High-level activity learning and recognition in structured environments

by

John Patrick Greenall

Submitted in accordance with the requirements

for the degree of Doctor of Philosophy.

The University of Leeds

School of Computing

June 2012

Acknowledgements

I would like to thank all members of the Co-Friend consortium that assisted with the provision of data used in this thesis and provided many hours of interesting discussion on the research problem. This applies particularly to Krishna Dubba with whom I had regular debate in the formative stages of this work. I am grateful to my supervisors Tony Cohn and David Hogg for their patience, enthusiasm and excellent advice throughout my study. On a personal note I would like to thank all my friends and family for putting up with my sporadic absences from social occasions at various points in the last 4 years. Particular thanks goes to Amelia for keeping me fed and looked after during the final stages of the work.

Declarations

The candidate confirms that the work submitted is his own, except where work which has formed part of a jointly-authored publication has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

JP Greenall, AG Cohn, DC Hogg, "Temporal Structure Models for Event Recognition", *British Machine Vision Conference,* , (2011) .

The paper above forms the basis of chapter 4 of this thesis. All research on this paper was conducted solely by me, under the guidance of my supervisors David Hogg and Tony Cohn.

Abstract

Automatic recognition of events in video is an immensly challenging problem. If solved, the number of potential domains in which such a system could be deployed is vast and growing; including traffic monitoring , surveillance, security, elderly care and semantic video search to name but a few. Much prior research in the area has focused on producing a solution that is tailored towards one of these applications, applying methods which are most appropriate given the constraints of the target domain. For the moment, this remains to some extent the only practical way to approach the problem. The aim in this thesis is to build a high-level framework for event recognition which is in the main generic and widely transferrable, yet allows domain-appropriate elements to be incorporated.

A detector is constructed for low-level events which is based on dense extraction of Histograms of Optical Flow. This descriptor has only recently been adopted by the event detection community, and as such there are aspects of the features which have not been optimized. This thesis performs extensive experimentation on normalization scheme and finds that the strategy most widely in use is suboptimal compared to one of the alternatives proposed. The detector is then trained on a challenging realworld domain to run in a sliding window fashion on continuous video input.

A high level model which exploits temporal relations between different event types is constructed. The model is designed with transferrability and computational tractability in mind. Several methods are benchmarked for learning the distributions over time differences between pairs of events. Three different connection strategies are proposed and evaluated for creating a tree structured prior that permits fast, exact inference. An efficient iterative optimization scheme is presented for handling scenarios which contain unknown numbers of event instances. Finally, the model is extended in a Conditional Random Field framework that allows weights to be learned to balance the response from independent detectors with the pairwise temporal relationships.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic recognition of events in video is an immensly challenging problem. If solved, the number of potential domains in which such a system could be deployed is vast and growing; including traffic monitoring , surveillance, security, elderly care and semantic video search to name but a few. Much prior research in the area has focused on producing a solution that is tailored towards one of these applications, applying methods which are most appropriate given the constraints of the target domain. For the moment, this remains to some extent the only practical way to approach the problem. The aim in this thesis is to build a high-level framework for event recognition which is in the main generic and widely transferrable, yet allows domain-appropriate elements to be incorporated.

The focus throughout this thesis is the recognition of events within a *scenario*, where a scenario is a grouping of related simpler events into a more complex pattern of activity. In many event recognition domains this notion is quite natural. Television programs are divided into episodes, sports footage is separated into games. Even in domains where the video input is continuous such as traffic monitoring, there may be natural ways to divide the footage up; such as the morning commuter rush, midday lull, evening rush etc. The high level temporal structure model described in this thesis is applicable in any domain where such a notion of a scenario is natural, and there exists some repeated temporal

structure.

The dataset which is the primary focus for the latter part of this thesis is the Co-Friend dataset. This dataset was created as part of the Co-Friend EU Framework 7 project.[1] The end goal of Co-Friend was to create a system which could reliably recognise all the various aircraft servicing events which take place on the airport apron, to assist airlines who would like to record timings in order to optimise turnaround procedures.

The initial phase of the research for this thesis involved researching ways of recognising events from the output of a multi-camera tracking system which was the research focus of another of the academic partners on the project. Many of the methods touched upon in the review of high level methods were applied to this data. Each method seemed to hit problems due to some element of noise arising from the tracker (which was itself an active research project). Clustering or reasoning with complete trajectories was very difficult due to the inability of the tracker to maintain consistent IDs and detect subtle motion in the challenging conditions inherent to the domain. It became clear that the available tracking information was not a sufficiently solid foundation on which to build a higher level system. Other recent research in the area, summarized in Chapter 2, led to the idea of working directly with the video and image-based features instead.

A further observation whilst performing the literature review for this work was that a great deal of research within the machine learning community has recently been directed at designing efficient probabilistic classifiers. At the time of writing, Support Vector Machines [16], Random Forests [13] and Gaussian Processes [65] are some of the most prevalant methods in the field. Additionally, there is almost as much research on the subject of squeezing as much performance as possible out of any given classifier with techniques such as bootstrapping [53] , boosting [67] and cascades [80]. Therefore it is very desirable to be able to leverage this body of work when building an event detection system. The obvious way to do this is to train a classifier for each event type and to detect

[1]The project was a collaborative effort between Universities of Leeds, Reading, Hamburg, INRIA and the private company AKKA.

