^d)}{d\theta} - \sum_{\mathbf{X}} p_\theta(\mathbf{X}|\mathbf{R}) \sum_{t=1}^{T} \frac{dF_\theta(x_t, x_{t-1}, \mathbf{R}^d)}{d\theta}\right) \tag{4.36}$$

A gradient ascent optimisation method is used to perform likelihood maximisation. In this case we use scalar conjugate gradient optimisation.

Efficient computation of the observation dependent normalisation can be computed using matrix multiplication. For a first order model a matrix $M_t(\mathbf{R})$ of size $c \times c$ which contains all possible assignments of pairs of neighbouring states to class labels.

$$M_t(\mathbf{R}) = \left[\exp\left(F_\theta\left(x_t, x_{t-1,}\mathbf{R}\right)\right)\right], x_{t,x_{t-1}} \in \mathbf{X} \tag{4.37}$$

The conditional probability of a class label sequence is:

$$p_\theta(\mathbf{X}|\mathbf{R}) = \frac{\prod^{T+1} \exp\left(F_\theta\left(x_t, x_{t-1}, \mathbf{R}\right)\right)}{Z_\theta\left(\mathbf{R}\right)} \tag{4.38}$$

with the observation dependent normalisation factor $Z_\theta\left(\mathbf{R}\right)$ calculated as:

$$Z_\theta(\mathbf{R}) = \left(\prod_{t=1}^{T+1} M_t\left(\mathbf{R}\right)\right)_{start,stop} \tag{4.39}$$

Here two dummy states ($x_0 = start$ and $x_{T+1} = stop$) have been added. The subscript indicates the particular entry of the matrix product. The potential functions at neighbouring sites can be chosen as:

$$F_\theta(x_t, x_{t-1}, \mathbf{R}) = \psi_\theta(x_t, \mathbf{R}) + \psi(x_t, x_{t+1}) \tag{4.40}$$

where

$$\psi_\theta\left(x_t, \mathbf{R}\right) = \sum_{a=1}^{A} \lambda_a f_a\left(x_t, \mathbf{R}\right) \tag{4.41}$$

and

$$\psi_\theta\left(x_t, x_{t-1}\right) = \sum_{b=1}^{B} \beta_b g_b\left(x_t, x_{t-1}\right) \tag{4.42}$$

In this formulation the parameters to estimate are given as :

$$\theta = \left\{\left(\lambda_a, \beta_b\right), a = 1, .., A, b = 1, .., B\right\} \tag{4.43}$$

The feature functions $\psi_\theta\left(x_t, \mathbf{R}\right)$ at each timestep are those which are given in section 4.4. The transition function $\psi_\theta\left(x_t, x_{t-1}\right)$ concerns itself with the class to class transitions which are given in the data labels.

Once trained novel input is given to a CRF model and a probability distribution is given throughout all timesteps for all classes. In this case we choose the highest probability as being the classification label of the new example. A Gaussian prior over the input data was used throughout all experiments.

The feature functions (as given in equation 4.40) describe the conditional dependency between the observations $\mathbf{R}$ and the model's state variables. In this case each dimension of the observation vector will have a feature function for each state (ie class)

variable. The observation is not the feature function itself, but there is a feature function for each observation. The transition function ($g_b(x_t, x_{t-1})$) represents the class transitions. The function we use simply states that if we are in class 1 at time $t-1$ then we expect to be in the same class at time t. Each feature pair ($F_\theta(x_t, x_{t-1}, \mathbf{R})$) consists of a feature function and a transition function. The model is made up of all combinations of features and class transitions. There are $numberofclasses \times numberoffeatures$ (in this case 11, see section 4.4) features. For example if there are 5 classes then there are 55 ($5x11$) features (with associated functions) which make up the CRF model.

When showing a novel example to a trained model the output of the model is a probability distribution ($p_\theta(\mathbf{X}|\mathbf{R})$). The class with the highest probability is selected as the class label for the novel example.

4.6.4 Oliver's Coupled Hidden Markov Model

Here we briefly cover Oliver's method ([Oliver et al., 2000a]) of classifying interacting individuals. This work is reviewed as it provides a state of the art method to compare our results with. Oliver used coupled hidden Markov models to model five different interactive behaviours. These behaviours are:

- Change direction to meet, approach, meet and go on separately.
- Change direction to meet, approach, meet and go on together.
- Approach, meet and go on separately.
- Approach, meet and go on together.
- Follow, reach and go on together.

We briefly present an overview of the methods and features that Oliver used in order to classify such behaviour. Oliver's work is used for comparison with the work presented here.

4.6.4.1 Oliver's features

Oliver et al. use 4 features to classify interacting individuals. Oliver measures the position, orientation and velocity. From these measurements several features are extracted. The two input vectors for the coupled model (one per chain) are :

$$\begin{aligned} f_1 &= [v_1, d_{1,2}, a_{1,2}] \\ f_2 &= [v_2, d_{2,1}, a_{2,1}] \end{aligned}$$

Here the velocity of each person is given by v and is calculated as $v_i = \sqrt{\dot{x_i}^2 + \dot{y_i}^2}$, where the dot represents the movement w.r.t. time. The derivative of the relative distance between two agents is given by $d_{i,j}$. The alignment between two people is given by $a_{1,2} = sign\,(p_1 \cdot p_2)$ where the dot denotes the dot product between two vectors. The vector p represents the directional vector of a person. Each feature vector is calculated at each timestep and used as input into the CHMM model.

4.6.4.2 CHMM inference and learning

The coupled HMM as used by Oliver is shown in figure 4.4 (b).

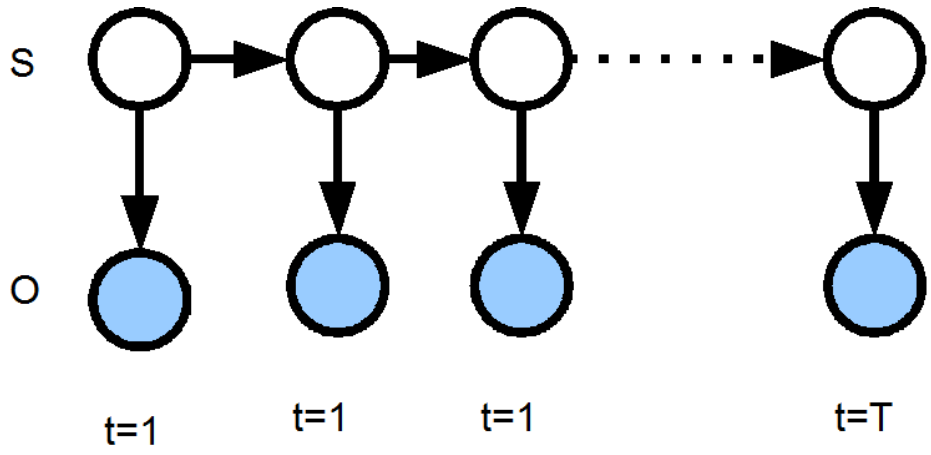
The posterior hidden state of a two chain CHMM is given in equation (4.44).

$$\begin{aligned} P(S|O) = &\frac{P(s_1)P(o_1|s_1)P(s'_1)P(o'_1|s'_1)}{P(O)} \cdots \\ &\times \prod_{t=2}^{T} P(s_t|s_{t-1})P(s'_t|s'_{t-1})P(s'_t|s_{t-1})P(s_t|s'_{t-1}) \cdots \\ &\times P(o_t|s_t)P(o'_t|s'_t) \end{aligned} \tag{4.44}$$

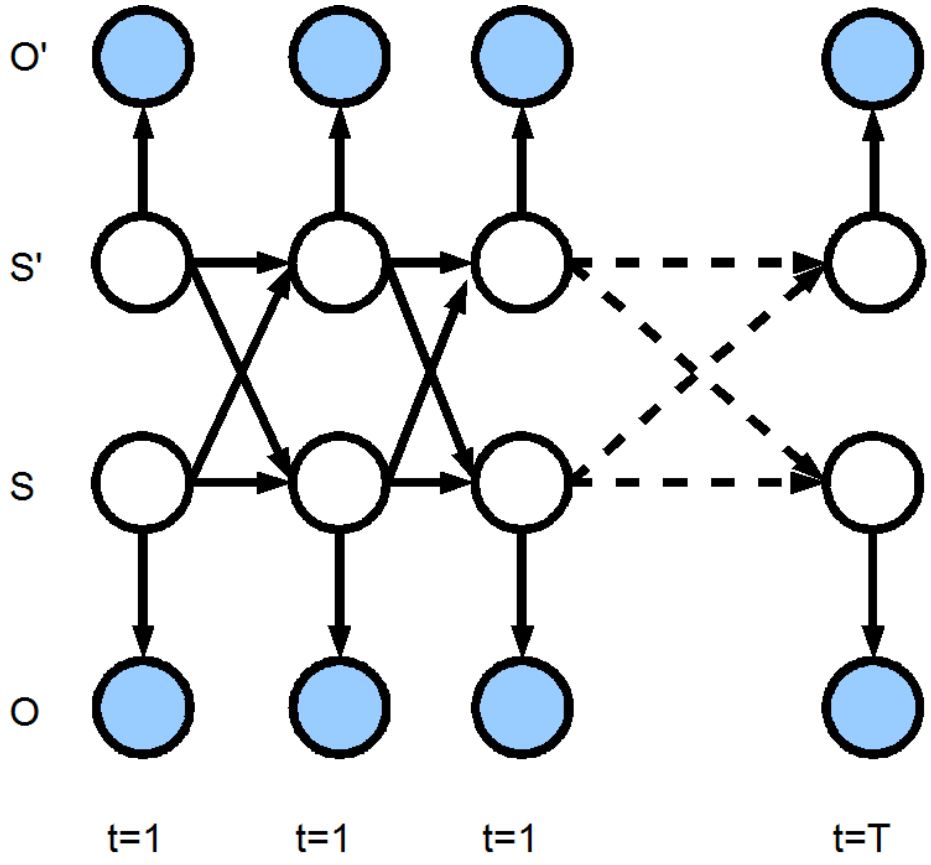
Here the state at a particular timestep is represented by s_t along with the observation o_t at a particular timestep. The prime (') denotes which of the chains the state or observation belongs to (as shown in figure 4.4). The intra state transition probability matrices are represented by $P(s_t|s_{t-1})P(s'_t|s'_{t-1})$ whilst the inter state (cross chain) probability matrices are represented by $P(s'_t|s_{t-1})P(s_t|s'_{t-1})$. The observation probabilities are given by $P(o_t|s_t)P(o'_t|s'_t)$. The learning algorithm used to update the parameters of the model is given by [Brand et al., 1997]. This procedure is not repeated in this thesis and an interested reader should consult [Brand et al., 1997]. In all subsequent experiments we use 2 hidden states with 2 mixture components per state. These values were determined empirically.

4.7 Results

This section details the results of classification on three datasets. First results are presented on synthetic data. Use is then made of real data from the publicly avail-



(a)



(b)

Figure 4.4: Structure of the hidden Markov model model (a), compared to the 2 chain coupled hidden Markov model (b) as used by Oliver.

able CAVIAR [project/IST 2001 37540, 2004] and BEHAVE [Blunsden et al., 2007b] datasets.

4.7.1 Experimental setup

The CAVIAR dataset has been previously used in [Dee and Hogg, 2004a, Wu and Nevatia, 2007] however there is not a universally agreed training and testing set. Therefore it was deemed that in order to characterise an algorithm's true performance upon a dataset it should be tested with different subsets of the entire data. This will give an indication of the expected performance of the algorithm rather than finding a particularly good (in terms of classification accuracy) subset and publishing results based on this. This allows performance to be characterised much more realistically by showing the true expected accuracy rather than simply a lucky configuration. All such tests are carried out independently of one another, for example the model's previously learned parameters are not used in subsequent runs.

We are interested in comparing the four methods as described in the previous section. Furthermore the role of time is investigated. We seek to investigate what is the optimal length of time a sequence should be watched before a decision is made. Results comparing each method and the effects of time are given in section 4.7.2.

Throughout the training procedure the testing set was kept separate from the training data and was unseen by the learning algorithm until testing time. Therefore only the training set was used when determining the parameters of the learning model. Training and testing sets were split 50/50 on a per class basis. Partitioning was done per-class rather than over the whole dataset due to the uneven distribution of classes. Such a step means that in the training stage the learning algorithm will have examples of every type of class. We would not expect a correct classification on unseen classes and so this measure can stop misleading results. In order to show the average performance this procedure was repeated over 50 different partitions of the training and test data.

The dataset contains examples of complete sequences, for example a sequence consisting of two people walking together may be hundreds of frames long. Our goal is to classify each frame correctly. If we were to take this sequence and split it up as training and testing frames then the classification task would be much simpler as training and testing points would be simply a matter of interpolation between highly similar points. It is for this reason that when deciding on the training and testing data we partition based on the complete sequences. This means that an entire walking

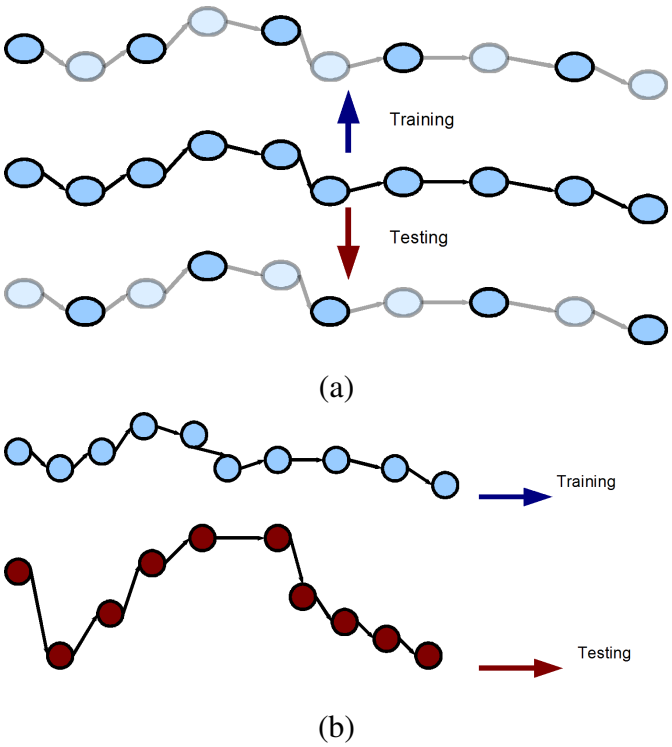


Figure 4.5: Different training and testing partitioning schemes. The top diagram (a) represents data partitioning which would result in interpolation. The method shown in (b) is preferred and should result in a more challenging problem.

together sequence will be assigned to the training set whilst another complete walking together sequence will be assigned to the testing set. This should avoid the pitfall of having training and testing data which is essentially the same. This is especially true as the data has a very strong temporal coupling. The two approaches are shown in figure 4.5.

4.7.2 Classification Results

This section presents the results obtained by using the methods described in the preceding chapters. Results using a conditional random field (CRF) , hidden Markov model (HMM) and its coupled variation (CHMM) are presented. Results using a linear discriminant model (LDA) are also presented and used as a baseline non-probabilistic method to which results are compared. We present the result of classification over many training and testing subsets to give an indication of the standard deviation and the expected performance when using a method. Results are presented over many different window sizes. The classification results are presented on three data sets. The

details of these datasets can be found in the Appendix.

In order to ensure a fair comparison over varying window sizes we restrict the test set so that it is composed of only those frames for which it is possible to compute features for the largest window size. For example if there is a sequence consisting of 5 frames then when using a window size of greater than 5 frames to classify it would not be possible to classify this frame (eg if we were trying to take into account the previous 10 frames it would not be possible to classify this frame). This allows a fair comparison between differing window sizes and ensures that results are presented on the same data.

Graphs are presented showing the averaged performance along with one standard deviation.

4.7.2.1 Synthetic Data set

The synthetic dataset is used to test out the proposed approach with known data. The synthetic dataset consists of 12,385 frames over five classes. These classes are: walk together (2950) meet (1636), wait (2101), split (3243) and follow (2455) with the number of example frames given in brackets. These examples are divided into sequences which range in length between 100 and 500 frames.

Results graphs

The overall results upon the synthetic dataset are presented in figure 4.6. For both the HMM and CHMM models there is a rise in classification performance as the window size is increased from 1 until around 10 frames for the HMM and 20 frames for the CHMM. After this time there is only a very slight gradual improvement. The case of the CRF there is a slight decline as the window size increases. Classification by CRF produces the best results and works well when classifying frames using only a small amount of information. The standard deviation (1σ shown by the shaded regions) for all three methods remains fairly constant throughout. All three of the probabilistic graphical models out perform LDA.

The per class results are shown in figures 4.7 and 4.8. When classifying the behaviour "following" (figure 4.7 (a)) the window size has the most significant effect and all methods show an improvement when the information content is increased. Again the CRF model shows the best performance. When classifying instances of "meeting" and "splitting" there is an initial improvement for both HMM and CHMM methods be-

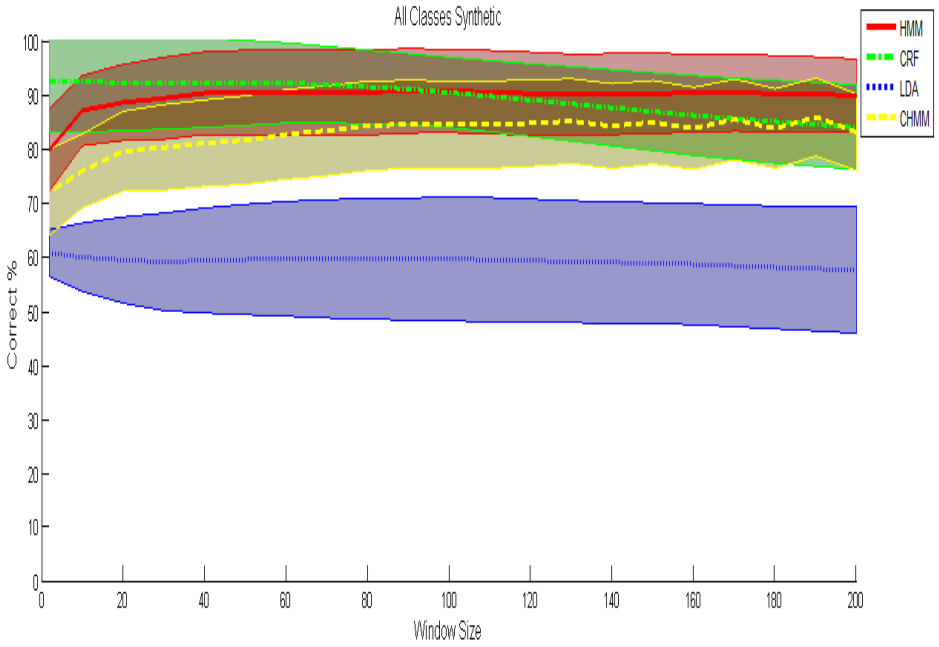


Figure 4.6: Overall results on the synthetic dataset for each method. Lines show averaged results (over 50 runs) whilst the shaded regions show one standard deviation.

tween window sizes of 1 and 20. There is no subsequent improvement when increasing the window size further.

Attempts to classify instances of people "walking together" (figure 4.8 (a)) give the most trouble for all classification methods. This is visible by the low performance and the high standard deviation when classifying examples. Both the HMM and CHMM initially quickly improve when the window size is increased before this improvement declines or even reverses slightly (as is the case of the HMM). Both the CRF and the LDA's performance decreases slightly when the length of time increases. In all cases the standard deviation of performance remains high but fairly constant.

Again a reduction in the accuracy of classification is observed when classifying the waiting behaviour (figure 4.8 (b)) for both the CRF and LDA methods. This reduction tails off for window sizes greater than 140. This is likely to be caused by the longer sequences averaging out better than medium length ones. Classification by LDA gives the highest variation in performance.

This may be caused by the CRF classifier having a better short term model. The CRF features are only made up of features which state the previous class should be the same as this class. The classification of multiple frames just uses the probability of each of these separate frames. For longer window sizes there may be more noise

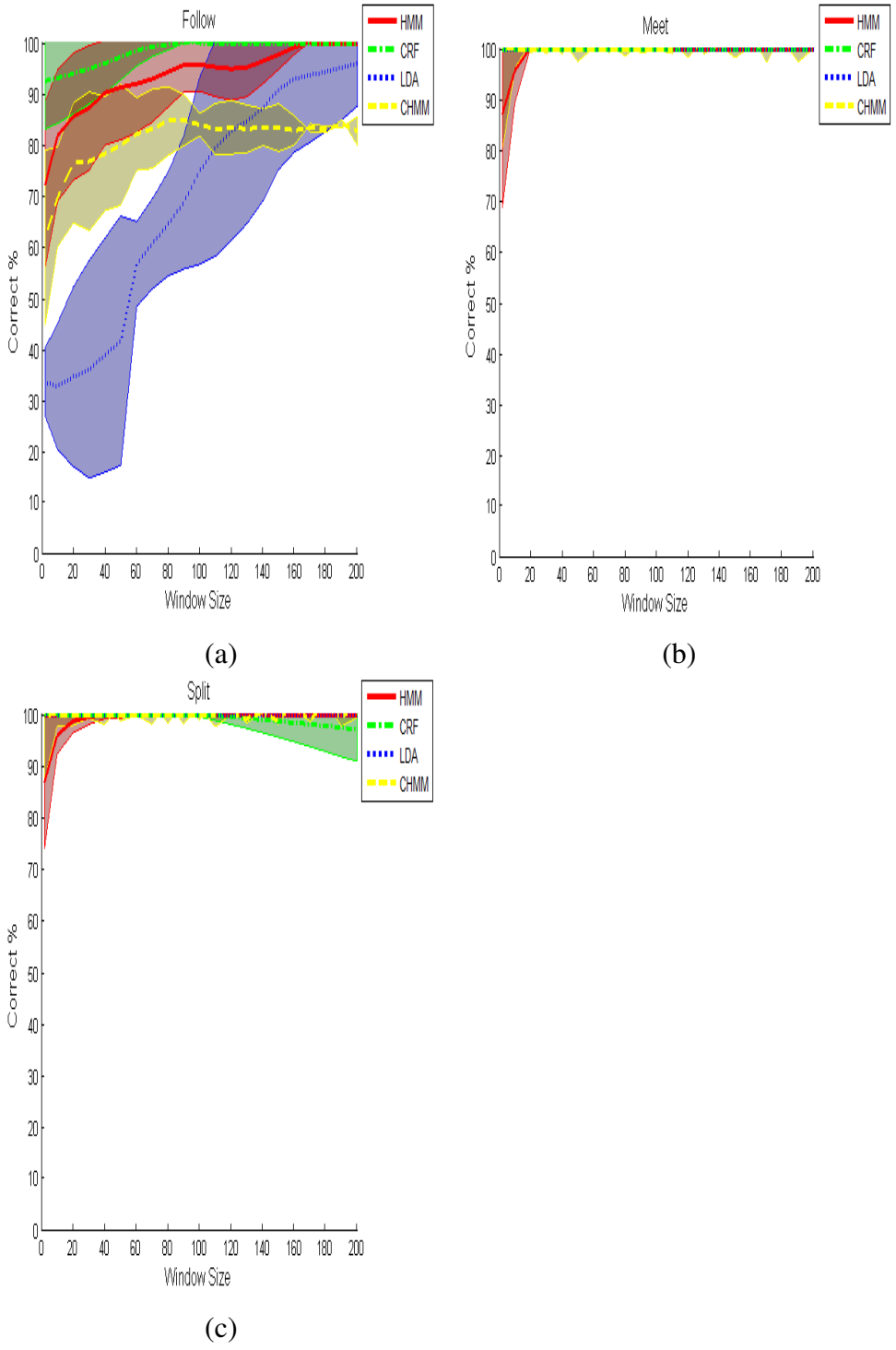
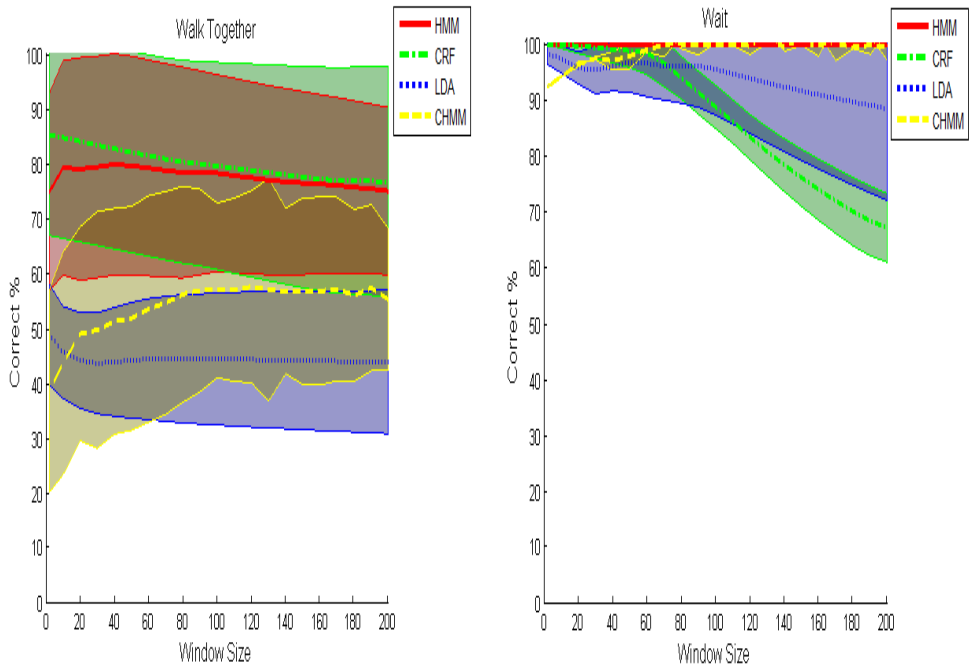


Figure 4.7: Per class results upon the synthetic data. The classes are: (a) follow (b) meet (c) split



(a) (b)

Figure 4.8: Per class results upon the synthetic data. The classes are: (a) walk together (b) wait.

which confuses the short term model of the CRF. In contrast the HMM model is much better at smoothing the signal across the entire sequence. The forward algorithm used to determine the likelihood seems to smooth the signal much better than the CRF. The case of the CHMM the more gradual improvement could be attributable to the larger number of parameters which requires more data in order to represent the data adequately.