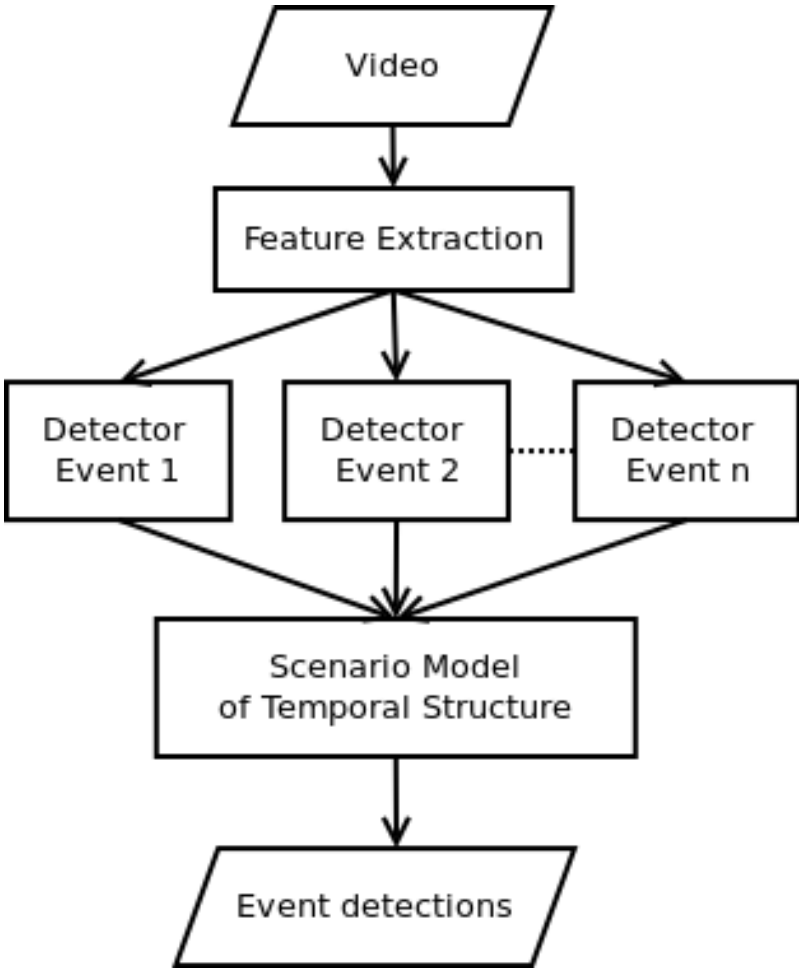


Figure 1.1: Flow diagrams describing the proposed pipeline for event detection within scenarios with temporal structure.

instances of each event type independently with a sliding window detector that works by extracting features over each possible interval and passing those features to a classifier. Detections are then obtained then by simply taking thresholded local maxima. The problem with this method when applied to event detection in structured environments is that it ignores the useful temporal dependencies between events. The principle contribution of this thesis is to propose a framework for combining the response from independent probabilistic classifiers with probabilistic distributions over time differences observed between instances of events.

A significant degree of effort was put into testing and building the feature extraction

component of the proposed pipeline. Chapter 3 details the implementation of an event detector which extracts local features and uses a bag of word representation with a nonlinear classifier to give a probabilistic output for all candidate intervals in a video sequence. As this detector builds on recent research, several design choices are encountered for which there is no extensive evaluation available in the literature. One of the contributions of this thesis is a set of experiments which goes further than any previous work in evaluating several aspects of the popular 'Histograms of Optical Flow' descriptor.

In Chapter 4 a model is defined for efficiently combining the response from independent event detectors with a prior over the structure of inter-event timings. The model is designed to cope in situations where events of different types can potentially overlap and are not strictly ordered. These are precisely the conditions which cause problems for state-based models such as HMMs that attempt to encode the state of the world at each time step with a latent variable. In the optimization defined here, the temporal midpoints and durations of event instances in a previously unseen sequence are the latent variables to be optimized. The method only requires that the timings of at least some of the observed events are loosely correlated. The greater the degree of correlation, the greater the benefit will be; whilst the absence of parameters in the model makes it generic and easily applicable to other domains. The method is evaluated on the large real world Co-Friend dataset.

In Chapter 5, the model defined in the previous chapter is extended by reformalizing the problem as a Conditional Random Field. This allows the introduction of additional weighting parameters into the model and provides a well-founded methodology for training these parameters. Evaluation is provided which shows this relatively straightforward CRF reformulation to perform marginally better than the simpler model. Several suggestions are also proposed for future work in the area involving more densely connected models which would require further research into efficient approximate inference algorithms outside the scope of this thesis.

In Chapter 6, the findings of the thesis are summarized.

Chapter 2

Related Work

Previous works on human action recognition can be divided into two broad categories: the low-level, which focus on the recognition of short basic actions involving a single participant and the high-level, which focuses on recognising more semantically complex events, possibly involving multiple participants and temporally dependent sets of actions. The former has received much attention from the vision community in recent times and there has been intense competition across several datasets. Research into high-level event recognition has been comparatively sparse, and no dataset seems yet to have captured the support of the community, with individual studies typically reporting performance only on their own dataset. This thesis describes a methodology for tackling the high-level problem, through exploitation of methods which have proved successful at the lower level. Thus a review of both areas is considered to be useful.

Figure 2.1: Samples from some of the most frequently used datasets. Columns from left to right are Weizmann Actions, KTH Actions and Hollywood2

2.1 Low Level Action Recognition

A significant portion of the discussion in this section is devoted to the feature extraction problem as this is the area where much of the innovation has come. Once appropriate features have been engineered and extracted, the classification part of the task is often solved through a relatively straightforward application of a standard machine learning method.

2.1.1 Common Datasets

The most popular datasets in this area have so far consisted of many short clips of actions from different classes: thus mapping the task to a classification problem. A brief description of some of the most popular datasets follows. The Weizmann Actions dataset consists of 90 video clips; 9 different people performing 10 different actions. The video is low resolution (180x144) and each clip consists of a single actor performing a single

Figure 2.2: Illustration of MEI (centre) and MHI (right) for an aircraft arrival event taken from the *cofriend* dataset.

action. There is no occlusion and very little variation in scale. The KTH Actions dataset is larger; consisting of 2391 sequences also at low-resolution (160x120). There are 6 classes of action performed by 25 different actors under varying conditions - indoors, outdoors, with different clothes and at slightly different scales. This dataset was introduced by Schuldt *et al*. in [68] and was probably the most widely used benchmarking dataset between 2004-2010. Interest in the dataset is now fading as several authors have recorded near-perfect results and more challenging datasets are sought. Hollywood 2 is one such dataset, consisting of around 1600 clips sampled from Hollywood movies with actions of 12 different classes. Stanford Olympic Sports [57] is another similarly challenging dataset. Common to both the newer datasets is that the camera angle is highly variable and there is much greater occlusion. Essentially the move is towards footage which is much more natural and unconstrained than in the previous generation of datasets.

2.1.2 Methods Based on Global Features

An interesting early contribution in this area is that of Davis and Bobick [19]. The authors first use a crude global descriptor to coarsely categorize human action in video and thereafter outline a pipeline which involves segmenting an image into salient patches. The video sequence is then described by the transformations that describe the motion of these patches. The implementation of that stage in the pipeline is listed as future work

and requires manual intervention to assist with the intialization and registration of the patches. One significant innovation in this paper is the introduction of Motion-Energy Images (MEI) as a simple global descriptor. Later work [10], refines the definition of the MEI to the following:

$$E_\tau(x,y,t) = \bigcup_{i=0}^{\tau-1} D(x,y,t-i) \tag{2.1}$$

where $D(x,y,t)$ is a binary image sequence indicating the regions of motion and τ is an integer representing the size of the temporal window. Thus the MEI at time t simply indicates whether any motion has been observed at each pixel in the last τ frames. In [10], the MEI is once again used as a crude global descriptor, however the earlier patch-based approach at the finer grained level is abandoned; the reasons cited for this are the difficulties inherent in the automation of the patch identification, tracking and registration procedure. This is an observation that will arise several times within the context of this thesis: higher level systems which are highly dependent on unstable or imperfect tracking are likely to be less reliable than those which can work directly with simpler features. On this basis, David and Bobick introduce another simple global descriptor: the Motion-History Image (MHI):

$$H_\tau(x,y,t) = \begin{cases} \tau & \text{if } D(x,y,t) = 1 \\ \max(0, H_\tau(x,y,t-1)-1) & \text{otherwise} \end{cases} \tag{2.2}$$

where D, τ are defined as before. Essentially, the MHI encodes the time of the most recent motion at each pixel. Figure 2.2 shows an example of an MHI and MEI on the same sequence. It is clear that the MHI is a much richer representation than the MEI; but the authors insist that the MEI is still useful (though this assertion is refuted in [86]). Hu moments are taken of both the MEI and MHI in order to abstract to features which have a degree of scale, rotation and translation invariance. The key problem with these features is that they assume all motion in the image should be incorporated into the templates. With

more than one person in the field of view this would clearly be inappropriate. Though it might be possible to use tracking and extract features from within bounding boxes; significant problems will still be caused by occlusion since the moments of the MEI/MHI for a partially occluded action won't necessarily bear much similarity to those of the unrestricted view.

The work of Davis and Bobick is extended in [86] to be 'view invariant'. This is done using a multicamera system which captures motion in 3D that can then be used to create voxel-based Motion History Volumes. View invariance is gained by aligning volumes using Fourier transforms. The major limitation with this method is that it requires multiple calibrated cameras overlooking the same scene and a reliable 3D reconstruction algorithm, which is an unsolved problem for complex real-world domains.

Ragheb *et al*. [63] also use segmented binary silhouettes as input for human action recognition. An activity volume is created by aligning and scaling images from a sequence around the centre of mass of the human silhouette. The volume is then divided spatially into a number of different windows and power spectrum features computed for each window. A nearest-neighbour scheme based on a weighted euclidean distance measure is then used to classify actions. Strong results are published on the Weizmann actions dataset.

Similar work that focuses on recognising action in much lower quality video such as (pre-HDTV) broadcast sports footage is that of Efros *et al*. [22]. Here, video is tracked to get person-centered regions of a fixed scale. The optical flow is then smoothed with a Gaussian kernel and split into four non-negative channels. Rather than abstracting to a compact descriptor, the authors define a similarity kernel with which to compare flow fields after which action recognition is performed using Nearest Neighbour classification. The form of the kernel

$$S(i,j) = \sum_{t \in T} \sum_{c=1}^{4} \sum_{x,y \in I} a_c^{i+t}(x,y) b_c^{j+t}(x,y) \tag{2.3}$$

suggests that the method's robustness to occlusion could be superior to the Hu Moments used in [10], since in comparing an occluded to a non-occluded prototype of the same action, there would still be some regions of correlation. However, the lack of evaluation on a common dataset means this cannot be easily verified.