Best Results : Confusion Matrices

The confusion matrices giving the best overall result for over all the runs are now presented. The window size at which this performance was achieved is also given. The results for the HMM and CRF models are given in figure 4.9.

The results for the CHMM and the LDA methods are given in figure 4.10.

All methods have some degree of trouble in classifying people walking together. This is most frequently mis-classified as “splitting”, however the converse is not so true. This is perhaps due to the fact that when walking together there are small variations in the distance between people which can be confused with the start of a people splitting up. None of the methods seem to improve dramatically with time which is

		True				
		Walk Together	Meet	Wait	Split	Follow
Classified	Walk Together	0.88	0	0	0	0
	Meet	0	1	0	0	0
	Wait	0	0	1	0	0
	Split	0.12	0	0	1	0
	Follow	0	0	0	0	1
	Total Samples	667	102	164	493	550

HMM - Best 96% at window size 100

		True				
		Walk Together	Meet	Wait	Split	Follow
Classified	Walk Together	0.91	0	0	0	0
	Meet	0	1	0	0	0
	Wait	0	0	1	0	0
	Split	0.09	0	0	1	0
	Follow	0	0	0	0	1
	Total Samples	752	102	174	906	369

CRF - Best 97% at window size 2

Figure 4.9: Confusion matrices displaying the best results for classification upon the synthetic dataset for the hidden Markov model and the conditional random field.

		True				
		Walk Together	Meet	Wait	Split	Follow
Classified	Walk Together	0.57	0	0	0	0.15
	Meet	0	1	0	0	0
	Wait	0	0	1	0	0
	Split	0.43	0	0	1	0
	Follow	0	0	0	0	0.85
	Total Samples	892	363	442	751	1027

CHMM - Best 84% at window size 90

		True				
		Walk Together	Meet	Wait	Split	Follow
Classified	Walk Together	0.48	0	0	0	0
	Meet	0	1	0	0	0
	Wait	0	0	0.95	0	0
	Split	0.34	0	0.05	1	0.03
	Follow	0.18	0	0	0	0.97
	Total Samples	1362	94	175	455	105

LDA - Best 67% at window size 102

Figure 4.10: Confusion matrices displaying the best results for classification upon the synthetic dataset for the coupled hidden Markov model and linear discriminant analysis.

not what you would expect. However the constant movement of people away from one another over a short time period may explain this effect to some degree. The fact that all methods show a similar pattern may mean its cause is rooted in the features used rather than by the classification algorithms themselves. A feature which represents the distance over a long period may improve performance in this respect.

Summary of Classification Upon the Synthetic Dataset

When classifying with a CRF the best performance is achieved with small amounts of information. It should be remembered that some features (such as the speed and the change in distance etc.) use a few frames to calculate them. Even when the window size is small information is still coming from multiple frames. This may explain the good performance with small window sizes.

For all classes the CRF method gives the best performance (ie the CRF attains the highest average performance over all window sizes for all classes). For a particular window size another method may outperform it but the best average performance over all window sizes will be a result of CRF classification.

When looking at the overall results the HMM method performs almost as well when using a window size of greater than 100 frames. Conversely the CRF methods performance declines slightly when the window size decreases. This is most likely due to the temporal model only taking into account the previous frame. The decline may be due to noise throughout the sequence which is not "averaged out" as effectively as in the case of the HMM.

The results of both the HMM and the CRF perform significantly better than the CHMM method as proposed by *[Oliver et al., 2000a].* Perhaps this is not surprising as there is a high degree of redundancy in the information input Oliver's proposed method. Both the standard HMM and the CRF also require significantly less computing time both for training and classification. However, the results using a CHMM creates a similar pattern to that of a HMM in that there is significant early improvement in performance followed by a very slight improvement. This is expected as the likelihood propagates through the model in a similar way (forward-backward algorithm). In contrast for both the CRF and the LDA larger window sizes can cause a decrease in performance (walk together/wait). Again this is most likely due to the noise introduced in the longer sequences coupled with the short term temporal model (or no temporal model, for the LDA). All classes seem to confuse walking together and splitting. This is perhaps understandable as both contain a walking motion and the distance between

the the people is not completely constant.

An extension to the temporal model of the CRF could help to improve results further. The temporal feature (equation 4.42) could be extended to aggregate previous timesteps rather than just using the previous one.

4.7.2.2 CAVIAR Dataset

The CAVIAR dataset contains 11,415 frames which have labelled examples of interactions. Within this set there are 5 distinct classes which we seek to identify and classify. The 5 classes consists of examples of people: walking together (2,543), approaching (942), ignoring (4,916), meeting one another (1,801), splitting up (879) and fighting (334). The numbers in brackets indicates how many frames contain this behaviour.

Results graphs

In similarity to the synthetic data the CRF method performs the best at smaller window sizes. However in this case the HMM method attains the highest averaged performance at a window size of 90. As the window size is increased there is an increase in classification performance for both the dynamic probabilistic models of the HMM and the CHMM. The CRF retains roughly the same performance, whilst performance using the LDA decreases slightly. In all cases the variability in performance increases slightly with an increase in window size.

When looking at the per class results (displayed in figure 4.12) there are some interesting patterns. When classifying one person approaching another (figure 4.12 (a)) the variation in classification is very high. In all cases except when using a HMM the accuracy decreases. For the HMM a decrease is followed by an increase at which point there is no further improvement. The improvement is in contrast to the other classifiers and suggests that the model is able to pick out approaching behaviour it has seen enough evidence. The HMM is able to better distinguish between approaching and ignoring (which gives the other methods the most trouble) after a window size of 30. This is most likely to its better temporal model compared to the CRF and LDA. The larger number of parameters in the CHMM combined with the small number of examples (942) could explain its poor performance.

In the case of fighting (figure 4.12 (b)) there is a systematic pattern which is followed by all classification methods. For all methods performance is poor but after a window size of 30 performance rapidly decreases for all methods. This is mainly to

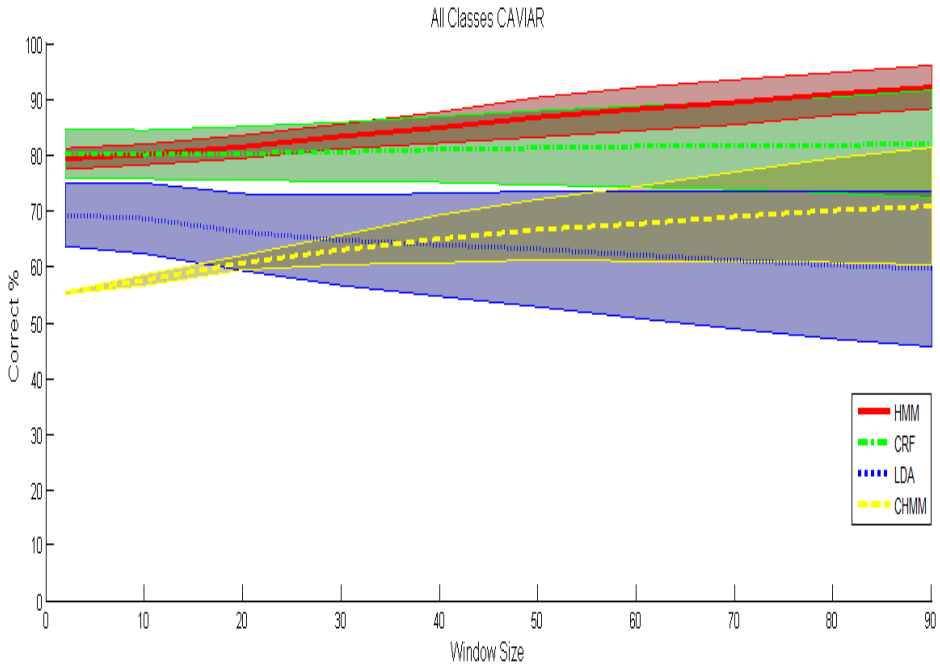


Figure 4.11: Overall results on the CAVIAR dataset for each method. Lines show averaged results (over 50 runs) whilst the shaded regions show one standard deviation.

do with the examples themselves. The actual fighting (ie somebody throwing a punch) is over very quickly, for some examples it takes less than a second. Larger window sizes seem to swamp examples with frames containing seemingly normal behaviour and so lose the significant but quick action. There are also relatively few examples of fighting (only 4 complete sequences). This may lead to there simply not being enough information to accurately learn a generalisable representation.

Classes ignore and meet (figure 4.12 (c) and (d)) display comparatively little variation as window size is increased. Oliver's CHMM method performs particularly well on the ignore class.

The results of classifying people splitting from one another are shown in figure 4.13 (e). With a HMM the results improve with a larger window size. Once the window size has increased beyond 75 there is no variance as the method seems capable of consistent perfect classification. The standard deviation decreases for both the CRF and for the CHMM at a window size of over 50. This indicated that enough time has passed to conclude that the people are indeed splitting. However there are a few cases where after walking away the person stops or turns around before carrying on again. This type of behaviour can lead to mis-classification for all methods but the HMM seems to be particularly robust to it. It should also be noted that there are relatively few

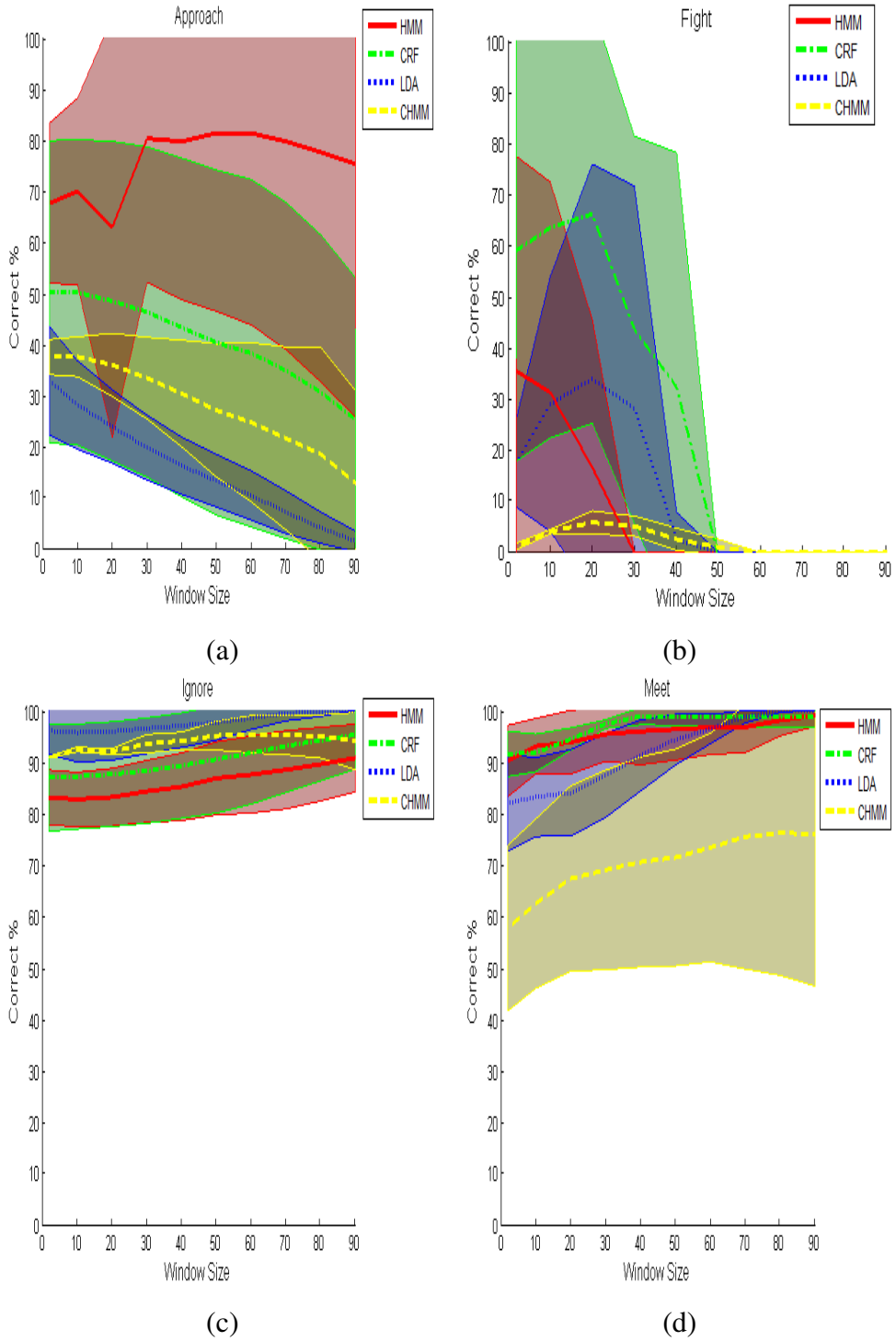
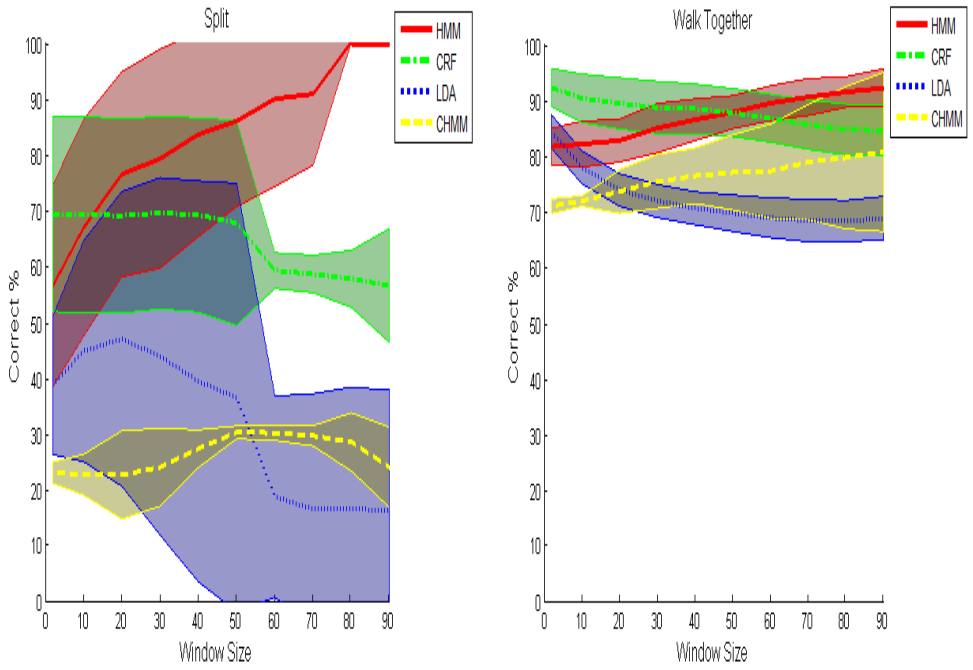


Figure 4.12: Per class results for the CAVIAR dataset showing performance of differing classification methods over varying window size. The classes shown are (a) approach, (b) fight, (c) ignore , (d) meet.



(e) (f)

Figure 4.13: Per class results for the CAVIAR dataset showing performance of differing classification methods over varying window size. The classes shown are (a) split, (b) walk together.

examples of splitting behaviour (879) within this dataset.

Walking together (figure 4.13 (f)) shows little variation in classification accuracy depending upon the window size.

Best Results : Confusion Matrices

The confusion matrices for the best results are given in figure 4.14 for HMM (a) and CRF (b) classification and in figure 4.15 for CHMM (a) and LDA (b) classification.

The HMM confuses walk with meet around 8% of the time but the biggest error is confusing splitting with meeting. The confusion may arise as again there is no feature which seems to be able to distinguish between cases where somebody moves away and then moves back and when they continually move away (as is the case for splitting). Examples for fighting are completely missed. This is most likely down the the very small sample size of this class.

For the CRF the method has problems confusing walk together and ignore. In addition split is often classified as two people ignoring one another. This can be understood as after some distance the two people appear to be ignoring one another if you can only

		True					
		Walk Together	Approach	Ignore	Meet	Split	Fight
Classified	Walk Together	0.92	0	0.002	0	0	0
	Approach	0	1.0	0	0	0	0
	Ignore	0	0	0.99	0	0	0
	Meet	0.08	0	0	0.86	0	1.0
	Split	0	0	0.004	0.14	1.0	0
	Fight	0	0	0.004	0	0	0
	Total Samples	479	1.0	1652	336	41	7

HMM - Best 97% at window size 90

		True					
		Walk Together	Approach	Ignore	Meet	Split	Fight
Classified	Walk Together	0.87	0	0	0	0	0
	Approach	0	0.43	0	0	0	0
	Ignore	0.09	0.57	1.0	0	0.51	0.35
	Meet	0	0	0	1.0	0	0
	Split	0.03	0	0	0	0.48	0
	Fight	0.01	0	0	0	0	0.65
	Total Samples	624	77	1243	356	87	17

CRF - Best 93% at window size 45

Figure 4.14: Confusion matrices displaying the best results for classification upon the CAVIAR dataset for the hidden Markov model and the conditional random field.

look a few frames into the past (ie its not possible to determine that the persons were previously together over the timescale (45 frames) given).

Like the CRF model the CHMM method appears to have problems distinguishing between walking together and ignoring one another. However in the case of the CHMM the error is greater. Approaching is often confused with ignoring. This may be to do with a large window size identifying people as getting nearer one another but who do not actually meet (eventually). The constrained exits and entrances to the scene mean that many people do get nearer one another before passing each other. Other methods using the new feature set do not display this problem. Again fighting classification poses a problem.

LDA performs the worst of all methods and confuses many of the same classes as

		True					
		Walk Together	Approach	Ignore	Meet	Split	Fight
Classified	Walk Together	0.70	0	0	0	0	0
	Approach	0.05	0.26	0.02	0	0	0
	Ignore	0.24	0.74	0.98	0	0.7	0.83
	Meet	0	0	0	0.97	0	0
	Split	0	0	0	0.03	0.3	0.17
	Fight	0.01	0	0	0	0	0
	Total Samples	841	70	708	834	185	128

CHMM - Best 78% at window size 90

		True					
		Walk Together	Approach	Ignore	Meet	Split	Fight
Classified	Walk Together	0.78	0	0	0.09	0.09	0.1
	Approach	0.05	0.39	0.04	0.02	0	0.34
	Ignore	0.04	0.61	0.92	0.01	0.29	0
	Meet	0	0	0	0.78	0	0.05
	Split	0.7	0	0.04	0.08	0.6	0.22
	Fight	0.06	0	0	0.02	0.02	0.29
	Total Samples	1140	390	1389	573	223	222

LDA - Best 75% at window size 4

Figure 4.15: Confusion matrices displaying the best results for classification upon the CAVIAR dataset for the coupled hidden Markov model and linear discriminant analysis.

the CHMM. However it seems to also have trouble classifying the meeting class.

Summary of Classification Upon the CAVIAR Dataset

Again it is visible that for small window sizes the CRF method offers the best performance. However in this dataset the HMM model gives the best possible average performance. Both the HMM and the CRF using the new proposed features show superior performance to Oliver's method. A particular problem for all methods is in the classification of fighting behaviour. A small sample size and the short timescale where any fighting actually occurs contribute towards this. We see that for some cases the window is too small (such as walking together and ignore) or too large (such as ignore and approach). A per class time window or an enhanced feature set would help this

problem.

One of the other features of this dataset is that for "approaching", "splitting" and "fighting" there were perhaps not enough examples to get a build a sufficiently generalisable model. A simple answer would be to say that more data is required. However in many real surveillance applications such an approach is not possible so showing how a method performs using only limited data is still of value.

4.7.2.3 BEHAVE Dataset

The BEHAVE dataset contains 134,457 frames which have labelled examples of interactions. Within this set there are 5 distinct classes which we seek to identify and classify. The 8 classes consist of examples of people: In a group (91,421), Approaching (8,991), walking together (14,261), splitting (11,046), ignoring (1,557), fighting (4,772), running (1,870) and chasing (539) one another. The numbers in brackets indicates how many frames contain this behaviour.

Results graphs

The overall averaged classification is shown in figure 4.16. The CRF clearly performs better than all other methods on this dataset for all window sizes. There is a slight increase in performance when the window size is increased when using a CRF. However the effect is more dramatic for both the CHMM, HMM and the LDA method. All three of these methods (HMM, CHMM, LDA) increase in performance as the window size is increased. Significant performance increases are observed between window sizes of 1 and 20.

Per class results (figures 4.17 and 4.18) display a similar story where increasing the window size has little effect upon CRF classification. When classifying splitting (figure 4.17 (a)), approaching (figure 4.17 (b)) and fighting (figure 4.17 (c)) increasing the window size improves the performance of the HMM, CHMM and the LDA classifiers. The HMM classifier gives the best performance of all methods when classifying fighting (figure 4.17 (c)).

Increasing window size also gives a similar increase in performance for both the in group classification (4.18 (a)) and the walking together class (4.18 (b)). However when classifying people in a group (4.18 (a)) the LDA method decreases in performance.