A more recent work which shares many similarities with the works of [10, 22] is that of Ali and Shah [1]. Here, several global features have again been carefully chosen to compactly describe each video frame. In this case they are calculated from the dense optical flow field in each video frame and are termed *kinematic* features; defined as follows: *Divergence* is a scalar quantity borrowed from fluid dynamics and represents the amount of local expansion taking place. *Vorticity* is a scalar measure of local spin around the axis perpendicular to the plane of the flow field. *Symmetric* and *Assymetric Fields* are calculated by the sum and the difference between the flow field and its transpose respectively and emphasise the symmetry or assymmetry in the observed action. Several rotation-invariant features are also derived from the *Gradient Tensor*:

$$\nabla U(\mathbf{x}, t_i) = \begin{pmatrix} \frac{\partial u(\mathbf{x},t_i)}{\partial x} & \frac{\partial u(\mathbf{x},t_i)}{\partial y} \\ \frac{\partial v(\mathbf{x},t_i)}{\partial x} & \frac{\partial v(\mathbf{x},t_i)}{\partial y} \end{pmatrix}. \tag{2.4}$$

where u,v are the horizontal and vertical components of the optical flow field. Analysis is performed on a standard dataset which verifies that none of the features defined are redundant. Snapshot Principal Components Analysis (PCA) is used on these features to obtain *Kinematic Modes*. An action classifier is then learned using multiple instance learning with each video represented as a bag of kinematic modes. Results reported on the KTH dataset are competitive with the state of the art but cannot really be directly compared as the authors admit to a greater degree of temporal annotation (breaking videos down to single cycles of repetitive action) than normally used with the dataset. As in the methods which have so far been investigated in this section, it is necessary to perform person localization and then rescaling of the image to a fixed scale before this method can

be applied and it is difficult to get a sense of how robust the features will be to occlusion.

This is the primary reason that global-feature based methods don't currently seem competitive on more complex recent datasets such as Hollywood 2 and UCF Sports and in general, this is why Bag of Words (BOW) methods based on local descriptors have been so dominant in the literature over recent years. Given that the primary dataset of interest in this thesis contains actions involving multiple agents operating at times under heavy occlusion, the focus of the review is now switched to action recognition systems based on local features. Global representations should not be overlooked completely, however. For example, the MHI as a descriptor has several attractive properties: They are cheap to compute (only requiring a basic motion detector to be applied), can be intuitively visualized, and encode enough information to be discriminative. In early experimentation for Chapter 3, the MHI's viability as a local descriptor was tested on the Hollywood dataset; yet results were significantly poorer than features extracted based on optical flow, so experimentation was discontinued.

This discussion is concluded with a method which incorporates some local weighting into a global feature representation. In [8, 30], actions are regarded as space-time shapes created from segmented silhouettes. Each space-time point within a shape is assigned the mean time required for a particle undergoing a random-walk, starting from the point, to hit the boundaries. From this representation, several local features are defined based on gradients and orientations. Video clips are segmented into space-time cubes of fixed duration and scale and global descriptors are created for each cube by taking weighted moments of the local features. A simple Nearest-Neighbour classification scheme is then used for evaluation. The features are shown to outperform standard space-time shape moments and demonstrate some robustness to partial occlusion and other anomalous factors. However, the set of 'difficult' sequences they have constructed is quite limited and slightly contrived. There is some small amount of occlusion in some of the clips (a wall of boxes reaching barely above ankle-height) and other complicating noise such as walking a dog

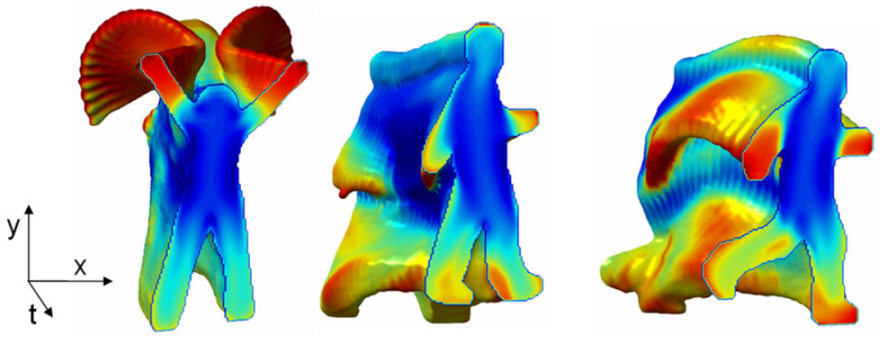


Figure 2.3: Illustration of the local space-time saliency features used in [8]

or walking in an unusual way but the manufactured difficulties really aren't representative of those that are likely to be encountered in the kind of real-world footage that will be the focus of this thesis. Hence the review proceeds to local features.

2.1.3 Methods Based on Local Features

This discussion is separated into three sections to reflect the elements in an pipeline which is common amongst almost all work that involves local features. First there is *feature detection*, which determines where the pertinent features in the video are located. Next there is *feature extraction*, which involves choosing an appropriate *feature representation* for the volumes which have been identified by the feature detector. Finally there is *event classification* or *detection*, which involves predicting which event(s) is contained within a given video volume.

2.1.3.1 Local Feature Detectors

Many of the most successful approaches employing local features for action recognition have employed feature detectors to determine where (in the video volume) they should be extracting features. The criteria one would hope a strong feature detector would fulfil are as follows:

- Features detected should capture as much relevant motion as possible,

- Features should be repeatable, meaning the same things captured at different scales,
- Features need to be significantly sparser to speed video processing or else show improved performance over dense sampling.

With these things in mind, one of the most popular methods used in the literature is summarized.

The first major study to motivate the use of local features in video analysis is the work of Laptev [44], which highlights the benefits of an action recognition system that can function without the need for segmentation, tracking or object flow computation. Laptev extends the Harris [31] corner detector to three dimensions to detect space-time interest points in video. This detector uses spatio-temporal image gradients within a Gaussian neighborhood of each point to find corners. Firstly the image sequence, f, is translated into its scale space representation by convolution with a separable Gaussian kernel. The parameters of this kernel define the *local* scale for smoothing prior to the computation of image derivatives.

$$L(\cdot;\sigma_l^2,\tau_l^2) = \mathcal{N}(\cdot;\sigma_l^2,\tau_l^2) * f(\cdot) \tag{2.5}$$

Next, the second moment matrix is computed as follows

$$\mu(\cdot;\Sigma) = \mathcal{N}(\cdot;\sigma_i,\tau_i) * (\nabla L(\nabla L)^T) \tag{2.6}$$

Where the parameters of this Gaussian are known as the *integration* scale for accumulating the non-linear operations on derivative operators into an integrated image descriptor. The assertion made in [31] is that for perfectly uniform regions, this second moment matrix will have zero-valued eigenvalues, a strong edge would have one large eigenvalue and one near zero whilst a corner would have two eigenvalues of reasonable size. Since the aim is to detect corners, one then seeks to find the coordinates where there are two large eigenvalues. However since calculating the eigenvalues is relatively expensive, the

following response function is used as an approximation.

$$H = \det(\mu) - \kappa \text{trace}^3(\mu) \tag{2.7}$$

Space-time corners are then identified by taking the positive maxima of this function. The value of κ is determined empirically and Laptev suggests a value of $\kappa = 0.005$ is appropriate in many cases. Also defined in this paper is a scale-adaptive algorithm, which initially searches over a sparsely defined range of scales to find a set of interest points at various scales before iterating over the interest points and optimizing the response of each one with respect to its scale and location.

An alternative interest point detector is described in the 2005 paper by Dollar *et al*. [20]. This paper claims that the space time corners just described are well suited to recognizing actions that are characterized by the reversal in direction of arms and legs (referring here to KTH), yet are too seldom found in activities characterized by more subtle motions. The question is raised as to whether space-time corners are indeed suitable features for general behaviour recognition: the features one needs are those which maximize discrimination between behaviours. Neither a spinning wheel nor the jaw of a chewing horse give rise to any space-time corners, as their motion appears gradual. Yet these motions could be relevant features for behaviour recognition, clearly indicating that a vehicle may be moving or a horse is eating. The proposed alternative to the corner detector is a response function calculated by application of separable linear filters.

$$R = (I * \mathcal{N} * h_{ev})^2 + (I * \mathcal{N} * h_{od})^2 \tag{2.8}$$

where $\mathcal{N}(x, y; \sigma)$ is the 2D Gaussian smoothing kernel applied to spatial dimensions and h_{ev} and h_{od} are a quadrature pair of 1D Gabor filters applied temporally. The strongest response under this detector will be caused by periodic motion such as a bird flapping its wings, yet it is claimed any region with spatially distinguishing characteristics undergoing

a complex motion will induce a strong response. There is some limited evaluation done to demonstrate the effectiveness of this detector versus the Harris 3D detector on two small datasets: The first of these involves human actors expressing different emotions through exaggerated facial expressions in a series of short clips. The second is of the behaviour of a single mouse in a cage. Whilst the datasets are too limited to be of wider interest, they do help further the argument against space-time corners; though it is not clear whether any effort has been made to tune the κ parameter used to control the sensitivity of the corner detector and there is no comparison against dense sampling.

Willems *et al*. see problems with both representations and propose a further detector in [87]. Dollar's detector is criticised due to the lack of scale invariance in the cuboid detector. Laptev and Lindeberg's corner detector is criticised due to the computational cost of the iterative procedure that is used to optimise the scale of features restricting the density of features that it is feasible to extract. The method of Willems is in spirit similar to the Harris 3D detector but it instead uses the scale-normalized determinant of the Hessian matrix

$$H(\cdot,\sigma,\tau) = \begin{pmatrix} L_{xx} & L_{xy} & L_{xt} \\ L_{yx} & L_{yy} & L_{yt} \\ L_{xx} & L_{ty} & L_{tt} \end{pmatrix} \tag{2.9}$$

as its response function. Due to some convenient properties of the Hessian, it is possible to perform detection over multiple scales by simply taking local maxima over the 5D space defined by (x, y, t, σ, τ) meaning it is very easy to vary the sparsity of the features extracted without significantly increasing computation. The evaluation section in this paper introduces some nice ideas. The criterion on which the three different feature detectors are evaluated is *repeatability*, that is how often the same spatio-temporal features are extracted in spite of various geometric and photometric transformations. They take a set of video sequences then apply various artificial transformations to see how the repeatability

Figure 2.4: Comparison of Bregonzio's interest points (shown in green) versus the cuboid detector (shown in red), taken from [12]. On this evidence, the cuboid detector picks out lots of irrelevant regions whilst missing salient motion. The apparent effectiveness of Bregonzio's method is largely down to a frame differencing step which works well for the static backgrounds and slow camera motion of the KTH dataset.

of the three detectors varies with scale change, in-plane rotation, camera motion and video compression. The evaluation shows the Hessian method to be consistently strong under all complicating factors, whilst the Harris 3D detector struggled with scale change and compression and the cuboid detector struggled with camera motion. A limitation with the reported evaluation is that although they say they experimented on several sequences, the results they show are taken from just one of those.

A later paper by Bregonzio *et al.* [12] highlights problems with the cuboid detector, namely that it ignores pure translational motion, is particularly prone to false detection due to video noise and noise in areas of high texture. Therefore a new detector is proposed which first contains a frame differencing step 'for focus of attention and region of interest detection', then does filtering in the spatial domain with a combination of 2D Gabor filters

at 5 different orientations and a 2D Gaussian envelope. Figure 2.4 shows examples where Bregonzio's detector appears to be highlighting more relevant features than the cuboid detector. Unfortunately, quantative evaluation of this detector is not provided. The frame differencing step which appears to work well on the KTH dataset would most likely be much less helpful on more challenging datasets.