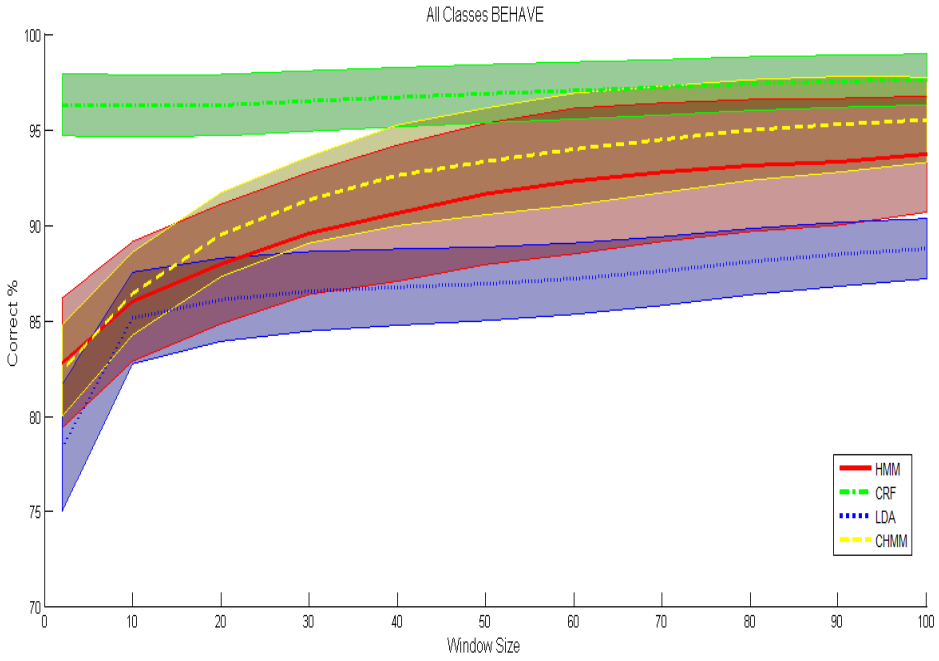


Figure 4.16: Overall performance on the BEHAVE dataset for each method. Lines show averaged results (over 50 runs) whilst the shaded regions show one standard deviation.

Best Results : Confusion Matrices

For the HMM classifier (figure 4.19 (a)) there is some confusion between being in a group and fighting. There are similarities in that a lot of the fighting occurs between groups of people in close proximity. Walking together and fighting are also confused. Again the method seems to be identifying the movement and close proximity but not the long term erratic trajectories that fighting produces. Splitting mainly gets confused with being in a group. At the initial split people are near each other so this can look similar to being in a group over a short time period. Interestingly the best performance attained by the HMM is higher than that of the CHMM (Figure 4.20 (b)) even though its average performance is worse.

For the CRF classifier (figure 4.19 (b)) the main confusion arises between confusing approaching and walking together and being in a group. This is likely to be caused by high similarities during the later stages of approaching when a person is almost in a group. The transitional effect may also be responsible for confusing the early stages of walking together and approaching one another (ie. to walk together). There also seems to be some confusion when classifying splitting behaviour. The main confusion seems to be with walking together. Again early stages of such sequences appear similar (ie. movement and near each other, at least for the initial stages of splitting).

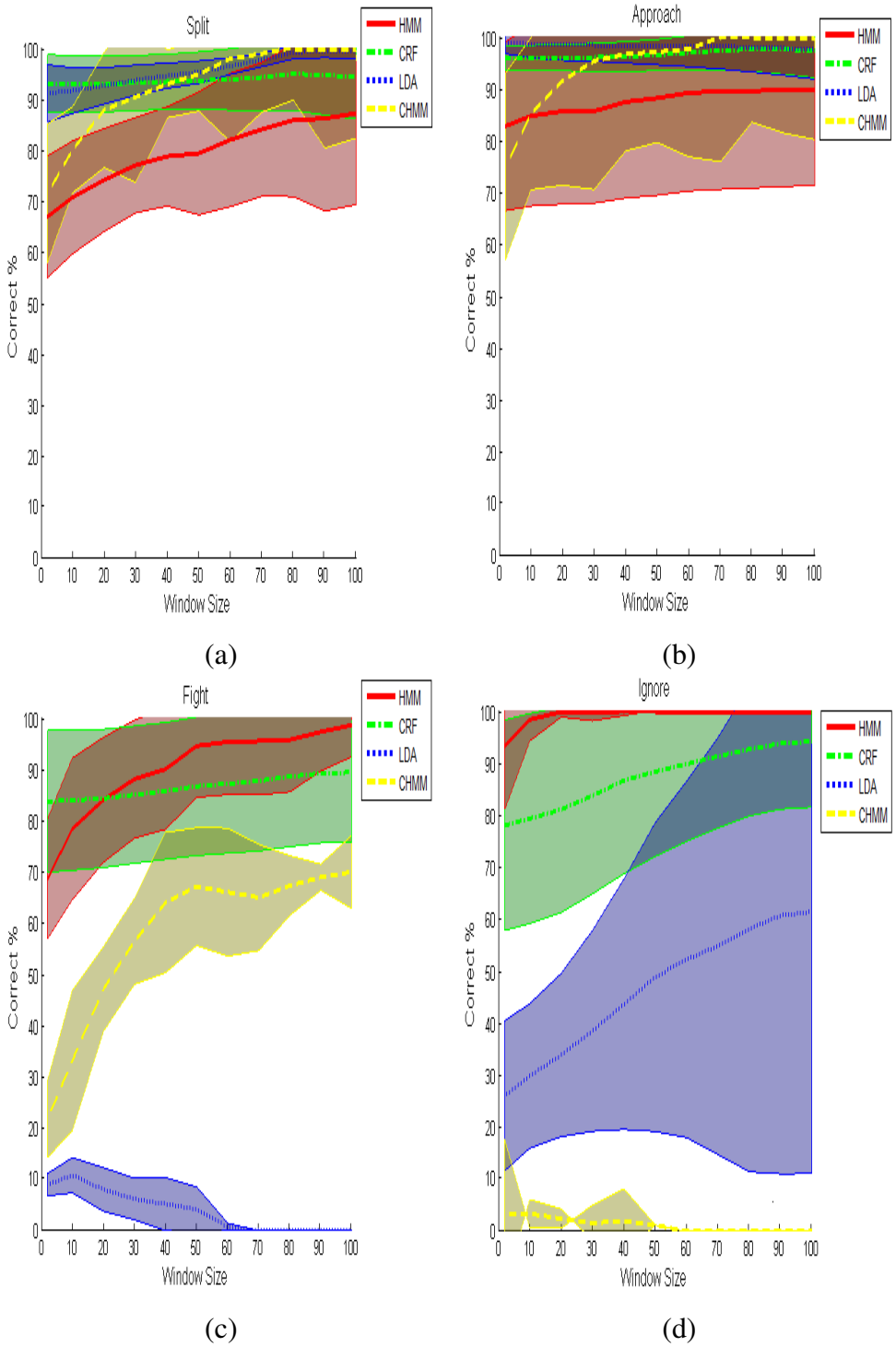
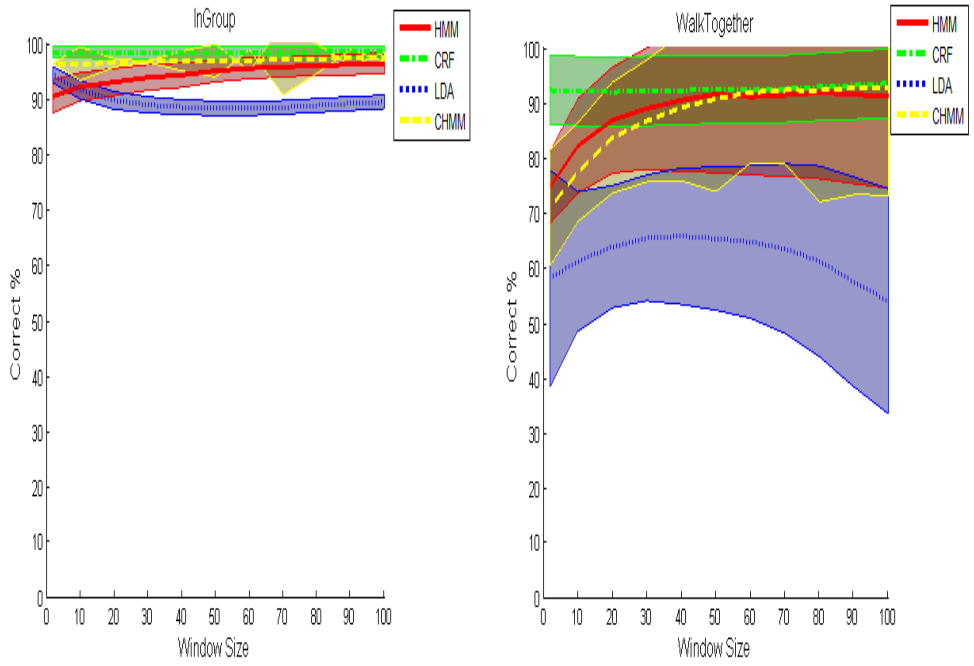


Figure 4.17: Per class results for the BEHAVE dataset showing performance of differing classification methods over varying window size. The classes shown are (a) split, (b) approach, (c) fight , (d) ignore.



(a) (b)

Figure 4.18: Per class results for the BEHAVE dataset showing performance of differing classification methods over varying window size. The classes shown are (a) in group, (b) walk together.

		True					
		In Group	Approach	Walk Together	Split	Ignore	Fight
Classified	In Group	0.94	0	0	0.17	0	0
	Approach	0	1	0	0	0	0
	Walk Together	0	0	0.96	0.02	0	0.05
	Split	0	0	0	0.73	0	0
	Ignore	0	0	0	0.08	1	0
	Fight	0.6	0	0.04	0	0	0.95
	Total Samples	1510	272	190	237	253	162

(a) - HMM - Best 94% at window size 32

		True					
		In Group	Approach	Walk Together	Split	Ignore	Fight
Classified	In Group	0.99	0.02	0	0.01	0	0
	Approach	0	0.96	0	0	0.04	0
	Walk Together	0	0.02	0.97	0.02	0	0.04
	Split	0	0	0	0.97	0	0
	Ignore	0.005	0	0	0	0.96	0
	Fight	0.005	0	0.02	0	0	0.96
	Total Samples	42176	3321	4994	3293	293	1628

(b) - CRF - Best 98% at window size 45

Figure 4.19: Confusion matrices displaying the best results for classification upon the BEHAVE dataset for the hidden Markov model and the conditional random field.

		True					
		In Group	Approach	Walk Together	Split	Ignore	Fight
Classified	In Group	0.97	0.03	0	0.02	0.27	0.15
	Approach	0.01	0.97	0	0	0.49	0.04
	Walk Together	0	0	0.9	0.02	0	0.14
	Split	0.01	0	0	0.95	0.17	0
	Ignore	0.01	0	0	0.01	0.01	0
	Fight	0	0	0.1	0	0.05	0.67
	Total Samples	40486	1656	4207	2305	895	1443

(c) - CHMM - Best 93% at window size 50

		True					
		In Group	Approach	Walk Together	Split	Ignore	Fight
Classified	In Group	0.94	0	0.09	0.01	0.27	0.31
	Approach	0.003	1	0.01	0	0.44	0.05
	Walk Together	0.2	0	0.71	0.07	0.03	0.47
	Split	0.007	0	0.02	0.87	0	0
	Ignore	0.01	0	0	0.01	0.24	0
	Fight	0.02	0	0.16	0.02	0.02	0.15
	Total Samples	44262	3029	4873	3529	758	1491

(d) - LDA - Best 88% at window size 10

Figure 4.20: Confusion matrices displaying the best results for classification upon the BEHAVE dataset for the hidden Markov model and the conditional random field.

As with the HMM and CRF, splitting and in group are confused using the CHMM (Figure 4.20 (a)). However the CHMM seems to have a great deal of trouble also classifying the ignore behaviour. Fighting is frequently confused with in group and approach which also have movement close to one another. The higher errors over multiple classes indicate that the feature set is unable accurately classify the data well.

LDA classification (Figure 4.20 (b)) also displays a high level of confusion across many of the classes. The worst classification results are for fighting (15% correct) and ignoring behaviour (24% correct).

Summary of Classification Upon the BEHAVE Dataset

It is again shown that the CRF method performs well when there is limited information available (a few frames). The CRF classifier also gives the best overall classification performance (figure 4.19 (b)) and the best averaged performance (figure 4.16) over all window sizes. However again we see that the method's temporal model is insufficient to cope with some situations (as highlighted by confusion between approaching, walking together and being in a group). A better model over a long time period will help in such instances. Oliver's CHMM method performs well and gives better average classification than the HMM and the LDA methods. However its performance is 5% less accurate that of the CRF classifier at best. For smaller window sizes (up to 40) this gap is up to 20%.

4.8 Window Size Classification

Based upon results in the previous section we can see that when classifying examples each class has an associated window size at which the classification is optimal. In this section we present an initial investigation to exploit that fact.

The original classification scheme as detailed in previous sections (4.4,4.5) is modified to allow each class to use a different window size when classifying an example. We select the window size per class as follows :

1. Divide the data into training and testing as before. Set aside some examples contained in the training set, label them the validation set.
2. Train model as before using only the training set.
3. For a range of window sizes classify the validation set.
4. For each class pick the best performing window size.
5. Final overall model consists of C class models each with its own window size (where C is the number of classes).

The classifier for each class may require a differing window size. In order to classify using this model the example must be of at least equal length to the largest window size selected.

When classifying a novel frame the following steps are taken. Each per class classifier is shown the number of frames required (as previously selected). For example if

class 1 has a window size of 10 then 10 frames will be presented to the classifier for class 1. Each class produces a likelihood and the class with the highest likelihood is taken.

Results using this classification method are shown in the following sections. The multiple window classifier is compared to the best achieved using only a single window across all classes. We also show the previous maximum best and the window size at which this was achieved. The previous best refers to just the best class result for a particular class at a particular window size. This number is included for comparison. The best per class result only displays the best performance achieved in this class but does not take into account the overall result (ie it is simply the best result for this class. eg if we only wanted to optimise this class then we would choose this window size for all classes). We average over 50 runs as before. All other parameters (eg hidden variables etc) are kept the same.

4.8.1 Synthetic Dataset

Results on the synthetic dataset are given in figures 4.21 and 4.22. Overall the multi window classifier performs at least 3% better over all methods. It is visible that for each class the multiple window method does not perform as well as the previous best recorded performance for that class. This is due to the multiple window classifier confusing classes due to the combination of multiple windows. It produces a different overall classifier which can confuse classes in a different way than before. In one way the best overall result is not obtainable as optimising upon a single window can affect performance upon another window size and thus reduce overall performance as is the case here. We can see ho using differing window sizes can improve upon the previous best performance by the generation of a new classifier as shown in figure 4.21 when using a CRF classifier. However outperforming the best previous on a particular class is not generally true.

4.8.2 CAVIAR Dataset

Tables giving results on the caviar data are given in figures 4.24 and 4.23. In similarity to the synthetic dataset the multiple window size classifier slightly outperforms the best result using a single window size. However there are individual cases where the best single window outperforms using class dependent window sizes. For example when using a HMM to classify splitting behaviour the single window method classifies

HMM				
Class	Single Window (70)	Multiple Window	Best Per Class	Best Window
Walk Together	0.78	0.93	0.8	40
Meet	1.00	100	1.00	20
Wait	1.00	0.99	1.00	2
Split	1.00	0.97	1.00	60
Follow	0.92	0.72	1.00	170
Overall	0.9	0.93		
CRF				
Class	Single Window (2)	Multiple Window	Best Per Class	Best Window
Walk Together	0.85	1.00	0.85	2
Meet	1.00	1.00	1.00	2
Wait	0.99	0.92	0.99	2
Split	0.99	0.95	1.00	10
Follow	0.92	0.94	1.00	100
Overall	0.92	0.96		

Figure 4.21: Results of using a variable window size for each class on the synthetic dataset. The best single window refers to the best result using a single window across all classes. Results when using a different size window to classify each class window are given in the Multiple Window column. The best possible classification result obtained for that particular class is given in 'Best Per Class'. Finally the best window size is given.

LDA				
Class	Single Window (200)	Multiple Window	Best Per Class	Best Window
Walk Together	0.44	0.54	0.48	2
Meet	1.00	1.00	1.00	2
Wait	0.88	0.80	0.98	2
Split	1.00	0.98	1.00	2
Follow	0.95	0.1	0.95	2
Overall	0.57	0.65		
CHMM				
Class	Single Window (70)	Multiple Window	Best Per Class	Best Window
Walk Together	0.54	0.65	0.56	110
Meet	1.00	1.00	1.00	160
Wait	0.99	1.00	1.00	120
Split	1.00	1.00	1.00	150
Follow	0.83	0.90	0.85	140
Overall	0.83	0.88		

Figure 4.22: Results of using a variable window size for each class on the synthetic dataset. The best single window refers to the best result using a single window across all classes. Results when using a different size window to classify each class window are given in the 'Multiple Window' column. The best possible classification result obtained for that particular class is given in 'Best Per Class'. Finally the best window size is given.

90.77% correct whereas the multiple window method only gives a 81.58% classification rate. This effect is likely because we only choose the best window size for each class and its effect upon other classes is not considered. This can mean that another classifier for a particular window size has produced a higher likelihood and so misclassified the example.

4.8.3 BEHAVE Dataset

The results upon the BEHAVE dataset were much closer with the largest difference in overall classification performance of 5.38% between the HMM models. When examining best window size which has been selected by the algorithm in figures 4.25 and 4.26 it is visible that the best performance occurs with a window size of 100 for many of the classes. Therefore selecting the best window size per class produces a similar classifier to one in which a single window size is used.

4.8.4 Summary

Using a different window size for each class produces a slightly superior overall classifier. The increase in performance ranged from 2% up to 8%. For cases such as the BEHAVE dataset where many of the optimal window sizes are the same this effect is not as pronounced.

Choosing the window size based upon only the performance of the class may not be the best way to determine the best window size. This is effect is seen when comparing the best per class performance (ie maximum obtained for this class) against the multiple window size performance. However the best class does not take into account the effect on other classes. By selecting a certain combination of window sizes within the overall classifier those test samples may be confused where previously they were not (when all classes had the same window size). Future work would investigate the best way to combine the best per-class window size.

4.9 Feature Set Comparison

Within this section we investigate the role of the new feature set (as given in equation 4.8), against that proposed by [Oliver et al., 2000a] (section 4.6.4). Experiments were performed using the window size classifier using features calculated as proposed in this thesis and those proposed by [Oliver et al., 2000a]. The per class window size was

HMM				
Class	Single Window (70)	Multiple Window	Best Per Class	Best Window
Walk Together	0.89	0.90	0.92	90
Approach	0.79	0.89	0.81	50
Ignore	0.87	0.80	0.90	90
Meet	0.96	1.00	0.98	90
Split	0.90	0.81	1.00	80
Fight	0	1.00	0.35	2
Overall	0.89	0.87		
CRF				
Class	Single Window (90)	Multiple Window	Best Per Class	Best Window
Walk Together	0.84	0.90	0.87	2
Approach	0.25	0.81	0.49	2
Ignore	0.95	0.87	0.95	90
Meet	0.99	0.88	0.99	50
Split	0.57	0.66	0.68	40
Fight	0	0.69	0.66	20
Overall	0.89	0.88		

Figure 4.23: Results of using a variable window size for each class on the CAVIAR dataset. The best single window refers to the best result using a single window across all classes. Results when using a different size window to classify each class window are given in the 'Multiple Window' column. The best possible classification result obtained for that particular class is given in 'Best Per Class'. Finally the best window size is given.

LDA				
Class	Single Window (90)	Multiple Window	Best Per Class	Best Window
Walk Together	0.65	0.73	0.81	2
Approach	0.02	0.43	0.32	2
Ignore	1.00	0.82	1.00	90
Meet	1.00	0.64	1.00	90
Split	0.11	0.73	0.42	20
Fight	0	0.19	0.34	20
Overall	0.58	0.65		
CHMM				
Class	Single Window (90)	Multiple Window	Best Per Class	Best Window
Walk Together	0.80	0.73	0.80	90
Approach	0.12	0.58	0.35	10
Ignore	0.93	0.92	0.95	60
Meet	0.75	0.84	0.76	80
Split	0.24	0.32	0.30	60
Fight	0.0	0.05	0.06	20
Overall	0.70	0.7		

Figure 4.24: Results of using a variable window size for each class on the CAVIAR dataset. The best single window refers to the best result using a single window across all classes. Results when using a different size window to classify each class window are given in the 'Multiple Window' column. The best possible classification result obtained for that particular class is given in 'Best Per Class'. Finally the best window size is given.

HMM				
Class	Single Window (60)	Multiple Window	Best Per Class	Best Window
In Group	0.95	0.99	0.96	100
Approach	0.89	1.00	0.90	100
Walk Together	0.91	0.94	0.92	80
Split	0.81	0.87	0.87	100
Ignore	1.00	1.00	1.00	50
Fight	0.95	0.80	0.98	100
Overall	0.92	0.97		
CRF				
Class	Single Window (2)	Multiple Window	Best Per Class	Best Window
In Group	0.98	0.99	0.98	100
Approach	0.96	0.99	0.97	80
Walk Together	0.92	0.97	0.93	100
Split	0.93	0.99	0.95	80
Ignore	0.78	1.00	0.94.01	100
Fight	0.84	1.00	0.89	100
Overall	0.96	0.98		

Figure 4.25: Results of using a variable window size for each class on the BEHAVE dataset. The best single window refers to the best result using a single window across all classes. Results when using a different size window to classify each class window are given in the 'Multiple Window' column. The best possible classification result obtained for that particular class is given in 'Best Per Class'. Finally the best window size is given.

LDA				
Class	Single Window (2)	Multiple Window	Best Per Class	Best Window
In Group	0.94	0.91	0.94	2
Approach	0.99	0.99	0.99	2
Walk Together	0.58	0.59	0.66	40
Split	0.91	0.73	0.99	100
Ignore	0.26	0.0	0.61	100
Fight	0.09	0.05	0.10	10
Overall	0.78	0.80		
CHMM				
Class	Single Window (60)	Multiple Window	Best Per Class	Best Window
In Group	0.97	0.98	0.97	100
Approach	0.97	1.00	1.00	80
Walk Together	0.92	0.92	0.93	100
Split	0.97	1.00	1.00	80
Ignore	0.0	0.07	0.03	2
Fight	0.65	0.7	0.7	100
Overall	0.94	0.95		

Figure 4.26: Results of using a variable window size for each class on the BEHAVE dataset. The best single window refers to the best result using a single window across all classes. Results when using a different size window to classify each class window are given in the 'Multiple Window' column. The best possible classification result obtained for that particular class is given in 'Best Per Class'. Finally the best window size is given.

LDA	Feature Set	
Class	Blunsden	Oliver
1	0.50	0.4
2	1.00	1.00
3	0.80	0.79
4	0.98	1.00
5	0.08	0.01
Overall	0.69	0.58

CRF	Feature Set	
Class	Blunsden	Oliver
1	1.00	0.61
2	1.00	0.91
3	0.92	1.00
4	0.95	0.99
5	0.94	1.00
Overall	0.96	0.85

HMM	Feature Set	
Class	Blunsden	Oliver
1	0.93	0.90
2	1.00	1.00
3	0.99	1.00
4	0.97	0.99
5	0.72	1.00
Overall	0.93	0.90

CHMM	Feature Set	
Class	Blunsden	Oliver
1	0.66	0.65
2	1.00	1.00
3	1.00	1.00
4	1.00	0.98
5	0.91	0.89
Overall	0.88	0.85

Figure 4.27: Comparison of classification performance between the feature set proposed in this thesis and that proposed by [Oliver et al., 2000a] on the synthetic dataset. The classes are (1) Walk Together, (2) Meet (3) Wait, (4) Split, (5) Follow.

the same for both sets of features. The results for the synthetic dataset are given in table 4.27 whilst the tables showing the comparison for the CAVIAR dataset and the BEHAVE dataset are given in tables 4.28 and 4.29 respectively.