Finally, a recent review paper [82] which describes extensive evaluation over the KTH, UCF sports and Hollywood2 datasets appears to suggest that in fact, there is no clear winner amongst the detectors thus far discussed. There is no evaluation of the feature detectors in isolation in this study but rather evaluation of combinations of feature detectors and descriptors applied within a fixed pipeline to the task of classification. The performance statistics suggest that the nature of the feature detector used is actually less important than the descriptor used; and that in the case of the more complex datasets dense sampling may in fact be preferable (if the computational expense can be borne). This intuition is applied in Chapter 3, with the implementation of a low-level event detector that uses dense sampling.

2.1.3.2 Local Feature Representations

In Laptev and Lindeberg's early work on local features [46], 'spatio temporal jets' are introduced as features to be used in recognition. These are vectors of third order partial normalized derivatives. These features have also been used to achieve respectable results on the KTH dataset [68]. This work is expanded in a later work by Laptev [45], which introduces some invariance to camera movement by using a response function which is made invariant to Gallilean transformations.

In the work of Dollar *et al*. [20], a cuboid is extracted at each interest point which contains pixel values. The size of the cuboid is set to contain most of the volume of data that contributed to the response function at the interest point, meaning six times the scale

at which they were detected. Pixel values are then transformed into either (1) normalized pixel values, (2) brightness gradient (on three channels G_x, G_y, G_t) or (3) optical flow (on two channels V_x, V_y). The cuboid is then flattened into a feature vector, the dimensionality of which is reduced by Pricipal Components Analysis. Gradient proved most effective experimentally and was preferred in later experiments. Experiments were also performed to check whether representing the cuboids as histograms over smaller features improved performance, but the results suggested otherwise.

In a later paper [47], the 3D Harris detector is paired with features from space-time cuboids around interest points. The volume is subdivided into a grid of smaller cuboids and 3D Histogram of Oriented Gradient (HOG3D) and Histogram of Optical Flow (HOF) features are calculated for each cuboid. Normalized histograms are concatenated into HOG3D or HOF vectors. An in depth explanation of these features follows in Chapter 3.

The features so far described have been hand-crafted to encode sufficient information to be discriminative whilst retaining various properties of invariance to increase their robustness. However, several works have attempted to learn the descriptors themselves by reference to the task to which they will be applied. One such example is the work of Fathi and Mori [25], which concentrates on working with five frame blocks of figure-centric images of fixed size. They calculate optical flow over the image sequence then blur, normalize and split the image into five non-negative channels corresponding to up, down, left, right motion plus another channel corresponding to the l^2 norm over the other four channels. Each voxel within the block is then considered to be a 'low-level feature' which can be used to discriminate between action classes by means of a simple threshold. The threshold is optimised for each feature independently to give a large number of (exceedingly weak) classifiers. Next, 'mid-level' features are defined as a weighted sum of all low-level classifiers within a cuboid of fixed size. The weightings are learned using Adaboost. Classification is ultimately performed with a weighted sum over the mid-level features, with weights again trained with Adaboost. Evaluation is performed on KTH,

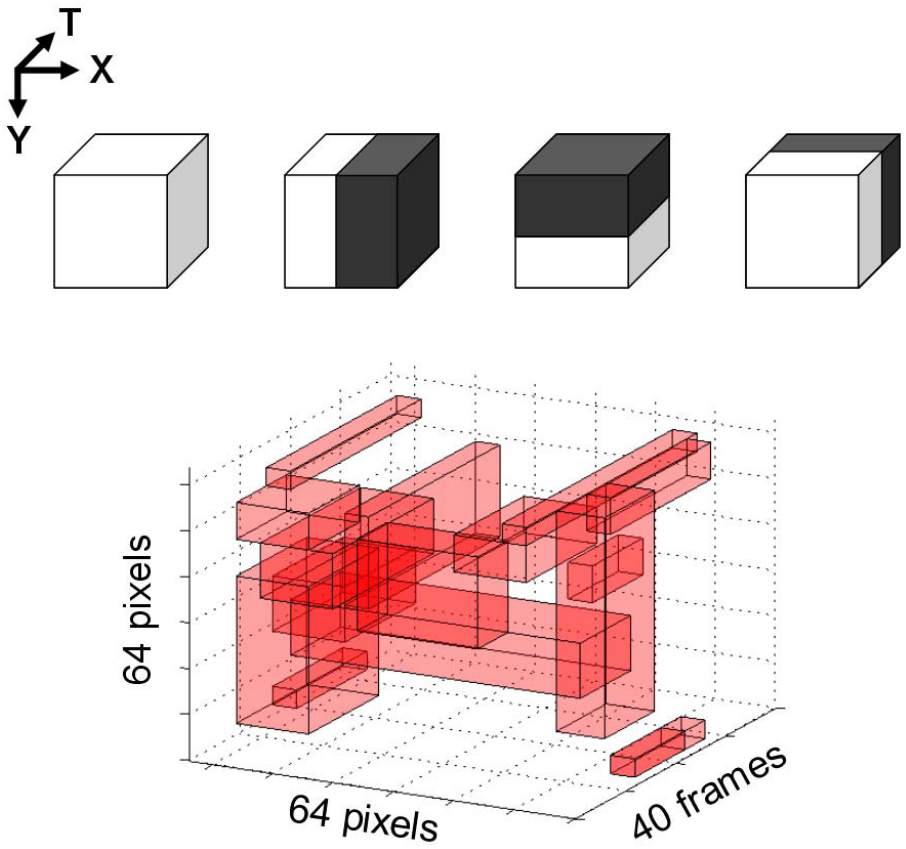


Figure 2.5: Depiction of features used by Ke *et al*. Figure taken from [40] The top row illustrates the 3D volumetric features used in their classifiers. The first feature calculates the volume. The other three features calculate volumetric differences in X, Y, and time. The bottom row shows multiple features learned by the classifier to recognize the handwave action in a detection volume.

displaying results close to state-of-the-art. Performance is also improved over [22] on the soccer dataset introduced by Efros *et al*.

An earlier example of this kind of approach is the work of Ke *et al*. [40]; where again person-centric volumes of fixed size are extracted from training videos. Millions of potential features are considered, composed from four basic types of cuboid at various scales and positions within the larger volume. The learning process involves the training of a cascade of single-feature classifiers which thus selects a subset of these features in order to achieve a target level of performance. This approach is heavily inspired by the work of Viola and Jones [81], which has proved immensely successful in the field

of object detection. The results published for the KTH dataset however are relatively unimpressive, suggesting that the approach needs more refinement to work as well in the activity recognition domain. One issue is that Viola-Jones inspired face detectors are typically trained on thousands of images to achieve a good level of performance. The KTH dataset in contrast comprises only hundreds of examples per class and the video clips are of higher dimensionality than the relatively tiny images used to train face detectors. Treating the temporal information as just another dimension means the data is much more sparse.

The evaluation for these two methods however don't evaluate the quality of the features learned but rather the overall recognition performance of the system. It isn't clear how one might make the features independent of the classification strategy in order to evaluate them in isolation. These are interesting contributions but they are highly dependent on accurate pedestrian tracking in order to obtain training data, making them susceptible to most of the criticisms levelled at the global feature representations discussed earlier.

Returning to the review paper of Wang *et al*., their evaluation shows that HOF features used in conjuction with the Harris3D corner detector are proven to be the most effective when deployed on the KTH dataset. Bregonzio *et al*. made one of the most recent and successful attempts to tackle the KTH dataset [12]. They first deploy their feature detector, which was described earlier, in order to get a response over a frame, before an edge detector is employed to segment regions of high response (object of interest) from the foreground. Once the object is segmented, two features are computed, measuring the height/width ratio of the object and the speed of the object. Further features are extracted from interest point 'clouds' of different scales from within the object. Features extracted include height/width ratio of cloud, density of feature points within it and the absolute speed. Whilst the paper is well cited, there is no record in later work of the features being applied to more recent or difficult datasets. This fact, combined with the method's

reliance on frame differencing and object segmentation raises the question as to whether it is generally applicable.

2.1.4 Action Classification Methodologies

For datasets such as KTH, Hollywood and Wiezmann the action recognition task is mapped to a classification problem. Therefore, the majority of papers which evaluate on these datasets use classifiers. N-Nearest Neighbour has been used in several works [8, 22, 68] and in the work of Schuldt *et al*. [68] was compared to a Support Vector Machine (SVM). In this, the SVM was found to marginally outperform the N-Nearest Neighbour method so was preferred in the final system. This agrees with trends in the wider machine learning community where SVMs have been quite dominant in classification tasks [79]. Several recent works have thus used the SVM to good effect [20, 47, 68].

Schuldt *et al*. [68] evaluate two different methods of exploiting features. The first representation considers each sequence as a bag of features and the kernel between two clips as

$$\hat{K}(\mathbf{L_h}, \mathbf{L_k}) = \frac{1}{n_h} \sum_{j_h=1}^{n_h} \max_{j_k=1,\ldots,n_k} \left\{ K_l(\mathbf{l}_{j_h}, \mathbf{l}_{j_k}) \right\} \tag{2.10}$$

where $\mathbf{L_h} = \{\mathbf{l}_{j_h}\}$ and $\mathbf{l}_{j_h}$ is a descriptor of an interest point j in sequence h and likewise for $\mathbf{L_k}$.

$$K_l(x, y) = exp\left\{ -\rho \left(1 - \frac{\langle x - \mu_x, y - \mu_y \rangle}{\|x - \mu_x\| \cdot \|y - \mu_y\|} \right) \right\} \tag{2.11}$$

Intuitively, this is comparing each feature in bag h to each feature in bag k and matching to the most similar feature. The total similarity between the two bags is then the sum of these scores.

The second method first applies k-means clustering to build a codebook of features then each video sequence is modelled as a histogram over these features, with the kernel between clips in the histogram representation simply being: $K(x, y) = exp\{-\gamma\chi^2(x, y)\}$. Experimental results show that the bag of local features does marginally better than the

histogram, though in later works by the authors the histogram approach is favoured. This is most likely due to the cost of evaluating the kernel between two clips, h and k, 2.10 being proportional to $D * N_h * N_k$, where D is the dimensionality of the feature, N_h, N_k are the number of features in each clip.

In the work of Dollar [20], a dictionary of cuboid prototypes is also built by k-means clustering of training data. Video clips are represented as histograms over cuboids and classified by SVM. Laptev's more recent work [47], also follows the exact same pipeline, albeit with a multi-channel kernel used to combine features of varying shapes and scales. There are several strengths to this pipeline: firstly the bag of words representation ensures clips of arbitrary length can be represented with a fixed length vector, meaning clips of different lengths can be easily compared. Secondly, because of the popularity of the SVM across the machine learning community in recent times, there are several mature and efficient implementations which can be easily deployed. In the experiments in Chapter 3, LibSVM [15] was deployed.

2.2 High Level Scenario Recognition

2.2.1 State-Based Methods

One popular approach to high-level scenario recognition is to have latent variable(s) storing the state of the world at each timestep, and to model the probability distribution over transitions between states.