When classifying the synthetic dataset (figure 4.27) the overall performance improved when using the new feature set by at least 1% and in some cases (CRF) by over 10%. The method showing the smallest difference is that of LDA, which performs poorly for both feature sets. The results are quite close however it should be remembered that the synthetic data set contains the cleanest data which corresponds to a simple movement model. This means that fewer features may be required to accurately represent it.

The most noticeable difference is between features sets when someone is fighting. Only recording the direction, speed and alignment seems to miss some of the high movements over a short space of time. The vorticity feature is an example of something which would pick this up in the new feature set. The new feature set seems better at

LDA	Feature Set	
Class	Blunsden	Oliver
1	0.73	0.55
2	0.43	0.37
3	0.82	0.69
4	0.64	0.81
5	0.73	0.69
6	0.19	0
Overall	0.65	0.58

CRF	Feature Set	
Class	Blunsden	Oliver
1	0.90	0.68
2	0.81	0.48
3	0.87	0.65
4	0.88	0.9
5	0.66	0.69
6	0.69	0.0
Overall	0.86	0.70

HMM	Feature Set	
Class	Blunsden	Oliver
1	0.9	0.88
2	0.9	0.78
3	0.80	0.87
4	0.99	0.91
5	0.82	0.15
6	1.00	0.0
Overall	0.88	0.82

CHMM	Feature Set	
Class	Blunsden	Oliver
1	0.74	0.70
2	0.59	0.51
3	0.92	0.89
4	0.84	0.76
5	0.32	0.45
6	0.06	0
Overall	0.71	0.65

Figure 4.28: Comparison of classification performance between the feature set proposed in this thesis and that proposed by [Oliver et al., 2000a] on the CAVIAR dataset. Classes are (1) Walk Together, (2) Approach, (3) Ignore, (4) Meet, (5) Split, (6) Fight.

LDA	Feature Set	
Class	Blunsden	Oliver
1	0.91	0.92
2	0.99	0.71
3	0.57	0.55
4	0.72	0.34
5	0.0	0.0
6	0.05	0.01
Overall	0.80	0.80

CRF	Feature Set	
Class	Blunsden	Oliver
1	0.98	0.98
2	0.99	1.00
3	0.96	0.94
4	0.99	1.00
5	1.00	0.83
6	1.00	0.95
Overall	0.98	0.98

HMM	Feature Set	
Class	Blunsden	Oliver
1	0.98	0.98
2	1.00	0.99
3	0.94	0.96
4	0.86	0.96
5	1.00	0.84
6	0.80	0.69
Overall	0.97	0.96

CHMM	Feature Set	
Class	Blunsden	Oliver
1	0.98	0.93
2	1.00	0.99
3	0.92	0.96
4	1.00	0.89
5	0.07	0.12
6	0.7	0.87
Overall	0.95	0.94

Figure 4.29: Comparison of classification performance between the feature set proposed in this thesis and that proposed by [Oliver et al., 2000a] on the BEHAVE dataset. Classes are (1) InGroup, (2) Approach, (3) Walk Together, (4) Split, (5) Ignore, (6) Fight, (7) Run Together, (8) Chase

dealing with real data which contains noise and less simplistic motion models which form part of the synthetic dataset.

The results are much closer (around 1% difference) when using the BEHAVE dataset 4.29. The dataset is much larger than the CAVIAR one with more examples so it is possible that the models can smooth out many of the short term variations which occur in the Oliver feature set which previously had to be encoded in the feature set (ie vorticity or features predicting a meeting).

4.9.1 Summary

The new feature set improves classification performance. However, it improves performance most when there are limited examples and where fast changes occur. Through-

out this thesis we have not considered feature reduction methods such as PCA. This is mainly because we want to be able to see how the features are used by the classifiers and to compare them fairly (for example if PCA based feature reduction was used then we wouldn't know if PCA was the reason or if another factor was at work when comparing with Oliver's method). However future work should investigate feature selection more thoroughly.

4.10 Conclusions and future work

Over all datasets the CRF classifier performs well (80-95%) when using limited information. The previous best method suggested by Oliver [Oliver, 2000] is improved upon using the CRF classifier in conjunction with the new proposed feature set. The new proposed feature set also outperforms Oliver's method when using a HMM model for a great many cases. When classifying the CAVIAR dataset the performance of the HMM classifier improves as the window size is increased until it becomes the best performing classifier.

The CRF classifier displays an ability to better classify data in the short term compared to the HMM. In contrast the HMM model improves more rapidly when the observation window size is increased suggesting it is better at smoothing the signal over longer sequences. The forward algorithm used to determine the likelihood seems to smooth the signal much better than the CRF. The case of the CHMM the more gradual improvement could be attributable to the larger number of parameters which requires more data in order to represent the data adequately. A suggestion for future work would be therefore to improve the long term temporal model of the CRF that we are using. It should be noted that these comments about the CRF model apply for the single chain structure which is used here. There are many architectures which can be used within the CRF framework. A higher order CRF may produce a better temporal model and so we would expect to see larger improvements when the observation window size is increased. However a CRF model will always be discriminative compared to the HMM and CHMM's generative ability. The ability to generate samples may be important in certain cases (such as estimating a model's complexity [Gong and Xiang, 2003b]) however this ability is not required for the classification tasks as presented here.

Throughout all experiments on all of the datasets it is visible that there seems to be an optimal window size for classification of a particular class. For some activities such as fighting in the CAVIAR dataset (4.12 (b)) the window size is quite short (due to the

speed of a fight) where as for other classes such as the ignore behaviour (figure 4.17 (c)) a longer window size improves performance. An initial investigation into using per class window sizes was presented however other ways of combining different window sizes could be investigated. Other feature sets or feature selection algorithms could be investigated.

Chapter 5

Pre-Fight Detection

5.1 Introduction

This chapter investigates pre-fighting situations as viewed from a video camera within a surveillance domain. The main aim is to establish the feasibility of detecting fighting situations. Here, we are also interested in investigating the possibility of detecting pre-fighting situations. Pre-fighting is useful in surveillance situations where the timely intervention of a CCTV operator could avoid a potentially criminal situation and thus prevent an escalation of violence. The main approach taken is to detect rough movement based visual features and to use them to represent the activity that is occurring. Specifically, we follow a similar approach to Dollar *et al.* [Dollar et al., 2005] who use spatio-temporal features calculated throughout image sequences. We do not try to model each limb or hand of each person as this is not possible to perform reliably with the high degree of movement and occlusion which is prevalent in such situations. Once the features have been extracted from the sequence per-class clustering is then applied in order to form a dictionary of cluster centres, as detailed in section 5.5. This dictionary is used to generate a histogram based description of an image sequence. Histogram representations are then supplemented by other features (section 5.4.2).

A method to determine the best structure for classification is also presented using a hierarchical AdaBoost algorithm (section 5.6).

5.2 Contribution

This chapter presents work upon the feasibility of classifying pre and post fighting situations in addition to identifying actual fighting and non fighting (normal) situations.

It is shown that such classification is possible. A way to identify a (hierarchical) structure for classification is presented along with a generalisation which makes it possible to apply the method to other datasets containing more classes.

5.3 Previous and Related Work

Human ability to predict dangerous or criminal activities from CCTV has previously been investigated by Troscianko *et al.* [Troscianko et al., 2004]. In their work participants from either an expert or a non-expert group were shown videos from CCTV cameras. At a particular point in time the video was paused and the participants were asked to predict on a scale of 1 to 5 if they thought a dangerous act would be committed by an individual or individuals in the video.

There were 16 videos displaying criminal behaviour with a further 16 matched as closely as possible to the situation where nothing happened. Human performance classified 80% of criminal incidents correctly with 65% of normal but similar incidents matched correctly. Dee and Hogg [Dee and Hogg, 2004a] also investigate human performance using a computational model and found correlations in the rating of 'interestingness'.

The most similar work to ours is that of Datta *et al.* [Datta et al., 2002] who detect person on person violence using a range of measures derived from a background removed and segmented representation of the person. The measures include acceleration and jerk along with the leg and arm orientations. All are computed from a side on point of view and results indicate good performance on their dataset of 62 situations with a correct classification of 97%.

Cupillard *et al.* [Cupillard et al., 2002] also investigate fighting situations within the domain of Metro surveillance. They use pre-defined templates of activity to match the on screen activity and classify the image sequence.

Ribeiro *et al.* [Ribeiro and Santos-Victor, 2005] also attempt to classify what a person is doing within the CAVIAR dataset using a hierarchical feature selection method. Others such as Davis and Bobick [Davis and Bobick, 2001] used moments based upon a stabilised silhouette image to classify more general motion. Efros *et al.* [Efros et al., 2003] used a optical flow based similarity measure to match different persons actions.

Within this work we seek to improve upon previous work by not requiring background subtraction [Datta et al., 2002, Ribeiro and Santos-Victor, 2005, Davis and Bobick, 2001] or detailed identification of the person's limbs and other details [Datta

et al., 2002]. The use of interest point features also allows partial matching in the presence of occlusion and a smaller number of total samples (whole person matching such as [Efros et al., 2003, Davis and Bobick, 2001] has been shown to require a large number of samples [Robertson and Reid, 2005]).

5.4 Temporal Features

This section describes the features that are extracted from each sequence. Sequences have been labeled as belonging to one of four classes, either fighting, pre-fighting, post-fighting or normal behaviour. From the original (manual) bounding box tracking each box is scaled to a uniform size of $x = 30, y = 90$. This size was chosen as it is the average size of all bounding boxes over this particular dataset. An example of the re-scaled data is shown in figure 5.1, along with the corresponding response function (described below). The reason for the re-scaling is so that all bounding boxes can be described in the same way and to make the process as scale invariant as possible. It is apparent that the bounding boxes contain other information such as background and occlusions from other people.

5.4.1 Features

We make use of Dollar *et al.*'s [Dollar et al., 2005] approach to sequence representation as it has been previously successful [Niebles et al., 2006, Dollar et al., 2005], can deal with occlusions and does not require background subtraction. The method is briefly reviewed here.

Dollar *et al.* [Dollar et al., 2005] developed a spatio-temporal response function for classifying sequences of behaviours. Their approach assumes a stationary camera (or that the effects of camera motion can be compensated for). The response function is given in equation (5.1).

$$R = (I \otimes g \otimes h_{ev})^2 + (I \otimes g \otimes h_{od})^2 \tag{5.1}$$

The 2D smoothing Gaussian function $g(x, y, \sigma)$ is applied only along the spatial dimensions of the image sequence I. The two functions h_{ev} and h_{od} are a pair of Gabor filters which are defined as $h_{ev}(t; \tau, \omega) = -cos(2\pi t\omega)e^{-t/\tau^2}$ and $h_{od}(t; \tau, \omega) = -sin(2\pi t\omega)e^{-t^2/\tau^2}$. They are applied along the temporal dimensions of the image sequence. Throughout all experiments we set $\omega = 4/\tau$. This gives the response function

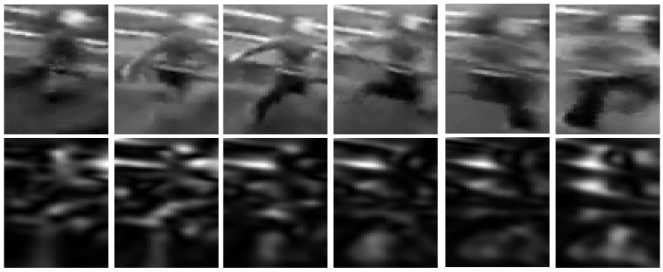


Figure 5.1: The scaled original image (top row) along with the corresponding response image (R - equation 5.1) bottom row.

two parameters corresponding to the spatial scale (σ) and the temporal scale (τ). They were set to ($\tau = 3, \sigma = 3$) throughout all experiments. This follows on from work by Dollar [Dollar et al., 2005] and separately Niebles *et al.* [Niebles et al., 2006] who found that the $3 \times 3 \times 3$ spatial and temporal resolution was sufficient for action recognition. Only those responses above a threshold value are recorded. Examples of feature responses are shown in figure 5.1.

From these response functions a cuboid descriptor is formed. This is a three dimensional cuboid formed from the original image sequence in space and time. It consists of all (greyscale) pixel values within an area of six times the scale at which it was detected. Only those regions where the response is above a certain threshold are used. The descriptor is the cuboid volume of the original intensity image centred at the points selected from the response image.

5.4.2 Other Features

In addition to features derived purely from local visual information, additional features were calculated from the tracking process. One feature which is not possible to get from the re-scaled image information is the amount of actual movement of a person over the sequence. This movement is averaged over the length of the sequence:

$$d_i = \frac{\left\| \mathbf{p}_1^i - \mathbf{p}_T^i \right\|}{T} \tag{5.2}$$

$\mathbf{p}_t^i = \left[x_t^i, y_t^i\right]$ is the position (in image coordinates) of the i^{th} person at the t^{th} timestep. T is the number of timesteps in the whole sequence. In addition to positional features we also include a measure of how active the sequence is. To do this we sum over all response images in the sequence ($90 \times 30 \times T$) the absolute value of the response image $\mathcal{R}$. This is then used to give the mean response ($\mathcal{R}_\mu$). The standard deviation ($\mathcal{R}_\sigma$)

is also taken and used as a feature. These additional features were found to improve performance.

5.5 Sequence Representation

Each sequence generates a set of cuboids (as detailed in section 5.4.1). Each pre-identified class (fighting, pre-fight, post fight and normal) generates a large number of cuboids over all sequences. From this large number of cuboids a smaller set is sub-sampled (using random sampling) so that these cuboids can be clustered. K-means clustering [Duda et al., 2000] is used to identify k cluster centres (using the Euclidean distance as a similarity metric). Clustering was performed per class, with the final dictionary consisting of all clusters concatenated into one. The question of how many cluster centres (k) to use is investigated in section 5.7.1.1.

For each sequence a histogram is created based upon the previously learned cluster centres. The response function (equation 5.1) is applied throughout the complete sequence. Cuboids are then generated from the complete sequence as described in section 5.4.1.

For each cuboid in the new sequence the nearest cluster within the learned cluster centre dictionary is found. A histogram is then made of all matches throughout the sequence. This histogram is then normalised. Examples of image sequences and their corresponding histograms are given in figure 5.2.

In addition to the histogram the features, as described in section 5.4.2, are included in the sequence representation. This gives a final representation ($\mathcal{S}_i$) of sequence i :

$$\mathcal{S}_i = \left[h_i \mid d_i, \mathcal{R}^i_\mu, \mathcal{R}^i_\sigma\right] \tag{5.3}$$

h_i is the i^{th} histogram and d_i the distance as given in equation 5.2 for the i^{th} sequence. The mean ($\mathcal{R}^i_\mu$) and standard deviation ($\mathcal{R}^i_\sigma$) of the response image sequence ($\mathcal{R}$) for the i^{th} sequence make up the other additional features.

5.6 Classification

Our aim is to investigate whether it is possible to detect dangerous situations involving fighting and within that if it is possible to further tell if a fight is likely to occur or has just finished (the usefulness of post fight situations arises from the fact that cameras

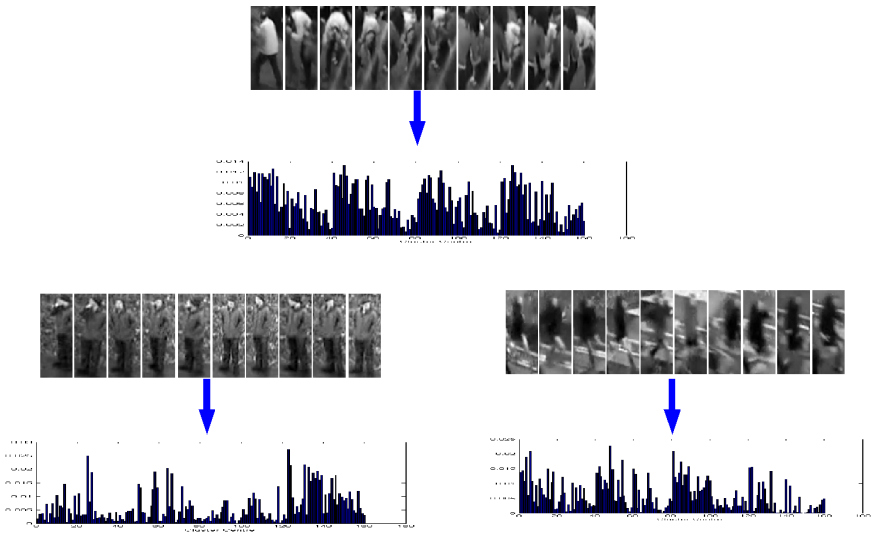


Figure 5.2: Examples of sequences along with the corresponding histogram representation. The histograms are computed over the entire sequence. Top is a fighting sequence, bottom left is a normal sequence and bottom right is a post fighting sequence. The fixed size histograms are composed from the whole complete sequence, whose length can vary. The histograms are normalised to unit weight

may not cover the entire area to capture the actual event but it would be useful to detect such situations).

Here AdaBoost [Freund and Schapire, 1996] is used to classify each sequence based upon the sequence representation. The implementation of AdaBoost uses a decision tree classifier as a weak learner. By using a decision tree we hope to exploit cases where the co-occurrence of different cluster centre responses can help in establishing the class of a sequence. An example of this could be a response indicating fast lower body movement occurring at the same time as a response indicating fast upper body movement.

The approach also differs from a standard AdaBoost classifier in that we employ a hierarchical classification method. Such a hierarchy is preferable to multiclass AdaBoost (such as that used by Zhu *et al.* [Zhu et al., 2006]) due to the nature of the problem itself. We are trying to group similar things together to make them easier to classify.

5.6.1 Hierarchical AdaBoost

Within this section we introduce the workings of the algorithm and then present an extension to allow us to classify the data. AdaBoost is used as it is one of the best classifiers available when dealing with lots of small features. We use a weak classifier (small decision tree) which makes use of both the strength of response in the histogram and may use multiple cluster centres simultaneously to classify an action. The possibility of using two or more cluster centres together means that there is a very large set of possible features. AdaBoost is capable of working with such a large feature set and producing good results. It is for this reason that we make use of it.

A general overview of the proposed method is that all possible partitions of positive and negative examples from the data are generated. For example a possible partition could be to treat classes 1 and 2 as being positive and classes 3 and 4 as negative. The next level down in the tree is made up of all possible partitions of classes from the parent. This level is then partitioned until no more partitioning is possible. In this example the next level down would contain leaf nodes for each class and the algorithm would terminate. This process continues until no examples remain. In section 5.7 we compute all possible trees which can be constructed in this way. The best tree (as evaluated using validation data) is then used for classification of the previously unseen test data.

The hierarchical component is added for two reasons. The first is that we would like to be able to interpret our final classifier in some way. For example the ability to check that the final structure is in some way aligned to our beliefs about how to classify the problem (one problem with many classifiers is you get a list of numbers out of them but the meaning of such numbers can easily be hidden). The second is perhaps far more important should one wish to pursue this approach in a real application. By using a hierarchical approach it is possible to refine the behaviour you are looking for. For example a surveillance operator may want to find all examples of fighting behaviour but then further split it into pre/post fight (as we do here) and even further into fights which start with a chase. Using a hierarchy will give such abilities whilst retaining the ability to quickly classify a broad range of behaviours of interest.

5.6.1.1 AdaBoost

The AdaBoost algorithm was introduced by Freund and Schapire [Freund and Schapire, 1996]. AdaBoost is an algorithm for constructing a strong classifier ($f(x)$) from a lin-

Algorithm 4 AdaBoost algorithm

Given: $(x_1, y_1), .., (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, 1\}$

Initialise weights $D_1(i) = \frac{1}{m}, i = 1, .., m$

For $t = 1, .., T$

Find the classifier $h_t : X \rightarrow \{-1, 1\}$

with minimum error w.r.t. distribution D_t

$h_t = \min_{h_j \in \mathcal{H}} \varepsilon_j$, where $\varepsilon_j = \sum_{i=1}^{m} D_t(i) \left[y_i \neq h_j(x_i) \right], \mathcal{H} = \{h(x)\}$

if $\varepsilon_t > 0.5$ then stop

Choose $\alpha_t \in \mathcal{R}$

Update $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$,

Z_t is a normalisation constant chosen so D_{t+1} is a

probability distribution (eg, sum over all x).

Output strong classifier

$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

ear combination of weak classifiers ($h_t(x)$) as shown in equation (5.4).

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{5.4}$$

The algorithm for calculating these weights (α_t) along with the final strong classifier $H(x)$ is given in algorithm 4.

The weak classifier $h(x)$ we use is a decision tree. The indicator function $\left[y_i \neq h_j(x_i) \right]$ evaluates to 1 or 0 depending whether the condition is true or not. The weighting α_t is set as in equation (5.5), with r_t is the weighted error rate of classification.

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right) \tag{5.5}$$

5.6.1.2 Hierarchy

To discover the best structure (in terms of classification performance) the set of $\mathcal{P}$ possible hierarchical partitions of the classes was created. At each level within the hierarchy we look at all possible partitions of the binary class labels. The number of possible partitions at a particular leaf is given in equations (5.6) and (5.7) :

Algorithm 5 Algorithm to build all trees given a set of classes.

T=**BuildTree**($\mathcal{C}$)
$T = \{\}$
P=**allPartitions**($\mathcal{C}$)
for each $(l,r) \in \mathcal{P}$
    a_l=**BuildTree**(l)
    a_r=**BuildTree**(r)
for each $t_l \in a_l$
    for each $t_r \in a_r$
        $T = T \cup \{(t_l, t_r)\}$

$$f(N,k) = \begin{cases} \binom{N}{k}/2 & if, k = \frac{N}{2} \\ \binom{N}{k} & otherwise \end{cases} \tag{5.6}$$

$$\|\mathcal{P}\| = \sum_{k=1}^{\lfloor (N/2) \rfloor} f(N,k) \tag{5.7}$$

N is the number of possible classes (in our case totaling four). The case where $k = \frac{N}{2}$ removes mirror partitions (ie partitions which are the same but simply swapped between the right and left side) from the set of partitions. For this four class problem there are $\|\mathcal{P}\| = 7$ initial possible partitions : (([1][2 3 4]),([1 2][3 4]),([1 3][2 4]),([1 4][2 3]),([2][1 3 4]),([3][1 2 4]),([4][1 2 3])). Another partition is calculated at every node of the tree from the classes assigned to that node until each node has only one class.

The hierarchical model starts with a set of all possible partitions $\mathcal{P}$ of the set of all class labels ($\mathcal{C}_n$) at the current node n. Each of these partitions (p_n) has a left (l) and right (r) branch such that:

$$p_n = \{l_n, r_n\} \tag{5.8}$$

$$l_n \subset \mathcal{C}_n \tag{5.9}$$

$$r_n = \mathcal{C}_n \backslash l_n \tag{5.10}$$

The method proceeds as shown in algorithm 6. All possible trees are built using all possible classes using the build tree function as given in algorithm 5. For each of

Algorithm 6 Evaluation of all trees

T=**BuildTree**(all Classes)

for each $p_i \in T$

    d_i = **HierarchicalAdaBoost**(l_i, r_i)

    for all $v_j \in \mathcal{V}$

        $\phi_{i,j}$ = **Classify**(d_i, v_j)

i=arg max$_i$ **assessment**$(\{\phi_{i,j}\})$

Algorithm 7 Hierarchical AdaBoost, d_i - Tree at ith level

d_i = **HierarchicalAdaBoost**(l_i, r_i)

if $|l_i| > 1$

    d_l = **HierarchicalAdaBoost**(l_i)

if $|r_i| > 1$

    d_r = **HierarchicalAdaBoost**(r_i)

x=**AdaBoost**(l_i, r_i)

$d_i = (x, d_l, d_r)$

the trees the hierarchical AdaBoost algorithm is run to learn the parameters (d_i) of the AdaBoost classifier. The validation data V is then assessed and the best tree structure is chosen. The pseudo code for the hierarchical AdaBoost learner is given in algorithm 7. Classification of data is shown in algorithm 8.

The purpose of the proposed algorithm is to estimate the structure of the decision making process involved in classification i.e., what is the best structure for the decision tree that does the classification. This leads to considering which classes should be merged or split at each decision level. It is hoped that by testing possible structures of the classification better performance can be obtained. Similar events could also be grouped and classified together at a higher level (eg fighting behaviour) whilst lower levels can provide more fine grain information such as pre-fighting behaviour.

A note about the general applicability of such a brute force approach to obtaining the structure should be given. The algorithms will always converge to a tree structure as at each level the remaining classes are divided until there no possible partitions left. However practical computing limitations mean that only a finite number of classes can be used before memory limits are reached. Equation 5.7 gives details of the number of partitions required for a specific number of classes to classify.

Algorithm 8 Data classification. V are the data samples. EvalLeft is the evaluation function which determines whether the data V is in the left or right partition.

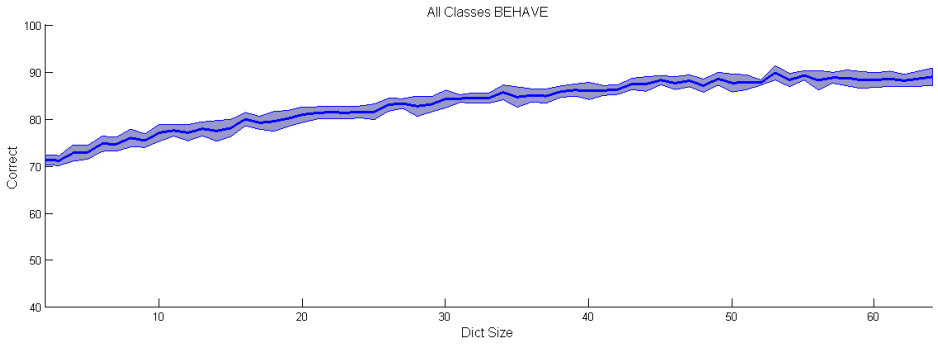
C=**Classify**(d, v)
if **EvalLeft**(d, v)
    if **leafNode**(d_l)
        $c = d_l$
    else
        $c =$**Classify**(d_l, v)
else if **leafNode**(d_r)
    $c = d_r$
else
    c=**Classify**(d_r, v)

5.7 Experiments and Results

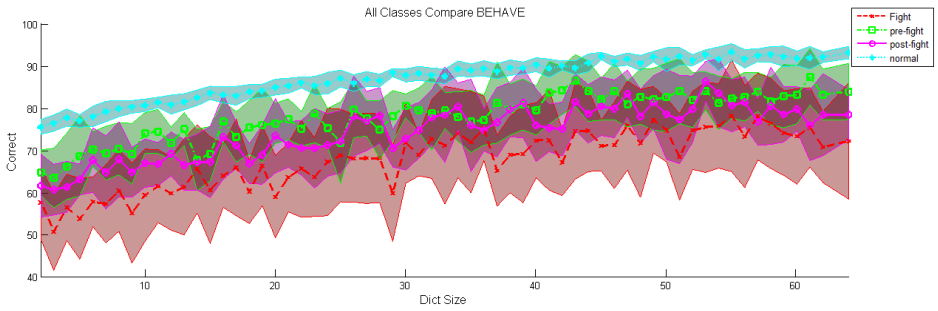
We used the publicly available BEHAVE [Blunsden et al., 2007b] and CAVIAR [project/IST 2001 37540, 2004] datasets. This sequence contains examples of multi-party interactions. There are over 70,000 frames within this dataset showing ten types of interaction. We label each person as belonging to one of four classes (fighting, pre-fighting, post fighting or normal). The dataset consisted of 1,138 complete sequences. Results on the smaller CAVIAR dataset are also presented. This dataset contains 56 complete sequences taken in the lobby of the INRIA Labs at Grenoble, France.

5.7.1 Classification of Complete Sequences

These experiments are similar in spirit to those of Troscianko *et al.* [Troscianko et al., 2004] who tested human ability to detect dangerous situations by using complete pre-segmented sequences prior to asking the question: what happens next? Here the complete test sequences of varying lengths are used to test the algorithm's performance. First the question of optimal dictionary size is investigated. The best performing dictionary size is then used to classify whole sequences and results are presented and discussed. We use two publicly available datasets to test the method. First we use the small scale CAVIAR dataset [project/IST 2001 37540, 2004] before also demonstrating the approach upon the BEHAVE dataset [Blunsden et al., 2007b].



(a)



(b)

Figure 5.3: The effect of dictionary size on classification performance on the BEHAVE dataset. Overall results are shown in (a) whilst per class results are given in (b). For figure (b) Blue dots denote normal behaviour, red dashes and crosses denotes fighting. Green squares with dashes and dots denotes pre-fighting whilst purple circles denote post fighting behaviours. For both graphs the shaded areas represent standard deviation.

5.7.1.1 How many cluster centres should the dictionary contain ?

Here an investigation into the optimal number of cluster centres is presented. For this experiment the number of cluster centres used to construct the dictionary was varied. For each dictionary size the dictionary was created as detailed in section 5.5. At each dictionary size the classifier was run 50 times over a different subset of training and testing data. The averaged performance for each dictionary size is shown in figure 5.3.

The graph shows that the best performance occurs when a dictionary size consisting of between 40 and 60 clusters is used. There is almost no improvement in performance for any of the classes when the dictionary is increased beyond 50 clusters. For this reason a dictionary consisting of 50 clusters is used through all subsequent experiments on the BEHAVE dataset. This size of dictionary is relatively small suggesting that there are few but important clusters which are sufficient to classify the data. When

more clusters are added this does not improve performance. This is most likely caused by the effect of adding clusters that occupy a similar region of the feature space. When this is converted to a histogram instead of having a few strong responses there are many weak responses which impair the ability of the classifier to further distinguish between sequences. As we make use of an AdaBoost classifier which uses a tree model as a weak learner it tends to over specialise when there are many features representing similar features resulting in no observable performance increase.

Results for the CAVIAR sequence are also given below in figure 5.4. Here the effect of the dictionary size is less pronounced. One of the reasons for this is that the majority of the examples within the dataset are from the normal class. There are very few examples (4) of fighting within this dataset. In the following section the per class results are given. With so few examples one may question why we pursue the classification problem on this dataset. One of the reasons is that this distribution of examples is more representative of the distribution of real data where fighting events are even rarer but we still want our algorithms to be able to capture and classify them.

Based upon figure 5.4 we use a dictionary of size 50 for all subsequent experiments upon the CAVIAR dataset.

An alternative approach could be to use one of the model order selection methods. Other authours such as [Gong and Xiang, 2003a] make use of metrics such as the Bayesian information criterion (BIC) to evaluate competing models. Such an approach is desirable when evaluating model complexity or making use of on-line updates of the model to determine its representation.

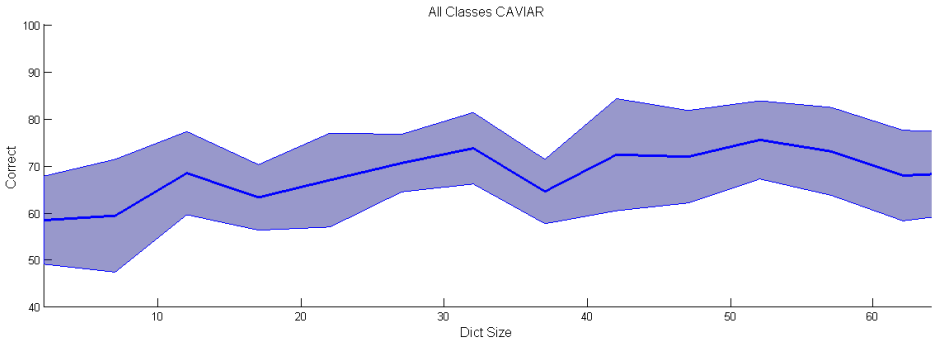
Due to the small number of parameters we want to set (cluster centres) it is possible to enumerate all feasible combinations (in this case feasible means before memory requirements become prohibitive).

5.7.1.2 Results

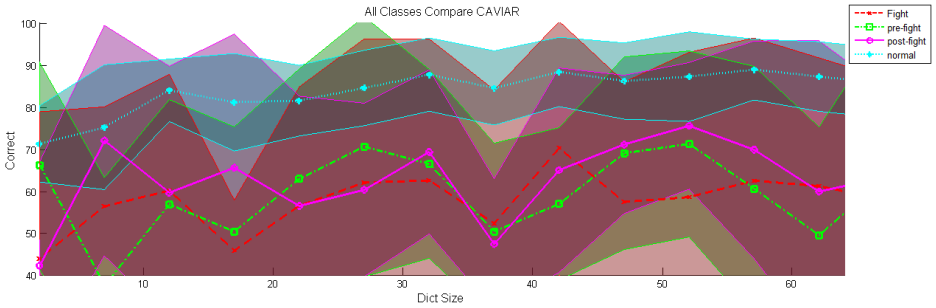
Results on the BEHAVE dataset

The classification tree was constructed by first separating the training and test data into two distinct and equal sized sets. The data was separated per sequence so that training samples were not taken from the same sequence as those used for testing. The best tree as determined by our method over a number of runs is given in figure 5.5. Confusion matrices for this tree are given in figure 5.6.

This tree gives an overall classification performance of 89.9% correct classification



(a)



(b)

Figure 5.4: The effect of dictionary size on classification performance on the CAVIAR dataset. Overall results are shown in (a) whilst per class results are given in (b). For figure (b) blue dots denote normal behaviour, red dashes and crosses denotes fighting. Green squares with dashes and dots denotes pre-fighting whilst purple circles denote post fighting behaviours. For both graphs the shaded areas represent standard deviation.

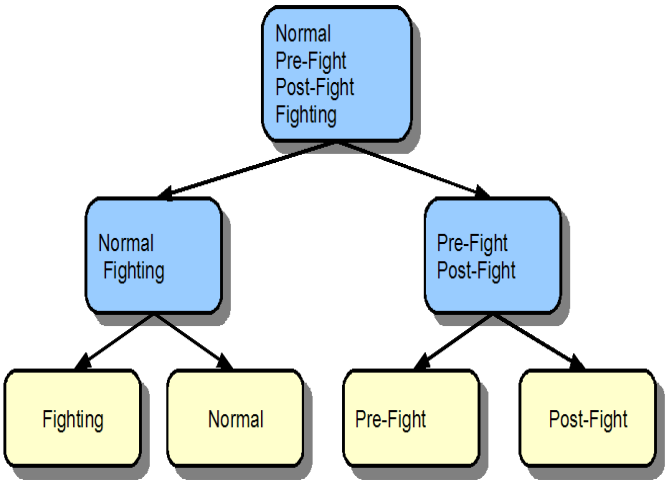


Figure 5.5: The final classification tree. Shaded nodes show the classes from which partitions of the data are formed.

		True			
		Fight	Pre-Fight	Post Fight	Normal
Classified	Fight	0.96	0.02	0.06	0.02
	Pre-Fight	0.04	0.88	0.08	0.01
	Post-Fight	0	0.08	0.78	0.01
	Normal	0	0.02	0.08	0.96
	Total Samples	25	97	65	382

(a)

		True	
		Fighting Related	Normal
Classified	Fighting Related	0.96	0.04
	Normal	0.04	0.96
	Total Samples	187	382

(b)

Figure 5.6: Confusion matrix for classification of sequences. (a) Shows the performance treating each class individually whilst (b) shows results with all fighting behaviour aggregated. Results are for the BEHAVE dataset.

with a standard deviation over multiple runs of 0.019. The confusion matrices for classifying individual classes and all fighting behaviour as one is given in figure 5.6(b). For normal vs fighting behaviour correct classification is at 96%. The structure groups post and pre fight behaviour together suggesting that there is a high degree of similarity between them.

When grouping all fighting based behaviour together the performance increases substantially. It is useful to show performance for such normal vs non-normal behaviour as there are many applications to surveillance situations. The cases where a fighting situation is classified as normal is relatively low with much of the confusion arising between pre, post and actual fighting.

Results on the CAVIAR dataset

For the smaller dataset the results are also promising. However it should be noted that the number of fighting examples is significantly less then examples from the BEHAVE dataset. Again when grouping all the fighting situations together (pre/post and actual fighting) the results improve significantly. None of the fighting situations are confused

		True			
		Fight	Pre-Fight	Post Fight	Normal
Classified	Fight	1	0	0	0.11
	Pre-Fight	0	1	0.2	0
	Post-Fight	0	0	0.8	0
	Normal	0	0	0	0.89
	Total Samples	2	3	5	18

(a)

		True	
		Fighting Related	Normal
Classified	Fighting Related	1	0.11
	Normal	0	0.89
	Total Samples	10	18

(b)

Figure 5.7: Confusion matrix for classification of sequences for the CAVIAR dataset. (a) Shows the performance treating each class individually whilst (b) shows results with all fighting behaviour aggregated.

with a normal situation. Overall performance is 89.3% with again a very small standard deviation of 0.1. The overall accuracy rises to 92.9% when considering all fighting vs no fighting situations. The tree retains the same structure as the one above and so is not reproduced here.

However some normal situations are misclassified as a fight situation. This is perhaps to do with some of the fighting scenes being acted out rather than being actual fights. Some of the scenes where a people are walking together and meeting one another can look similar to fighting scenes within this dataset. It is often the pre and post fight behaviour which also helps to identify a fight something which the normal sequences do not display.

5.7.2 Labeling of Continuous Sequences

A further experiment was conducted whereby sequences were not pre-segmented but instead a continuous video stream was presented to the classifier. This task is much harder than using pre-segmented sequences due to the high degree of overlap between different classes as they transition from one to the other.

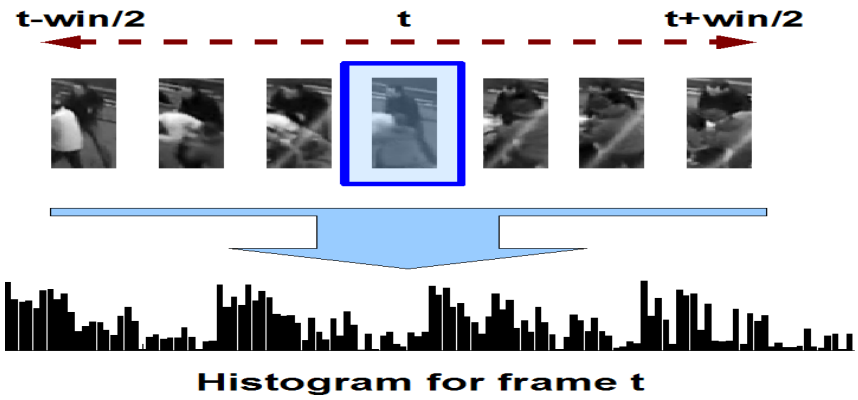


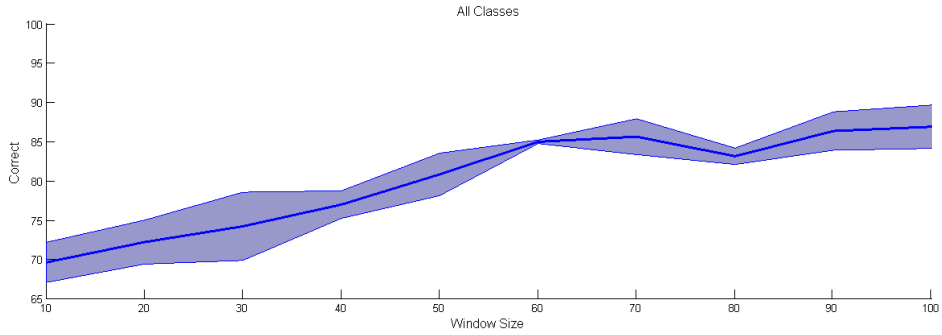
Figure 5.8: Construction of the histograms for continuously labeling all frames in the video. The current frames (highlighted) histogram is made up of cuboid centres from within a specified window (in this case 50 frames either side).

In order to continuously classify each frame a window around the current frame was used to provide the features which the classifier used. This approach is shown in figure 5.8. The reason a window around the current frame to classify is used is to help with lag when the activity changes. Whole sequences were again divided into training and testing with the results of classifying only the test set are presented. By dividing up complete sequences rather than only frames we ensure we are classifying data rather than interpolating it.

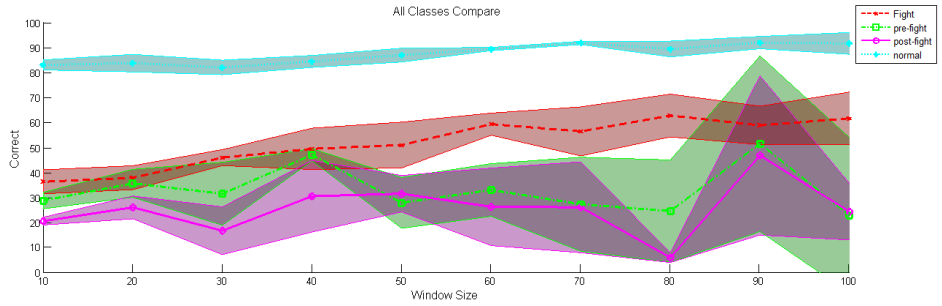
Every other step of the algorithm stayed the same, except that the features are derived from a finite window around the current frame. This gave a vast increase of the number of samples. For the BEHAVE dataset there are 31094 samples of size ± 50 frames to classify (vs 1138 complete sequences, as in the previous section). The CAVIAR dataset gives 3094 individual frames to classify (vs 56 complete sequences). First we investigate what window size it is appropriate to use.

5.7.2.1 Optimal Window Size

To determine how much video information to use to best classify a frame the window size was varied between 10 and 100 frames. It was found that 10 was the minimum amount of time needed (just under half a second) in order for there to be sufficient cuboid descriptors generated. At the other end, many sequences displaying a behaviour of interest (such as pre-fighting) are not much longer in duration (as taken from the previously segmented data). It is possible to generate window sizes of greater than 100 but several behaviours will be contained within that sequence and would not provide



(a)



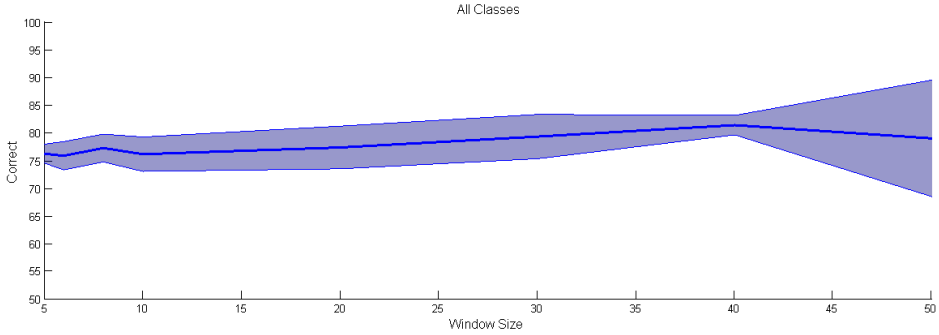
(b)

Figure 5.9: Results of varying window size when continuously classifying the BEHAVE dataset. Overall results are shown in (a) whilst per class results are given in (b). For figure (b) blue dots denote normal behaviour, red dashes and crosses denotes fighting. Green squares with dashes and dots denotes pre-fighting whilst purple circles denote post fighting behaviours. For both graphs the shaded areas represent standard deviation.

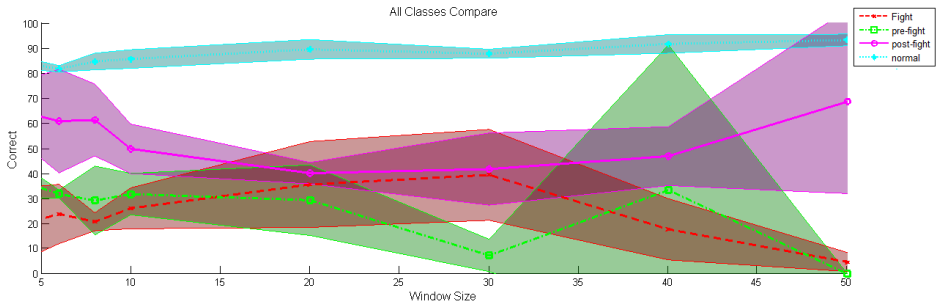
a fair test of the algorithm's ability to classify a short video clip. The results for the BEHAVE sequences are shown in figure 5.9

This experiment was also carried out upon the CAVIAR dataset. The length of time that some of the classes such as pre-fighting takes within the caviar sequences meant that it was only possible to run experiments up to a window size of 50. The results can be seen in figure 5.10.

Both sets of results were run over 50 runs using a different subset of training and testing data in order to establish the variance of each window size. The standard deviation is is given by the shaded regions. It is possible to see that around a window size of 60 the improvement in using larger window sizes is reduced. However both pre-fight and post fight improve when a window size of 90 is used as does fighting although not to the same degree. As these are the classes we are most interested in then the window



(a)



(b)

Figure 5.10: Results of varying window size when continuously classifying the CAVIAR dataset. Overall results are shown in (a) whilst per class results are given in (b). For figure (b) blue dots denote normal behaviour, red dashes and crosses denotes fighting. Green squares with dashes and dots denotes pre-fighting whilst purple circles denote post fighting behaviours. For both graphs the shaded areas represent standard deviation.

		True			
		Fight	Pre-Fight	Post Fight	Normal
Classified	Fight	0.67	0.38	0.32	0.05
	Pre-Fight	0.17	0.2	0	0.01
	Post-Fight	0.01	0	0.68	0.01
	Normal	0.15	0.42	0	0.93
	Total Samples	1382	254	25	12432

(a)

		True	
		Fighting Related	Normal
Classified	Fighting Related	0.81	0.07
	Normal	0.19	0.93
	Total Samples	1661	12432

(b)

Figure 5.11: Confusion matrices for the BEHAVE dataset continuous sequences at a window size of 90. (a) shows per class performance whilst (b) shows the results of aggregating fighting behaviour together.

size should be set to 90 in order to classify them as best as possible.

For the CAVIAR sequences there is not such a pronounced effect especially overall. There is a steep rise in classification performance for the fighting class between 40 and 50 frames so this was determined to be the best size overall. However it should be noted that for this dataset there are very few fighting situations compared to normal situations. Results are presented when using a window size of 45 for the CAVIAR dataset.

5.7.2.2 Classification Results

Behave Dataset

The best result when using this method on the BEHAVE sequence gave an overall classification performance of 89%. Again this rose to 92% when only fighting vs normal behaviour was considered. Confusion matrices for classification of continuous video data on the BEHAVE dataset is given in figure 5.11.

An example of classification is given below in figure 5.12.

When classification is performed in this manner parts of the sequences are misclassified as being normal when they are not. The lower number of examples when using a 100 window size for post-fight sequences is due to the short timescale upon which they happen (ie there are not as many post fight situations of 100 frames in length). A future improvement will be to construct the histograms to adapt their length based upon the video information available. The switching between normal and fighting frames is due to the similarity in their appearance over a relatively short timescale.

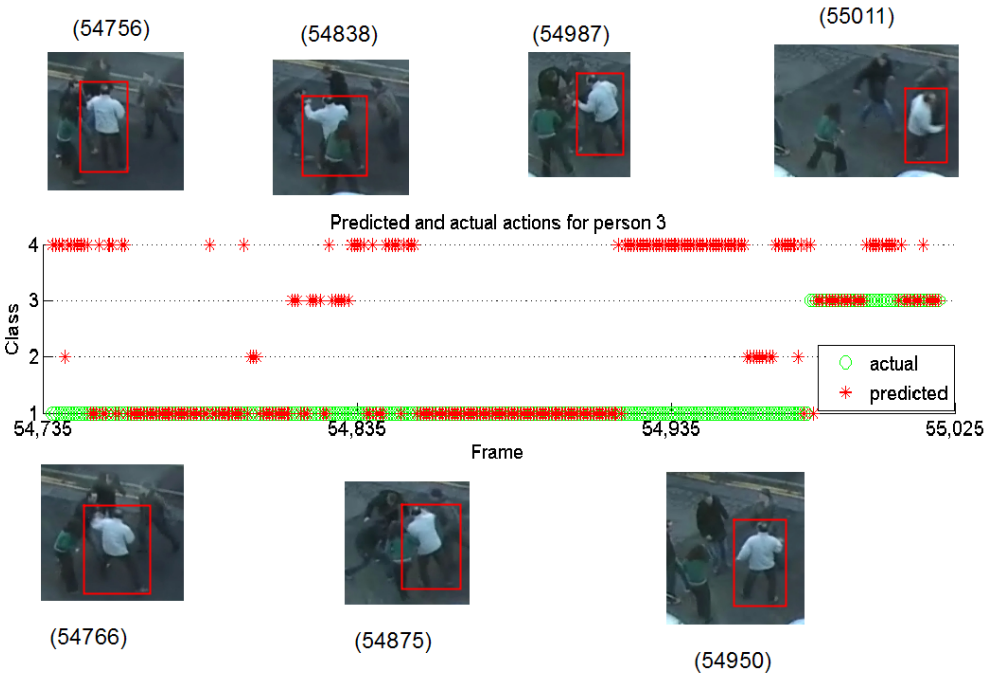


Figure 5.12: Predicted actions for the individual shown in the red box. The numbers in parenthesis refer to the frame numbers. Here Class 1 is fighting, 2 pre-fighting, 3 post fighting and 4 is for a normal situation. Around frame 54,935 the individual slowly breaks away from the fighting. This may explain the errors around this time, it looks very similar to a group splitting up. There is a slight prediction delay between fighting and post-fight behaviour of running away. This is down to using a window around the current frame, thus basing the classification on some portion of the past, coupled with the uncertainty as event change.

CAVIAR dataset

Results for the CAVIAR dataset are given in figure 5.13. For this dataset the results are not as good. Fighting and pre-fighting are frequently confused with normal behaviour.

		True			
		Fight	Pre-Fight	Post Fight	Normal
Classified	Fight	0.07	0	0	0.092
	Pre-Fight	0	0	0.03	0.004
	Post-Fight	0	0.11	0.32	0.004
	Normal	0.92	0.88	0.64	0.9
	Total Samples	27	62	103	1355

(a)

		True	
		Fighting Related	Normal
Classified	Fighting Related	0.24	0.1
	Normal	0.76	0.9
	Total Samples	192	1355

(b)

Figure 5.13: Confusion matrices for continuous sequences. (a) shows per class performance whilst (b) shows the results of aggregating fighting behaviour together. CAVIAR dataset. Results are for a window size of 45.

This may have to do with the very small number of fighting examples contained within this dataset coupled with the very short time span. When watching pre and post fighting behaviour some of the examples have less purposeful movement and speed than those contained in the BEHAVE sequences and real fights.

5.8 Generalisation of Structure

For problems in a more general context with more classes it may not be possible or desirable to exhaustively test every possible tree. There may simply be too many classes or one may just wish to further subdivide a class without re-calculating the entire classification structure. An example of this could be to subdivide pre-fighting into those where persons are already running at one another against those cases where persons are already in a group. Such a method would make the surveillance system more adaptable.

In this section we introduce a method to iteratively build a classification tree without exhaustive search, as was the case in the previous sections. The same classification

Algorithm 9 The overall algorithm for dynamically constructing classification trees.

DynamicBoost($\mathcal{C}$)

// add a single node with all possible classes

$\mathcal{T}$ =**addNodeToTree**($\mathcal{C}, \mathcal{T}$)

// then work out the best partition for these classes

While_More_Nodes($\mathcal{T}$)

$\mathcal{N}$ = **GetNextNode**($\mathcal{T}$)

[$\mathcal{P}, \mathcal{C}$] = **BestPartition**($\mathcal{N}$)

$\mathcal{T}$= **AddNode**($\mathcal{P}, \mathcal{C}, \mathcal{T}$)

Algorithm 10 Algorithm to find the best partition for a particular node of the tree

BestPartition($\mathcal{N}$)

$\mathcal{C} = \mathcal{N}_{\mathcal{C}}$

$\mathcal{P}$ = **all_partitions**($\mathcal{C}$)

for each $(l, r) \in \mathcal{P}$

res_i= **AdaBoost**(l_i, r_i)

$i = \max_i(res_i)$

return $[\mathcal{P}_i, \mathcal{C}_i]$

method is used upon the resulting final tree as given in algorithm 8. The general approach that is used is one of evaluating all possible class partitions, then selecting the best performing partition. The best performing partition is then used by nodes further down the hierarchy to split the data again (into left and right partitions) and then once again evaluating such partitions. This procedure of selecting the best partition and then creating new nodes with a reduced set is continued until each tree node is empty and thus there are no more classes to partition.

The algorithms to dynamically build such a tree are given below. The overall algorithm is given in algorithm 9 whilst the main modification to determine the best partition at a particular node is given in algorithm 10.

The addition of a new tree node is shown in algorithm 11.

5.8.1 Results Using a Dynamically Created Tree

Here the results when using a dynamical classification tree structure are presented. Again we compare them to the previous best results which were generated using a dictionary of size 90 for the BEHAVE dataset. The results are presented in figure 5.14.

Algorithm 11 Algorithm to add a node to the tree

$\mathcal{T}$ = **AddNode**($\mathcal{P}$,$\mathcal{C}$, $\mathcal{T}$)

if $\|\mathcal{C}\| > 1$

//create nodes for both the left and right partitions and add them to the tree

$\mathcal{T}$ = **addNodeToTree**(l,$\mathcal{T}$)

$\mathcal{T}$ = **addNodeToTree**(r,$\mathcal{T}$)

else

//add a terminating node to the tree

$\mathcal{T}$ = **addNodeToTree**($\{\}$, $\mathcal{T}$)

		True			
		Fight	Pre-Fight	Post Fight	Normal
Classified	Fight	0.78	0	0.05	0.01
	Pre-Fight	0.03	0.86	0.05	0.03
	Post-Fight	0.05	0.06	0.88	0.02
	Normal	0.14	0.08	0.02	0.94
	Total Samples	36	92	55	386

(a)

		True	
		Fighting Related	Normal
Classified	Fighting Related	0.93	0.06
	Normal	0.07	0.94
	Total Samples	183	386

(b)

Figure 5.14: Confusion matrix for the best performing dynamically created tree on the BEHAVE dataset.

		True			
		Fight	Pre-Fight	Post Fight	Normal
Classified	Fight	0.67	0	0	0.07
	Pre-Fight	0	0.75	0.2	0
	Post-Fight	0	0	0.8	0
	Normal	0.33	0.25	0	0.93
	Total Samples	3	4	5	15

(a)

		True	
		Fighting Related	Normal
Classified	Fighting Related	0.83	0.07
	Normal	0.17	0.93
	Total Samples	12	15

(b)

Figure 5.15: Confusion matrix for the best performing dynamically created tree on the CAVIAR dataset.

The previous averaged result when exhaustively searching over all of the possible trees was 89.9% with a standard deviation of 0.02. The overall accuracy rises to 95% when considering all fighting vs no fighting situations. Using a dynamically structured tree we attain 88.5% (a drop of 1.4%) overall classification accuracy when averaged over 50 runs along with a std deviation of 0.2 (a rise). Performance when considering only fighting vs non fighting gives 93% accuracy (compared with 95% previously) when considering all fighting vs non fighting events. Experiments were also conducted upon the CAVIAR dataset. These results are shown in figure 5.15.

When considering all trees on the CAVIAR dataset the overall classification accuracy is 89.3% with again a very small standard deviation of 0.1. When dynamically constructing the trees the average drops to 77% (drop of 12%) with a rise in the standard deviation to 0.8 (against 0.1). When considering fighting vs non fighting behaviour the accuracy increased to to 92.9% using an exhaustive search where as dynamically constructing the tree gave an average of 89%.

5.8.1.1 Summary

This section has demonstrated a way to successfully classify behaviour types without exhaustive search. It is perhaps not surprising that an exhaustive search yields better results. However there may be many situations where this is not possible.

For the larger BEHAVE dataset there is a very small difference between exhaustive search and dynamical construction of the classifier. The effect is greater on the smaller CAVIAR dataset where a drop in averaged performance of 12% is observed. Due to the small amount of fighting examples the method seeks to best classify the normal behaviour first. This effect is less pronounced when classifying all fighting behaviour together.

5.9 Comparison With Other Work

Based on the work of Trosciankio [Troscianko et al., 2004] the performance of our classifier system compares favourably with that of human performance (reported human performance is 80% of criminal incidents correctly with 65% of similar but normal incidents classified correctly). However it should be noted that the datasets are different (it was not possible to obtain Trosciankio's dataset). One of the main advantages of our method is that it has been demonstrated to work for small datasets such as the one we are using. This is an advantage when training data is limited, especially with regard to abnormal situations which by their very nature happen infrequently. The ability to learn the models from the visual evidence rather than requiring the use of a pre-determined [Cupillard et al., 2002] script is useful in chaotic situations where it is unclear what the correct script is.

When a smaller dataset is used the method struggles. The fights in the CAVIAR dataset look very different to one another and are also very short, especially when you partition them into specific classes such as pre and post fighting. The few examples may give rise as to why the method performs so poorly when continuously labeled data is used.

5.10 Conclusion and Future Work

The major contribution this chapter has addressed is that of investigating the feasibility of identifying pre-fight situations. The ability to identify when a fight is likely to break

out is useful in surveillance applications as it may be possible to intervene to stop a crime occurring or at least identify such situations at the earliest possible opportunity to allow useful intervention. The role of identifying post fighting behaviour is also of use as there may be some areas which CCTV cameras do not cover. They may only witness the end of a fight but it may be important to send assistance to this area in an effort to help victims and stop further criminal acts occurring.

The second major contribution is in publishing results on publicly available datasets. Such transparency is important in order to establish how well algorithms work in comparison to others.

This chapter has presented a way to classify fighting situations. Our method gives 96% correct classification on the BEHAVE dataset compared to Datta *et al.* [Datta et al., 2002] who reported 97% and Cupillard *et al.* [Cupillard et al., 2002] who report 95% for detection of fighting situations on other (and separate) datasets. However our method does not require the pre segmentation of parts of individuals, foreground extraction or pre compiled behaviour models. It has also been demonstrated that it is possible to identify pre and post-fight situations. Such cases are important to monitoring situations as intervention before the act is always preferable.

A hierarchical classifier is useful in many surveillance applications. Using such a structure can visually show you how the classification algorithm perceives the features which are given to it. This can be useful as a sanity check to make sure that the method is grouping things as you expect them to be.

However it is felt the most useful aspect of using a hierarchical classifier is in the ability to subdivide behaviours into a finer degree of granularity. For example in a surveillance application one may wish to identify all the fighting situations (as we have done here) and then obtain further granularity so as to identify pre and post fight situations as we have shown. This ability is useful as it can allow a fine tuning of a surveillance system. The dynamical method given in section 5.8 would be of use in such cases.

Future work should seek to improve the classification of continuous sequences perhaps by incorporating temporal models (eg, hidden Markov models) to improve classification. A further extension would be to remove the manual tracking component altogether (although some targets will be temporarily lost), or to combine individuals into group actions.

Acknowledgments

Pitor Dollar should be thanked for kindly making his cuboid code available.

Chapter 6

Conclusion

Within this thesis we have investigated the classification of multiple interacting persons. This has been approached in several ways throughout. First we looked at how team game interaction can be classified. This was demonstrated on publicly available data of sports games. Results were then presented demonstrating how a simpler linear classifier (a support vector machine) modelling the team as a whole could outperform a complex hierarchical linear dynamical system, which modelled individual players.

In chapter 3 we moved away from structured interactions and concentrated upon a more general surveillance application. We improved upon the previous best results as published by Oliver *et al.* [Oliver et al., 2000a] by introducing a new feature set and using an improved classifier. We demonstrated improved performance against the previous best method and demonstrated the effect of frame length in classification accuracy. Again results were presented over several datasets with a combined total of over 150,000 samples.

Finally we look at the ability to detect fighting events. In particular we look at the ability to detect pre and post fight events and have shown that this is possible using a hierarchical model. Performance upon two substantial datasets was reported. The hierarchical classifier was then extended to include a dynamically created tree. Such an approach would be useful in large scale problems or where a finer degree of detail is required without re-computing the whole structure.

6.0.1 Main Contributions

The main contribution's of this thesis are now expanded upon.

- Comparison and demonstration of a simpler and more accurate model for learn-

ing team activities. The presented model requires no pre-defined template and is quick to compute. It is also more accurate than the previously suggested model (Chapter 3).

  - In chapter 3 we demonstrated how using a feature vector calculated from team information can perform significantly better than individually modelling the players. When using a feature vector comprising of team information compared to individually modelling the players the data can be shown to be better separated when using PCA. The main benefit of taking this approach is the reduction in complexity in the model which is required. The simpler SVM model out performed a linear dynamical system. One of the reasons for this was that in cases where there are relatively few examples a simpler model does better as it has less parameters to learn. This is a common theme in video applications where the representation needs to be learned despite few training examples.

- Significant improvement in classification performance between interacting persons (Chapter 4).

  - Within chapter 4 we improve upon the previous best method of Oliver et al.'s [Oliver et al., 2000a] proposed method. A new richer feature set is also introduced. We demonstrate its superior performance of our proposed method over several datasets. We also make a comparison between other probabilistic and non probabilistic models. We demonstrate how a conditional random field (CRF) model can be used in surveillance applications. Results for a hidden Markov model (HMM) and linear discriminant (LDA) are also given.

- Demonstration and quantification of the performance effects relating to the length of time a sequence is viewed before making a decision (Chapter 4 and 5).

  - In chapter 4 the amount of frames it is necessary to observe before making a decision is investigated. We find that for several classes of interaction there is a optimal window size to use. For other classes this effect is not exhibited. CRF classification outperforms all methods when there are only a limited amount of frames to use for classification (small window size). It does not always improve significantly with larger window sizes suggesting

a longer range temporal component should be incorporated into its formulation. Both of the dynamic probabilistic models (HMM and coupled HMM) do generally improve when more information is afforded to them. There is usually an initial dramatic improvement past which there is very little improvement.

  - In chapter 4 we used a per class window size to improve classification performance. Such a classifier improved performance between 2% to 10%.
  - In chapter 5 we also observe this effect where the window size plays a role in classification accuracy. It is shown that increasing the window size improves performance up to a point. We also note that there is some variation between classes as to which is the optimal window size to use.

- Investigation and demonstration of pre-fight detection (Chapter 5).

  - Chapter 5 demonstrates the feasibility of identifying pre-fighting behaviour. This is demonstrated upon two datasets. We also show that the feasibility of detecting behaviour associated with fighting. In this case the ability to classify pre, post and actual fighting and to differentiate such situations from normal behaviour has been demonstrated. To our knowledge this is the first investigation of the ability of a computer to perform this task. Human performance has to perform such a task has been tested in [Troscianko et al., 2004].

- Demonstration of a scalable hierarchical classifier for detecting fighting behaviour (Chapter 5).

  - A hierarchical classifier with the ability to determine its structure from validation data was also presented in chapter 5. Such an ability would be useful in cases where additional subdivision of classification was required (eg. fights with running and fights with groups). It would also prove useful where there are a large number of classes and it would not be feasible to obtain the best tree by exhaustive search.

- The creation of the BEHAVE dataset [Blunsden et al., 2007b].

  - This is a publicly available and annotated dataset with a large number of frames showing different interacting behaviours. Most of the frames

demonstrating interacting behaviours are also marked up with ground truth consisting of a bounding box. The creation of such a dataset is important as it allows others to make comparisons to the results presented here. It will hopefully also be useful to other researchers in their work.

- Publish results on several publicly available datasets (all chapters).
  - The publication of results on publicly available datasets allows meaningful comparisons between methods to be easily made. Having a benchmark which is repeatable by others is essential in evaluating progress in the field.

6.0.2 Future Directions

Based on the work presented in this thesis some future directions are now suggested.

- Adaption of team based models to include unusual events like a player missing.
  - One of the major weaknesses in the team classification chapter was its inability to deal with unusual events such a player missing. A simple way would be to train a set of models without each individual player present. Should a player be removed one of these other models would be used. However this is a computationally heavy approach and not wholly satisfactory. A better way should be determined.
- Temporal Modelling improvements in the CRF
  - The CRF model as used in chapter 4 performed extremely well especially when classifying using limited frame information. However there was a shortcoming in that longer window sizes did not always improve classification and in some instances reduced it. Other dynamical models such as the HMM and CHMM always improved to some extent as window size was increased. However this improvement became less after an initial period of rapid improvement. Future work should seek to address the shortcomings of the CRF by incorporating a stronger medium to long term temporal model.
  - Another possibility for investigation should be the ability to make use of a classifier which can use a variable length window size depending upon

the class. For some classes the window size makes a large difference in performance. There is also no universally best window size (as its class dependent) so this should be exploited to give an improvement in performance.

- Incorporation of group building framework (such as that of Hakeen and Shah [Hakeem and Shah, 2007]).
  - Hakeen and Shah [Hakeem and Shah, 2007] presented a way to build up group interactions given a interaction ontology. They provided details of how to build up a model of being in a group given that you can detect low level behaviour similar to what we present here. They do not provide details of how such features are obtained. Their method could be used in conjunction with the classification methods presented here to build up a richer and more detailed picture of what is happening within a video sequence.
- Temporal smoothing of individual frame classification.
  - Throughout all chapters when classifying continuous sequences it is visible that there is some jumping of classification and also a lag. Whilst it is relatively easy to correct for lag (use a few frames ahead as well, giving a few seconds delay in classification) a temporal model would help to stop the rapid switching of classification labels. A Viterbi decoding or similar approach could be used to improve upon the performance shown here.
- Improvement in selection and combination of a per class window size classifier
  - In chapter 4 the effect of using a different window size for each class was investigated. In this instance the classifier gave slightly improved performance. There may be other ways to create a per class window size classifier which give a greater performance increase.

6.0.3 Final Word

This thesis has introduced the problem of multiple person interaction and presented some improved feelings for classification of multiple person interaction. This is still

very much an open problem. There are more possible avenues for exploration than those listed under future work in section 6.0.2.

One major stumbling block seems to be the lack of willingness to share essential tools such as well annotated datasests and tracking tools. Once the community can successfully negotiate these barriers then progress on surveillance and associated vision research will dramatically improve. In this respect probably the most lasting contribution of this thesis and associated work has been to make available a large annotated dataset. It is hoped others do the same.

Appendix A

Datasets

A.1 CAVIAR Dataset

The CAVIAR dataset [project/IST 2001 37540, 2004] contains 27 separate sequences ranging between a few seconds and a few minutes. The resolution of the images is half-resolution PAL standard (384 x 288 pixels, 25 frames per second) and compressed using MPEG2. In addition to the video data the positions of people have been hand labelled giving the bounding box of a person. This dataset was obtained inside the lobby of INRIA in France. A homography was available for this dataset and was used throughout the experiments.

From this data all two and multi person interactions were annotated. This dataset gives 81 complete sequences where an interaction is taking place and over 10,600 frames which contain examples of an interaction.

The CAVIAR dataset contains 11,415 frames which have labelled examples of interactions. Within this set there are 5 distinct classes which we seek to identify and classify. The 5 classes consists of examples of people: walking together (2,543), approaching one another (942), ignoring (4,916), meeting one another (1,801), splitting up (879) and fighting (334). The numbers in brackets indicates how many frames contain this behaviour. An example of a sequence is given in figure A.1.

A.2 BEHAVE Dataset

The intended purpose of the BEHAVE dataset [Blunsden et al., 2007b] was to generate many more interactions as compared to other data available. The video is of an outdoor scene obtained at a resolution of 640x480 at 25 frames per second. The BEHAVE

Figure A.1: Two frames from a fight sequence taken from the CAVIAR dataset. The people highlighted in the green and the blue are fighting.

(a) (b)

Figure A.2: Two frames from the BEHAVE sequence. The sequence on the left (a) shows one group approaching another whilst the frame on the right (b) demonstrates fighting behaviour.

dataset contains 134,457 frames which have labelled examples of interactions. Within this set there are 5 distinct classes which we seek to identify and classify. The 8 classes consist of examples of people: In a group (91,421), Approaching (8,991), walking together (14,261), splitting (11,046), ignoring (1,557), fighting (4,772), running (1,870) and chasing (539) one another. The numbers in brackets indicates how many frames contain this behaviour. Again situations were acted out with ground truth generated by manual marking of a bounding box for every person on screen. Example frames from this dataset are given in figure A.2.

A.3 Handball Data

The data set used throughout this paper is from the publicly available CVBASE dataset [CVb, 2006]. Only the handball sequences are used in the methods and results presented here. The handball dataset consists of 3 separate video cameras recording a 10

minute long game. Court coordinates of each of the players for one team throughout this sequence are available (first 1000 frames are shown in figure A.3). The activity the whole team is engaged in and the starting and end times of the activity is also annotated. The timeout class from the original dataset was removed as this is not regarded as a coordinated activity. In addition some classes have been merged together. This step was primarily taken due to the high similarity between the activities themselves. The classes 'offence against set-up defence, setting up an offence'(nfpn) and 'offence against setup defence, ending an offence' (nfan) were merged. These classes are highly similar in description and in appearance and require extra information about the game (ie the ability to detect an attempt on goal) to distinguish them. The classes 'defence, basic defence, against preparation of an offence'(obz) and 'defence, basic defence against ending offence'(obg) were merged for similar reasons. The classes 'defence returning' (ovpp) and 'defence slowly returning' (ovpc) were merged as the actual speed difference distinguishing each class was not well defined. These activities may have looked different if information on the other team was also available, particularly in distinguishing between trying to stop a fast break. This gives 5 final classes shown in figure A.4.

A.4 Synthetic Data

The synthetic dataset it used to test out the proposed approach with known data. The synthetic dataset consists of 12,385 frames over five classes. These classes are:

walk together (2950) meet (1636), wait (2101), split (3243) follow (2455) with the number of example frames given in brackets. These examples are divided into sequences which range in length between 100 and 500 frames.

The data was generated by a program. Noise was added to the initial trajectories in an effort to make the set more challenging.

A.5 Football Data

The football data was gathered from the synthetically generated TricTrac dataset [multitel, 2006]. The dataset provides a multi camera viewpoint for showing 3 separate scenarios within a football match. The first shows a team attacking with a goal being scored. The second shows one team attacking and shooting the ball at the goal without

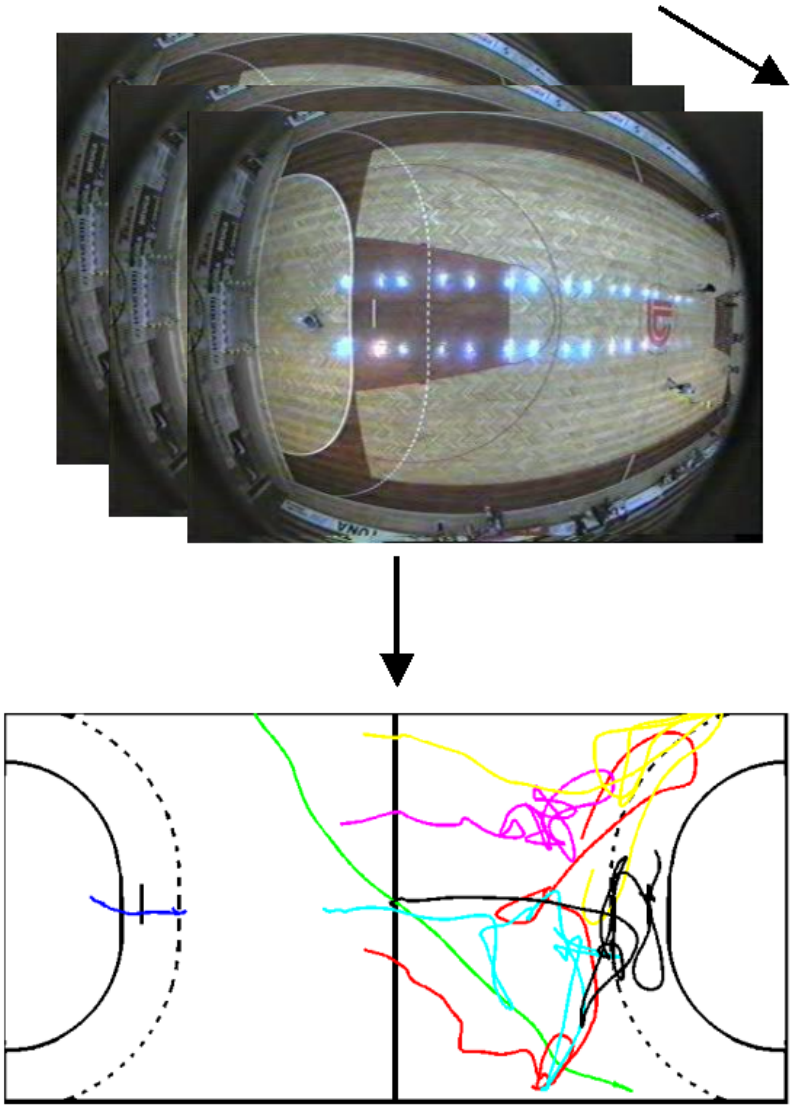


Figure A.3: The first 1000 frames from the video sequence and the associated player positions as given in court coordinates.

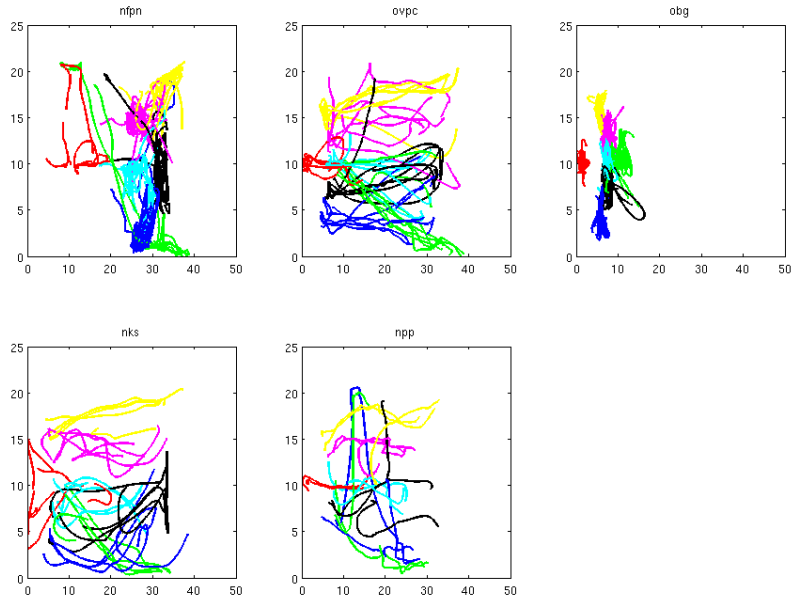
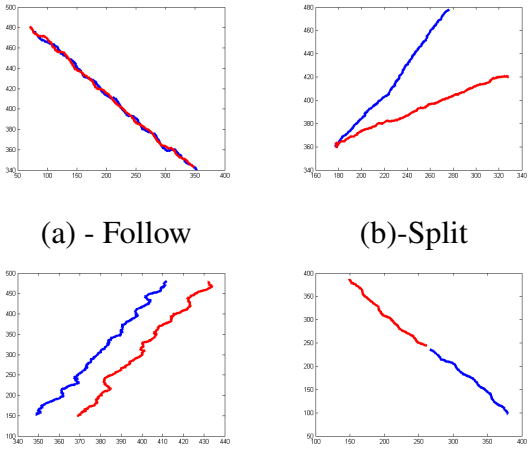


Figure A.4: Positional information of each player (different colour) plotted in court co-ordinates. The classes are: nfpn - offence against set-up defence , ovpc - defence, returning, obg - defence, basic defence, nks - offence, fast break, npp - offence, slowly going into offence.



(a) - Follow (b)-Split

(c) - Walk Together (d) Meet

Figure A.5: Example trajectories of four classes from the synthetic interaction dataset.

Figure A.6: Original video data from the TriacTrac dataset. This sequence is from scenario 1 where a goal is scored.

a goal being scored and the final case shows the team successfully defending an attack with no shot or goal resulting.

The data comprises of a total of 10,162 individual frames. For each frame the world co-ordinates of each individual player are given. The length of each sequence ranged from 682 frames to 971 frames.

Appendix B

Algorithms

B.1 Variational Estimate of theDynamical Systems Tree

The parameters for the aggregating processes (indexed by s) and the leaf processes (indexed by i) for the initial timestep are given as :

$$
\begin{aligned}
p(s_0^a = j) &= \phi^a(j) \\
p(s_0^i = j) &= \psi^i(j) \\
p(x_o^i | s_o^i = j) &= \mathcal{N}(x_0^i | \mu_j^i, q_j^i) \\
p(y_0^i | x_0^i) &= \mathcal{N}\left(y_0^i | C x_0^i, R\right)
\end{aligned}
$$

The function $\mathcal{N}(x|\mu,\sigma)$ represents the Gaussian function for variable x with mean μ and variance σ.The scaling factor is denoted by C whilst R is a (static) covariance matrix. The mean value of the i^{th} leaf process in state j is given as μ_j^i with associated variance q_j^i . For all subsequent timesteps the parameters are:

$$
\begin{aligned}
p(s_t^a = j | s_{t-1}^a = k, s_t^{\pi(a)} = l) &= \Phi^a(j,k,l) \\
p\left(s_t^i = j | s_{t-1}^i = k, s_t^{\pi(i)} = l\right) &= \Psi^i(j,k,l) \\
p(x_t^t | x_{t-1}^i, s_t^i = j) &= \mathcal{N}(x_t^i | A_j^i x_{t-1}^i, Q_j^i) \\
p(y_t^i | x_t^i) &= \mathcal{N}(y_t^i | C x_t^i, R)
\end{aligned}
$$

Computing the likelihood directly within this model is not tractable. In order to perform inference and subsequently expectation maximisation the model is partitioned into a simpler and less connected subset. An approximate (ie not the true) variational

distribution $Q(\mathcal{S},\mathcal{X})$ is used to model the true posterior $\mathcal{P}(\mathcal{S},\mathcal{X}|\mathcal{Y})$. In order to compute the variational distribution the model is simplified with all Markov chains being unlinked from one another. This allows the forward backward algorithm [Rabiner, 1990] to be applied to compute the likelihoods. Using the techniques described in [Neal and Hinton, 1998] the updating of parameters of this variational distribution can be completed. For the DST model the following inequality on the incomplete log-likelihood is given by:

$$log\mathcal{P}(\mathcal{Y}|\Theta) \geq \sum_{s}\int_{x} Q(\mathcal{S},\mathcal{X}) \log \frac{\mathcal{P}(\mathcal{S},\mathcal{X},\mathcal{Y}|\Theta)}{Q(\mathcal{S},\mathcal{X})} d\mathcal{X} \tag{B.1}$$

This variational bound is at equality $\Theta = \Theta^{*}$when $Q(\mathcal{S},\mathcal{X}) = \mathcal{P}(\mathcal{S},\mathcal{X}|\mathcal{Y},\Theta)$. Given that Q is a simpler bound than $\mathcal{P}(\mathcal{S},\mathcal{X}|\mathcal{Y},\Theta)$ the bound will be lowered and can no longer make tangential contact. The parameters of Q are thus optimised to get Q as close as possible to the posterior as measured by the Kullback-Leibler divergence ($KL(Q(\mathcal{S},\mathcal{X})||\mathcal{P}(\mathcal{S},\mathcal{X}|\mathcal{Y})$).

The update rules for Q are derived using Hamiltonians (denoted by H in equations B.2 and B.3) of the probability distributions, as described by Ghahramani and Hinton [Ghahramani and Hinton, 1998], given by :

$$P(S,X,Y) = \frac{1}{z}\exp\left(-H(\mathcal{S},\mathcal{X},\mathcal{Y})\right) \tag{B.2}$$

$$Q(S,X) = \frac{1}{z_{Q}}\exp\left(-H_{Q}(\mathcal{S},\mathcal{X})\right) \tag{B.3}$$

z is a normalisation function. This was described by Ghahramani and Hinton [Ghahramani and Hinton, 1998]. The Q distribution has the following parameters for each aggregating and leaf process.

$$Q(s_o^a = j) = \hat{\phi}^a(j), \qquad Q(s_t^a = j|s_{t-1}^a = k) = \Phi_t^a(\hat{j},k)$$
$$Q(s_o^i = j) = \hat{\psi}^a(j), \qquad Q(s_t^i = j|s_{t-1}^i = k) = \hat{\psi_t^i}(j,k)$$
$$Q(x_0^i) = \mathcal{N}(x_0^i|\hat{\mu}^i,\hat{q}^i), \qquad Q(x_t^i|x_{t-1}^i) = \mathcal{N}(x_t^i|\hat{A_t^i}x_{t-1}^i,\hat{Q}_t^i)$$

As suggested in [Ghahramani and Hinton, 1998] to find the Q which minimises the KL divergence the derivatives of the difference of Hamiltonians ($D = H_{Q} - H$) are set to zero as shown in equation B.4:

$$\frac{\partial\langle D\rangle}{\partial\langle s_t^a\rangle} = \frac{\partial\langle D\rangle}{\partial\langle s_t^a s_{t-1}^a\rangle} = \frac{\partial\langle D\rangle}{\partial\langle s_t^i\rangle} = \frac{\partial\langle D\rangle}{\partial\langle s_t^i s_{t-1}^i\rangle} = \frac{\partial\langle D\rangle}{\partial\langle x_t^i\rangle} = \frac{\partial\langle D\rangle}{\partial\langle x_t^i x_{t-1}^i\rangle} = 0 \tag{B.4}$$

Angled brackets ($\langle\rangle$) refer to an averaged value. Solving these updates the variational parameters for each aggregator process (indexed by a, in this case there is only one) and each leaf process (indexed by i) as follows:

$$
\begin{aligned}
\hat{\Phi}^a_t(j,k) &\propto \exp\left(\sum_l \left\langle s_t^{\pi(a)}(l)\right\rangle \log\Phi^a(j,k,l) + \sum_{c\in Child(a)}\sum_{h,i}\left\langle s_t^c(h)s_{t-1}^c(i)\right\rangle \log\Phi^c(h,i,j)\right)\\
\hat{\phi}^a(j) &\propto \exp\left(\sum_l \left\langle s_t^{\pi(a)}(l)\right\rangle \log\phi^a(j,l) + \sum_{c\in Child(a)}\sum_{h}\left\langle s_0^c(h)\right\rangle \log\Phi^c(h,j)\right)\\
\hat{\Psi}^i_t(j,k) &\propto \exp\left(\sum_l \left\langle s_t^{\pi(i)}(l)\right\rangle \log\psi^i(j,l) - \frac{1}{2}\log\left|Q^i_j\right| - \frac{1}{2}\left\langle (x^i_t - A^i_j x^i_{t-1})^T (Q^i_j)^{-1}(x^i_t - A^i_j x^i_{t-1})\right\rangle\right)\\
\hat{\psi}^i(j) &\propto \exp\left(\sum_l \left\langle s_t^{\pi(i)}(l)\right\rangle\right) \log -\frac{1}{2}\left|q^i_j\right| - \frac{1}{2}\left\langle \left(x^i_0-\mu^i_j\right)^T (q^i_j)^{-1}\left(x^i_0-\mu^i_j\right)\right\rangle\\
\hat{A^i}_j &= \hat{Q}^i_t \sum_j \left\langle s^i_t(j)\right\rangle (Q^i_j)^{-1} A^i_j\\
(\hat{Q}^i_t)^{-1} &= \sum_j \left\langle s^i_t(j)\right\rangle (Q^i_j)^{-1} + \sum_j \left\langle s^i_{t+1}(j)\right\rangle (A^i_j)^T (Q^i_j)^{-1} A^i_j - (\hat{A^i_{t+1}})^T (\hat{Q^i_{t+1}})^{-1} \hat{A^i}_{t+1}\\
\hat{\mu^i} &= \hat{q}^i \sum_j \left\langle s^i_0(j)\right\rangle (q^i_j)^{-1}\mu^i_j\\
\hat{q^i} &= \sum_j \left\langle s^i_0(j)\right\rangle (q^i_j)^{-1} + \sum_j \left\langle s^i_1(j)\right\rangle (A^i_j)^T (Q^i_j)^{-1} A^i_j - \left(\hat{A^i_1}\right)^T \left(\hat{Q^i_1}\right)^{-1} \hat{A^i_1}
\end{aligned}
\tag{B.5}
$$

These are conditional distributions and as such should be normalised. After iterating the variational parameter updates (as given above) the forward-backward algorithm is used to give an inference on Q to get the normalised probabilities and associated marginals which are used to compute expectations over the hidden variables. The marginal probabilities which are required for calculating the values of the hidden variables are given below :

$$
\begin{aligned}
p(x_t)\\
p(x_t, x_{t-1})\\
p(s_t)\\
p(s_t, s_{t-1})
\end{aligned}
$$

The above approximate expectation step (as given in equation B.5) is then used to perform a maximum likelihood update of parameters Θ using the current Q distribution. As previously mentioned the true log likelihood of the model is intractable,

here the bound is evaluated. During learning the bound increases monotonically as the variational parameters in Q and Θ are iterated and updated. The bound itself is computed via the expected Hamiltonian's $\mathcal{H}$ under Q summed with the entropy of the Q distribution:

$$\mathcal{B}(Q,\Theta) = E_{Q(\mathcal{S},\mathcal{X})}\left\{H(\mathcal{S},\mathcal{X},\mathcal{Y}) - H_Q(\mathcal{S},\mathcal{X})\right\} \tag{B.6}$$

Such expectations only involve calculations over at most pairwise cliques of Q. The term $\mathcal{B}(Q,\Theta)$ as given in equation (B.6) is recursively computed for each aggregator and leaf process throughout the tree. Due to the decoupling assumptions the variational parameter updates (equation B.5) are independent given the inferred expectations under the Q distribution.

A DST model is created for each class. The examples for that class are then presented to the model and the parameters are updated as described above. In order to classify a new sequence the data is shown to each model and the likelihood calculated. The model with the highest likelihood is then taken to be the correct class and the sequence is labeled as such.

Bibliography

Machine vision group, university of ljubljana, cvbase '06 workshop on computer vision based analysis in sport environments, found at url: http://vision.fe.uni-lj.si/cvbase06/, 2006. URL `http://vision.fe.uni-lj.si/cvbase06/`.

J.H. Ahn, K.C. Kim, and H.R Byun. Robust object segmentation using graph cut with object and background seed estimation. In *ICPR06*, pages II: 361–364, 2006.

S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking. In *IEEE Transactions on Signal Processing, 50(2)*, pages 174–188, 2002.

C. Asavathiratham. *The Influence Model: A Tractable Representation for the Dynamics of Networked Markov Chains*. PhD thesis, Massachusetts Institute of Technology, 1996.

P. Baldi and Y. Chauvin. Smooth on-line learning algorithms for hidden markov models. *Neural Computation*, 6(2):307–318, 1994.

D. A. Baldwin and J. A. Baird. Discerning intentions in dynamic human action. *Trends in Cognitive Science*, 4(5):171–178, 2001.

H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *IJCAI*, 1977.

S. Basu and T. Choudhury. Modeling conversational dynamics as a mixed memory markov process. In *Advances of Neural Information Processing Systems (NIPS 2004)*. MIT Press, December 2004.

S. Basu, T. Choudhury, B. Clarkson, and A. Pentland. Towards measuring human interactions in conversational settings. In *CVPR '01 (Workshop: Cues in Communication)*, December 2001.

J. Batista. Tracking pedestrians under occlusion using multiple cameras. In *ICIAR (2)*, volume 3212 of *Lecture Notes in Computer Science*. Springer, 2004. ISBN 3-540-23240-0.

L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8, 1969.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathimatical Statistics*, 41(1):164–171, 1970.

A Baumberg and D C Hogg. An efficient method for contour tracking using active shape models. In *IEEE Workshop on Motion of Non-rigid and Articulated Objects*, 1994a.

A. M Baumberg. *Learning Deformable Models for Tracking Human Motion*. PhD thesis, The University of Leeds, School of Computer Studies, 1995.

A. M. Baumberg and D. C. Hogg. An efficient method for contour tracking using active shape models. In *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 194–199, Austin,Texas, July 1994b.

B. Bennett, D. Magee, A. G. Cohn, and D. C. Hogg. Using spatio-temporal continuity constraints to enhance visual tracking of moving objects. *ECAI 2004 Proceedings of the 16th European Conference on Artificial Intelligence*, 2:922–926, 2004.

A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *European Conference on Computer Vision*. 2004.

Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *ICCV*, pages 1395–1402, 2005.

S. Blunsden, R.B. Fisher, and E.L. Andrade. Recognition of coordinated multi agent activities, the individual vs the group. In *European Conference on Computer Vision (ECCV), CVBASE workshop*, Graz, Austria, 2006.

S. Blunsden, E. Andrade, and R. Fisher. Non parametric classification of human interaction. In *IbPRIA07*, volume 2, pages 347–354, 2007a.

S. Blunsden, E. Andrade, A. Laghaee, and R. Fisher. Behave interactions test case scenarios, epsrc project gr/s98146,

http://groups.inf.ed.ac.uk/vision/behavedata/interactions/index.html. On Line, September 2007b. URL http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/INTERACTIONS/

A. F. Bobick and R. Bolles. The representation space paradigm of concurrent evolving object descriptions. In *PAMI*, pages 146–156, February 1992.

D. Bobrow. An overview of krl. In *Cognitive Science*. McGraw Hill, New York, 1977.

Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *In Proc. IEEE Int. Conf. on Computer Vision*, 2001.

M. Brand. Shadow puppetry. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1237, 1999.

M. Brand. Coupled hidden markov models for complex action recognition, 1996a. submitted.

M. Brand. Coupled hidden markov models for modeling interacting processes, November 1996b. URL "citeseer.ist.psu.edu/article/brand97coupled.html".

M. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *IEEE Transactions Pattern Analysis Machine Intelligence*, 22(8):844–851, 2000. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.868685.

M. Brand, N. M. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, June 1997.

H. H. Bui, S. Venkatesh, and G. West. Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model. *Hidden Markov models: applications in computer vision*, pages 177–196, 2002.

A. Caporossi, D. Hall, P. Reignier, and J. L. Crowley. Robust visual tracking from dynamic control of processing. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, volume 1, pages 23–31, June 2004.

W. S. Ching. A novel change detection algorithm using adaptive threshold. In *Pattern Recognition Letters*, 1994.

T. Choudhury and A. Pentland. Characterizing social networks using the sociometer. In *In the Proceedings of : the North American Association for Computational Social and Organizational Science*, Pittsburg, PA, June 2004.

T. Choudhury, B. Clarkson, S. Basu, and A. Pentland. Learning communities: Connectivity and dynamics of interacting agents. In *In Proceedings of the International Joint Conference on Neural Networks Special Session on Autonomous Mental Development*, Portland, OR, July 2003.

R. T. Collins. Mean-shift blob tracking through scale space. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, volume 2, 2003.

R. Collobert, S. Bengio, and J. MariÃ©thoz. Torch: a modular machine learning software library. Technical report, IDIAP, 2002. Technical report IDIAP-RR 02-46.

D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Analysis Machine Intell.*, 25:564–575, 2003.

D. Comanicu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions Pattern Analasis and Machine Intelligence*, 24:603–619, 2002.

A.I. Comport, D. Kragik, E. Marchand, and Chaumette. F. Robust real-time visual tracking: Comparison, theoretical analysis and performance evaluation. In *In IEEE Int. Conf. on Robotics and Automation, ICRA'05*, April 2005.

T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam. Training models of shape from sets of examples. In *British Machine Vision Conference*, pages 9–18, July 1992a.

T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam. Active shape models - 'smart snakes'. In *British Machine Vision Conference (BMVC)*, pages 266–275. Springer-Verlag, 1992b.

T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam. The use of active shape models for locating structures in medical images. In *Image and Vision Computing*, volume 12, pages 355–366, July 1994.

J. L. Crowley and P. Reignier. Dynamic composition of process federations for context aware perception of human activity. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, volume 1, page xx, October 2003.

J. Cui, H. Zha, H. Zhao, and R. Shibasaki. Robust tracking of multiple people in crowds using laser range scanners. In *International Conference on Pattern Recognition (ICPR)*, page 1, Aug 2006.

F. Cupillard, F. Bremond, and M. Thonnat. Group behavior recognition with multiple cameras. In *Sixth IEEE Workshop on Applications of Computer Vision (WACV).*, 2002.

F. Cupillard, F. Bremond, and M. Thonnat. Behaviour recognition for individuals, groups of people and crowd, 2004.

A. Datta, M. Shah, and N. D. V. Lobo. Person-on-person violence detection in video data. In *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 1*, page 10433. IEEE Computer Society, 2002. ISBN 0-7695-1695-X.

J. W. Davis and A. F. Bobick. The representation and recognition of action using temporal templates. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 23, pages 257–267. IEEE Computer Society, 2001.

J. W. Davis and A. F. Bobick. The representation and recognition of action using temporal templates. MIT, 1997.

H. Dee and D. C. Hogg. Is it interesting? comparing human and machine judgements on the pets dataset. *Sixth International Workshop on Performance Evaluation of Tracking And Surveillance*, 33(1):49–55, 2004a.

H. M. Dee and D. C. Hogg. Detecting inexplicable behaviour. In *British Machine Vision Conference*, pages 477–486, September 2004b.

F. Dellaert. The expectation maximization algorithm, February 2002. Technical Report.

J. Deutscher, B. North, B. Bascle, and A. Blake. Tracking through singularities and discontinuities by random sampling. In *ICCV (2)*, pages 1144–1149, 1999.

T. G Dietterich. Machine learning for sequential data: A review. In T. Caelli (Ed.) Lecture Notes in Computer Science. Springer-Verlag, 2002.

P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *PETS*, pages 65–72, China, 2005.

A Doucet, N de Freitas, and N Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

Youtian Du, Feng Chen, Wenli Xu, and Weidong Zhang. Interacting activity recognition using hierarchical durational-state dynamic bayesian network. In *Advances in Multimedia Information Processing*, volume 4261 of *Lecture Notes in Computer Science*, pages 185–192. Springer Berlin / Heidelberg, 2006.

K. Duan and S. S. Keerthi. Which is the best multiclass svm method? an empirical study. In *Neural Information Processing Systems*, 2003.

R.O. Duda, P. E. Hart, and G. D. Stork. *Pattern Classification, Second Edition*. Wiley Interscience, University of Texas at Austin, Austin, USA, November 2000.

A. Efros, A. Berg, G. Mori, and J. Malik. Recognising action at a distance. In *In 9th International Conference on Computer Vision*, volume 2, pages 726–733, 2003.

A. Ekin, A.M. Tekalp, and R. Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, July 2003.

A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modelling using nonparametric kernel density estimation for visual surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Patterand Recognition*, volume 2, page 1, jun 2001.

G. Fan, V. B. Venkataraman, L. Tang, and J. P. Havlicek. A comparative study of boosted and adaptive particle filters for affine-invariant target detection and tracking. In *OTCBVS06*, 2006.

P. Felzenszwalb. Learning models for object recognition. In *CVPR*, volume 1, pages 56–62, 2001.

F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-camera people tracking with a probabilistic occupancy map. In *accepted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 3(61):268–278, 1973.

Y. Freund and R. E. Schapire. Game theory, on-line prediction and boosting. In *Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

N. Friedman and M. Goldszmidt. Learning bayesian networks with local structure. In *In Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence (UAI 96)*, 1996.

N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, pages 139–147. Elsevier Science Publishing Comapny, Inc., 1998.

Akira Fujimura and Kokichi Sugihara. Geometric analysis and quantitative evaluation of sport teamwork. *Syst. Comput. Japan*, 36(6):49–58, 2005.

A. Galata, N. Johnson, and D. Hogg. Learning variable length markov models of behaviour. *Int. Journal of Computer Vision and Image Understanding (CVIU)*, 81 (3):398–413, March 2001.

A. Galata, D. Cohn, A. G.and Magee, and D. Hogg. Modeling interaction using learnt qualitative spatio-temporal relations and variable length markov models. In *European Conference on Artificial Intelligence (ECAI)*, pages xx–xx, July 2002.

D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 73. IEEE Computer Society, 1996.

H. Gerber, R. Nagel and H. Schreiber. Deriving textual descriptions of road traffic queues from video sequences. In *The 15-th European Conference on Artificial Intelligence (ECAI'2002)*, pages 736–740, Lyon, France, July 2000.

Z. Ghahramani and G. E. Hinton. Variational learning for switching state space models. *Neural Computation*, 12(4):963–996, 1998.

D. Gibson. Finite state machines - making simple work of complex functions. Technical report, SPLat Control Pty. Ltd, 2-12 Penninsula Blvd, Seaford, Victoria, AUSTRALIA, 1999. Technical report.

G. Gigerenzer, P. M. Todd, and ABC Research Group. *Simple Heuristics That Make Us Smart*. Evolution and Cognition Series. Oxford University Press, 1999.

S. Gong and T. Xiang. Discovering bayesian causality among visual events in a complex outdoor scene. In *IEEE International Conference on Advanced Video- and Signal-based Surveillance*, pages 177–182, July 2003a.

S. Gong and T. Xiang. Recognition of group activities using a dynamic probabilistic network. In *IEEE International Conference on Computer Vision*, pages 742–749, October 2003b.

D. Greenhill, J. R. Renno, J. Orwell, and G. A. Jones. Learning the semantic landscape: Embedding scene knowledge in object tracking. In *Special Issue on Video Object Processing*, 2004.

W. E. L. Grimson and C. Stauffer. Adaptive background mixture models for real time tracking. In *CVPR*, 1999.

W. E. L. Grimson, C. Stauffer, and R. Romano. Using adaptive tracking to classify and monitor activities in a site. In *CVPR*, 1998.

A. Gritai, Y. Sheikh, and M. Shah. On the invariant analysis of human actions. In *17th conference of the International Conference on Pattern Recognition*, 2004.

I. Guyon and F. Pereira. Design of a linguistic postprocessor using variable memory length markov models. In *In International Conference on Document Analysis and Recognition*, volume 14, pages 454–457. IEEE Computer Society Press, 1996.

A. Hakeem and M. Shah. Learning, detection and representation of multi-agent events in videos. *Artificial Intelligence*, 171:586–605, 2007.

A. Hakeem and M. Shah. Ontology and taxonomy collaborated framework for meeting classification. In *ICPR04*, pages IV: 219–222, 2004.

B. Han, D. Comaniciu, and L. Davis. Sequential kernel density approximation through mode propagation: applications to background modeling. In *ACCV - Asian Conference on Computer Vision*, 2004.

S. Hannuna, N. Campbell, and D. Gibson. Identifying quadruped gait in wildlife video. In *International Conference on Image Processing*. IEEE, September 2005.

I. Haritaoglu, D. Harwood, and L. S. Davis. W4: Who? when? where? what? a real time system for detecting andtracking people. In *AFGR98*, pages 222–227, 1998.

I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Real-time surveillance of people and their activities. *PAMI*, 22(8):809–830, August 2000.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machcine Learning*, pages 197–243, 1995. ISSN 0885-6125.

J. Hoey and J. Little. Representation and recognition of complex human motion. In *CVPR*, pages 752–759, 2000. URL `citeseer.ist.psu.edu/hoey00representation.html`.

T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, August 1999.

S. Hongeng and R. Nevatia. Multi-agent event recognition. In *ICCV*, volume 2, pages 84–91, 2001.

R. Hosie, S. Venkatesh, and G. A. W. West. Classifying and detecting group behaviour from visual surveillance data. In *ICPR*, volume 1, pages 602–604, 1998.

A. Howard and T. Jebara. Dynamical systems trees. In *Uncertainty in Artificial Intelligence*, July 2004.

M. Hu. Visual pattern recognition by moment invariants. In *IRE Transactions on Information Theory, IT-8*, volume 2, 1962.

C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *Inernational Journal of Approximate Reasoning*, 15:225–263, 1996.

S. Indupalli, M.A. Ali, and B. Boufama. A novel clustering-based method for adaptive background segmentation. In *CRV06*, 2006.

S. S. Intille and A. F. Bobick. A framework for recognizing multi-agent action from visual evidence. MIT Media Laboratory, 1999.

S. S. Intille and A. F. Bobick. Recognizing planned, multiperson action. *CVIU*, 81(3): 414–445, March 2001.

M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the European Conference on Computer Vision*, pages 343–356, Cambridge UK, 1996.

M. Isard and A. Blake. Condensation : conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

Y. A. Ivanov and A. F. Bobick. Recognition of multi-agent interaction in video surveillance. unknown.

R. Jain and H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Trans. Patt. Analy. Mach. Intell.*, 1:206–214, 1979.

N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. In *Image and Vision Computing, 14(8)*, volume 14, pages 609–615, 1996.

N. Johnson, A. Galata, and D. Hogg. The acquisition and use of interaction behaviour models. In *In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition - CVPR'98*, pages 866–871. IEEE Computer Society Press, 1998. URL `citeseer.ist.psu.edu/article/johnson97acquisition.html`.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, page 38. IEEE Computer Society, 1996.

Carlo Colombo Alberto Del Bimbo Walter Nunziati Jurgen Assfalg, Marco Bertini. Semantic annotation of soccer videos: automatic highlights identification. In *Computer Vision and Image Understanding*, 2003.

T. Kadir, R. Bowden, E. J. Ong, and A. Zisserman. Minimal training, large lexicon, unconstrained sign language recognition. In *Proceedings of the 15th British Machine Vision Conference, Kingston*, page to appear, 2004.

I. A. Kakadiaris and D. Metaxas. Model-based estimation of 3d human motion with occlusion based on active multi-viewpoint selection. In *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 81. IEEE Computer Society, 1996.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

H. B. Kang and S. H. Cho. Short-term memory-based object tracking. In *International Conference on Image Analysis and Recognition (ICIAR)*. Springer Verlag, September 2004.

M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *International Journal of Computer Vision*, pages 321–331, 1988.

R. Kauth, A. Pentland, and G. Thomas. Blob and unsupervised clustering approach for object detection. In *ICCV95*, pages 786–793, 1995.

Yan Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *International Conference on Computer Vision*, volume 1, pages 166–173, 2005.

M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

S. M. Khan and M. Shah. Detecting group activities using rigidity of formation. In *MM05*, Singapore, November 2005.

M. Koivisto and K. Sood. Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Res.*, 5:549–573, 2004. ISSN 1533-7928.

Jeremy Kubica, Andrew Moore, David Cohn, and Jeff Schneider. cgraph: A fast graph-based method for link analysis and queries. In Marko Grobelnik, Natasa Milic-Frayling, and Dunja Mladenic, editors, *Proceedings of the 2003 IJCAI Text-Mining and Link-Analysis Workshop*, pages 22–31, 2003a.

Jeremy Kubica, Andrew Moore, and Jeff Schneider. Tractable group detection on large link data sets. In Xindong Wu, Alex Tuzhilin, and Jude Shavlik, editors, *The Third IEEE International Conference on Data Mining*, pages 573–576. IEEE Computer Society, November 2003b.

S Kullback and R. A. Leibler. On information and sufficiency. In *Annals of Mathematical Statistics*, volume 55, pages 79–86, 1951.

Jaimyoung Kwon and Kevin Murphy. Modeling freeway traffic using coupled hmms. *Machine Learning*, May 2000.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *18th International Conference on Machine Learning*, 2001.

I. Laptev and T. Lindeberg. Velocity-adaptation of spatio-temporal receptive fields for direct recognition of activities: An experimental study. In *ECCV Workshop on Statistical Methods in Video Processing*, pages 61–66, 2002.

M. Lee and R. Nevatia. Human pose tracking using multilevel structured models. In *ECCV 06*, 2006.

S. Lefevre and N. Vincent. Real time multiple object tracking based on active contours. In *International Conference on Image Analysis and Recognition (ICIAR)*. Springer Verlag, September 2004.

B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *CVPR*, volume 1, pages 878–885, 2005.

M. E. Leventon and W. T. Freeman. Bayesian estimation of 3-d human motion. Technical Report TR1998-006 17, Mitsubishi Electric Research Laboratories, July 1998.

X. Liu and C. S. Chua. Multi-agent activity recognition using observation decomposed hidden markov models. *Image and Vision Computing*, pages 166–175, 2006. URL `http://www.sciencedirect.com/science/article/B6V09-4HM7RW0-5/2/8458ff7a694e1a`

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

D. J. C. MacKay. Introduction to monte carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*, NATO Science Series, pages 175–204. Kluwer Academic Press, 1998.

David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

D. Magee. *Machine Vision Techniques for the Evaluation of Animal Behaviour*. PhD thesis, The University of Leeds, School of Computer Studies, 2000.

D. Magee. Tracking multiple vehicles using foreground, background and motion models. In *Image and Vision Computing*, volume 22, pages 143–155, September 2004.

D. Makris and T. J. Ellis. Spatial and probabilistic modelling of pedestrian behaviour. In *British Machine Conference*, volume 1, pages 557–566, September 2002.

C. Mikolajczyk, C. Schmid, and A. . Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV04*, volume 1, pages 69–82, 2004.

T. Minka. Expectation-maximization as lower bound maximization, 1998. Tutorial published on the web at http://www-white.media.mit.edu/ tpminka/ papers/em.html.

M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*. McGraw Hill, New York, 1975.

Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(4): 349–361, 2001. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.917571.

D. D. Morris and J. Rehg. Singularity analysis for articulated object tracking. *Proceedings of CVPR '98*, pages 289–296, June 1998.

R. J. Morris and D. C. Hogg. Statistical models of object interaction. *International Journal of Computer Vision*, 37(2):209–215, 2000.

multitel. Trictrac project, found at url: http://www.multitel.be/trictrac. Internet, Feb 2006.

V. Nair and J. J. Clark. Automated visual surveillance using hidden markov models. McGill University, 2002.

M. Naylor and C. I. Attwood. Advisor annotated digital video for intelligent surveillance and optimised retrieval. ADVISOR Consortium, May 2003. Final Report.

R. M. Neal and G. E. Hinton. A new view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*. Kluwer, 1998.

C. Needham. *Tracking and modelling of team game interactions*. PhD thesis, UNiversity of Leeds, 2003.

Nguyen, Venkatesh, and Bui. Recognising behaviours of multiple people with hierarchical probabilistic model and statistical data association'. In *BMVC06*, 2006.

N. T. Nguyen, H. H. Bui, S. Venkatesh, and G. A. West. Recognising and monitoring high-level behaviours in complex spatial environments. In *CVPR*, pages 620–625, 2003.

J. C. Niebles, H. Wang, and L. FeiFei. Unsupervised learning of human action categories using spatial-temporal words. In *British Machine Vision Conference*, Edinburgh, 2006.

N. Oliver, F. Berard, and A. Pentland. Lafeter: lips and face tracking. In S. Juan, editor, *IEEE International Conference on Computer Vision and Pattern Recognition*, Puerto Rico, June 1997.

N. Oliver, B. Rosario, and A. Pentland. Graphical models for recognising human interactions. nips, 1998.

N. M. Oliver. *Towards Perceptual Intelligence: Statistical Modelling of Human Individual and Interactive Behaviours*. PhD thesis, Massachusetts Institute of Technology, June 2000.

N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modelling human interactions. *IEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), August 2000a.

N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modelling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000b.

N. M. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. *icmi*, 1(1), August 2002.

C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In *Proc. of Intelligent Vehicles*, pages 241–246, 1998.

N. Paragios and R Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *Int. J. Comput. Vision*, 46:223–247, 2002.

S. Park and J. K. Aggarwal. Event semantics in two-person interactions. In *ICPR04*, pages IV: 227–230, 2004.

S. Park and M. M. Trivedi. Multi-person interaction and activity analysis: a synergistic track and body-level analysis framework. *Machine Vision and Applications*, 18:151–166, 2007.

G. Paschos. Perceptually uniform color spaces for color texture analysis: an empirical evaluation. *IEEE Trans. Image Process.*, 10:932–937, 2001.

V. Pavlovic, J. M. Rehg, T. J. Cham, and K. P. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *ICCV (1)*, pages 94–101, 1999.

A. Pentland. Classification by clustering. In S. Juan, editor, *IEEE Symposium on Machine Processing and Remotely Sensed Data*, Pardue, IN, June 1976.

M. Perse, M. Kristan, J. Pers, and S. Kovacic. A template-based multi-player action recognition of the basketball game. In J. Pers and D. R. Magee, editors, *ECCV Workshop on Computer Vision Based Analysis in Sport Environments*, pages 71–82, Graz, Austria, May 2006.

M. Perse, M. Kristan, J. Pers, and S. Kovacic. Automatic evaluation of organized basketball activity. In M Grabner and H. Grabner, editors, *Computer Vision Winter Workshop*, pages 11–18, St. Lambrecht, Austria, 2007.

M. Piccardi and T. Jan. Mean-shift background image modelling. In *IEEE International Conference on Image Processing (ICIP-2004)*, October 2004.

EC Funded CAVIAR project/IST 2001 37540. found at url: http://homepages.inf.ed.ac.uk/rbf/caviar/, 2004. URL `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`.

G. Pryor, P. Vela, T. Rehman, and A. Tannenbaum. Layered active contours for tracking. In *British Machine Vision Conference*, 2007.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. View-invariant alignment and matching of video sequences. In *The Ninth IEEE International Conference on Computer Vision*, Nice, France, 2003.

P. Ribeiro and J. Santos-Victor. Human activities recognition from video: modeling, feature selection and classification architecture. In *Workshop on Human Activity Recognition and Modelling (HAREM 2005 - in conjunction with BMVC 2005)*, pages 61–70, Oxford, September 2005.

N. M. Robertson. *Automatic Causal Reasoning for Video Surveillance*. PhD thesis, Heartford College, University of Oxford, June 2006.

N. M. Robertson and I. D. Reid. Behaviour understanding in video: A combined method. In *International Conference on Computer Vision (ICCV), Beijing, China*, pages 339–344, October 2005.

R. Rodriguez and A.G. Suarez. An image segmentation algorithm using iteratively the mean shift. In *CIARP06*, pages 326–335, 2006.

M. C. Roh, B. Christmas, J. Kittler, and S. W Lee. Gesture spotting for low-resolution sports video annotation. *Pattern Recognition*, 41:1124–1137, 2008.

D. Ron, Y. Singer, and N. Tishby. The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.

P. Rosin and T. Ellis. Image difference threshold strategies and shadow detection. In *British Machine Vision Conference*, pages 347–356, 1995.

N. Rota and M. . Thonnat. Activity recognition from video sequences using declarative models. In W. Horn, editor, *14th European Conference on Artificial Intelligence (ECAI 2000)*, Berlin, 2000. IOS Press, Amsterdam.

C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *Proc. ACM Siggraph*, 2004.

D. Rowe, I. Huerta, J. Gonzalez, and J. J. Villanueva. Robust multiple-people tracking using colour-based particle filters. In *IbPRIA07*, 2007.

D. Russell and S. Gong. Segmenting highly textured nonstationary background;. In *BMVC2006*, 2006a.

D.M. Russell and S. Gong. Minimum cuts of a time-varying background. In *British Machine Vision Conference*, 2006b.

M. S. Ryoo and J. K. Aggarwal. Semantic understanding of continued and recursive human activities. In *18th International Conference on Pattern Recognition*, volume 1, pages 379–382, Hong Kong, 2006a.

M. S. Ryoo and J. K. Aggarwal. Recognition of composite human activities through context-free grammar based representation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1709–1719, New York, NY, 2006b.

M. S. Ryoo and J. K. Aggarwal. Hierarchical recognition of human activities interacting with objects. In *2nd International Workshop on Semantic Learning Applications in Multimedia (SLAM) in conjunction with CVPR*, Minneapolis, MN, June 2007.

D. Salber, A. K. Dey, and G. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *CHI99*, pages 434–441. ACM Publ, 1999.

J. Saragih and R. Goecke. Learning active appearance models from image sequences. In *VisHCI '06: Proceedings of the HCSNet workshop on Use of vision in human-computer interaction*, 2006.

L. K. Saul and M. I. Jordan. Boltzman chains and hidden markov models. In G. Tesauro, D.S. Touretzky, and T. Leen, editors, *NIPS*, volume 7, Cambridge, MA, 1995.

L. K. Saul and M. I. Jordan. Mixed memory markov models: Decomposing complex stochastic processes as mixtures of simpler ones. *Machine Learning*, pages 75–87, June 1999.

R. C. Schank and R. P. Abelson. *Scripts, Plans, Goals and Understanding,*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

R. E. Schapire. The boosting approach to machine learning: An overview. Technical report, University of California, Princeton University Department of Computer Science 35 Olden Street Princeton NJ 08544, 2002. In MSRI Workshop on Nonlinear Estimation and Classification.

Bernhard Scholkopf. Statistical learning and kernel machines. Technical report, Microsoft Research Cambridge, Guildhall Street, Cambridge CB2 3NH, UK, February 2000. Technical report.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

P. Scovanner and M. Ali, S.and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM Multimedia*, pages 357–360, 2007.

A. Senior. Tracking people with probabilistic appearance models. In *PETS02*, pages 48–55, 2002.

A. Senior, A. Hampapur, Y. L. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. In *in proceedings of Second International workshop on Performance Evaluation of Tracking and Surveillance systems in conjunction with CVPR'01*, December 2001.

Y. Shi, D. Huang, Y. Minnen, A. F. Bobick, and I. A. Essa. Propagation networks for recognition of partially ordered sequential action. In *CVPR (2)*, pages 862–869, 2004.

Jamie D Shutler and Mark S. Nixon. Zernike velocity moments for description and recognitionof moving shapes. In *BMVC*, 2001.

H. Sidenbladh and M. J. Black. Learning image statistics for bayesian tracking. In *Int. Conference on Computer Vision, ICCV-2001, Vancouver*, volume 2, pages 702–716, June 2001.

H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *European Conference on Computer Vision, D. Vernon (Ed.*, pages 702–718. Springer Verlag, June 2000.

C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional models for contextual human motion recognition. In *International Conference on Computer Vision*, 2005.

K. Y. Song, J. Kittler, and M. Petrou. Defect detection in random color textures. *Israel Verj. Cap Journal*, 9:667–683, 1996.

M. B. Stegmann. Object tracking using active appearance models. In *Proc. 10th Danish Conference on Pattern Recognition and Image Analysis*, volume 1, pages 54–60, Copenhagen, Denmark, July 2001. URL `http://www.imm.dtu.dk/pubdb/p.php?115`.

T. Taki, J. I. Hasegawa, and T. Fukumura. Group motion features for teamwork evaluation and its application to soccer games. In *ICPR98*, page SA21, 1998.

M. R. Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 8(70):920–930, 1979.

Y. H. Tian, Z. Mei, T. J. Huang, and W. Gao. Incremental learning for interaction dynamics with the influence model. In *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 60–68, August 2003.

C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

V. J. Traver, A. Bernardino, P. Moreno, and J. Santos-Victor. Appearance-based object detection in space-variant images: a multi-model approach. In *ICIAR - International Conference on Image Analysis and Recognition*, September 2004.

T. Troscianko, A. Holmes, J. Stillman, M. Mirmehdi, and D. Wright. What happens next? the predictability of natural behaviour viewed through cctv cameras. *Perception*, 33(1):87–101, February 2004. ISSN ISSN 0301-0066.

D. Tweed, R. Fisher, J. Bins, and T. List. Efficient hidden semi-markov model inference for structured video sequences. In *2nd Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, (VS-PETS)*, pages 247–254, Beijing, October 2005.

T. Van Vu, F. Bremond, and M. Thonnat. Automatic video interpretation: A recognition algorithm for temporal scenarios based on precompiled scenario models. In *ICVS*, 2003.

N. Vaswani, A. R. Chowdhury, and R. Chellappa. Activity recognition using the dynamics of the configuration of interacting objects. In *Computer Vision and Pattern Recognition*, 2003.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 609–615, 2001.

P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *In 9th International Conference on Computer Vision*, volume 1, pages 734–741, October 2003.

S. Wachter and H. H. Nagel. Tracking persons in monocular image sequences. *Comput. Vis. Image Underst.*, 74(3):174–192, 1999. ISSN 1077-3142.

K. N. Walker, T. F. Cootes, and C. J. Taylor. Automatically building appearance models from image sequences. In *British Machine Vision Conference*, 1999.

H. M. Wallach. Conditional random fields: An introduction. Technical report, University of Pennsylvania, 2004. CIS Technical Report MS-CIS-04-21.

C. Wren, A. Azarbayejani, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:780–785, 1997a.

C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997b. URL `citeseer.ist.psu.edu/wren97pfinder.html`.

Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

J. Wu, J. M. Rehg, , and M. D. Mullin. Learning a rare event detection cascade by direct feature selection. In *NIPS*, 2002.

Xianghua Xie and Majid Mirmehdi. Implicit active model using radial basis function interpolated level sets. In *Proceedings of the 18th British Machine Vision Conference*, 2007.

M. Yamamoto. Incremental tracking of human actions from multiple views. In *CVPR98*, 1998.

A. Yilmaz and M. Shah. Actions sketch: a novel action representation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 20–25, June 2005.

L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Neural Information Processing Systems*, 2004.

D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, and G. Lathoud. Modelling individual and group actions in meetings: A two-layer hmm framework. IDIAP Research Institute, Martigny, Switzerland, 2000.

T. Zhao and R. Nevatia. Bayesian human segmentation in crowded situations. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2003.

S. Zhong and J. Ghosh. A new formulation of coupled hidden markov models. Technical report, Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, USA, September 2001a. Technical report.

S. Zhong and J. Ghosh. Distance-coupled hidden markov models. Technical report, Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, USA, September 2001b. Submitted to NIPS 2001.

J. Zhu, S. Rosset, H. Zhou, and T. Hastie. Multi-class adaboost. Technical report, University of Michigan, Ann Arbor, 2006.

S. Zhu and A. Yuille. Region competition: unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Trans. Patt. Analy. Mach. Intell.*, 18:884–900, 1996.