The work of Hongeng and Nevatia [32,33] is a representative example of a type of approach used often in surveillance type applications. Agents are tracked, and these tracks then trigger atomic events based on their relative (agent A approaches agent B) or absolute movement (agent A stops). More complex activities are defined by single and multi-thread Finite State Machines. Simliar in spirit is the work of Makris *et al.* [49], whereby motion tracking across multiple overlapping cameras is used to abstract away

from image sequences to moving blobs in 3D space. The tracking information is used to build semantic models of the scene, typical routes through it. Suspiscious behaviour is then detected by learning a Hidden Markov Model over routes through the scene. HMMs are often deployed in this kind of context, where there is a single agent (or multiple agents whose activity is assumed independent) and a single state variable due to their computational convenience [14, 58, 73, 77, 89]. One of the main limitations of HMMs however is that the number of parameters in the model is proportional to the square of the number of hidden states. In scenarios where there are multiple overlapping processes, the state space of the hidden variable grows exponentially with the number of processes. This in turn increases the required amount of training data exponentially, which is a major problem in many domains. Extensions have been studied which mitigate the problem in certain special cases [11,60], but cannot be applied in general to cases with large numbers of overlapping events.

Conditional Random Fields (CRFs) can also be created with a structure that mimicks HMMs and in such cases [43], have been shown to outperform HMMs. One example of such a comparison in the event recognition domain is the work of Vail *et al*. [78], which applies HMMs and linear chain CRFs to the toy problem of identifying which of three robots involved in a game of 'robotic tag' is doing the chasing based on the position and motion of the three robots. Here CRFs are shown to outperform HMMs but the problem is simplistic (there is one state variable which can take one of three discrete values). Sminchisescu *et al*. [71] exploit more of the power of the CRF versus the HMM by exploring longer-range dependencies between state variables and observations but again, the dataset is small and simple. The problems described in the previous paragraph with respect to implementing such state-based methods in a complex domain are not solved or mitigated in any significant way by the switch to linear chain CRFs.

2.2.2 Grammars and Logic

Many existing approaches to high-level event recognition break complex activities down into a sequence of simpler atomic activities then exploit the temporal structure between these activities. In the work of Ivanov and Bobick [35], this structure takes the form of a Stochastic Context Free Grammar (SCFG). They describe the application of their system in two domains, the first of which focuses on recognition of complex gestures, with a SCFG used to associate probabilistic detections of simple gestures generated by independent HMMs. The second domain is more relevant to this work, focusing on multi-agent event recognition in CCTV footage over a carpark. In this case, a motion tracker is first applied to the video sequence before a hand-crafted 'trajectory event generator' creates detections with probabilities attached. These trajectory events consist of events such as 'car-enter' or 'person-leave' to represent people or cars entering or leaving the scene at commonly-used entry and exit points. If the tracker loses people or cars elsewhere in the scene, 'car-lost' or 'person-lost' events are fired. Probabilities may be attached to these events; for example the tracker may have difficulty classifying an object so might fire 'person-enter' and 'car-enter' with equal probability. In SCFG, the atoms which are observable and cannot be reduced into simpler components are referred to as *terminals*. A section of the SCFG used for interpretation on top of these terminals is shown here in Figure 2.6. It can be seen that the event vocabulary is relatively small and simple, essentially just distinguishing between cars dropping people off, picking people up or just driving through the scene yet the grammar is quite verbose as intermediate states are required to cope with noise. This highlights one of the main problems with this approach: the SCFG, including the associated rule weights must be created manually, meaning a lot of work is required to tailor the method to each domain. Additionally, there is no easy way to exploit expectations over time differences between events.

Damen and Hogg [18] tackle a problem of similar complexity. The domain of interest in this work is CCTV footage of bicycle racks, where the goal is to match instances

G_p :

```
TRACK          →     CAR-TRACK                                          [0.5]
                   |  PERSON-TRACK                                       [0.5]
CAR-TRACK      →     CAR-THROUGH                                        [0.25]
                   |  CAR-PICKUP                                        [0.25]
                   |  CAR-OUT                                           [0.25]
                   |  CAR-DROP                                          [0.25]
CAR-PICKUP     →     ENTER-CAR-B CAR-STOP PERSON-LOST B-CAR-EXIT        [1.0]
ENTER-CAR-B    →     CAR-ENTER                                          [0.5]
                   |  CAR-ENTER CAR-HIDDEN                              [0.5]
CAR-HIDDEN     →     CAR-LOST CAR-FOUND                                 [0.5]
                   |  CAR-LOST CAR-FOUND CAR-HIDDEN                     [0.5]
B-CAR-EXIT     →     CAR-EXIT                                           [0.5]
                   |  CAR-HIDDEN CAR-EXIT                              [0.5]
CAR-EXIT       →     car-exit                                           [0.7]
                   |  SKIP car-exit                                     [0.3]
CAR-LOST       →     car-lost                                           [0.7]
                   |  SKIP car-lost                                     [0.3]
CAR-STOP       →     car-stop                                           [0.7]
                   |  SKIP car-stop                                     [0.3]
PERSON-LOST    →     person-lost                                        [0.7]
                   |  SKIP person-lost                                  [0.3]
```

Figure 2.6: Part of the Stochastic Context Free Grammar used in [35]

of people dropping off cycles to instances of people collecting cycles. A Hiearchical Bayesian network is instantiated whose size is dependent on the output of a motion tracker for a sequence. Each one of n trajectories is assigned a node. A further m nodes are created to represent changes in bicycle clusters in the scene. $n \times m$ mid-level nodes are created to represent all the potential pairings of person trajectories with cycle cluster events. A third level in the vocabulary has variables to link all pairs of mid-level events into compound 'drop-pick' events, meaning a further $\binom{n \times m}{2}$ nodes. The topology of the DBN makes exact inference intractable therefore MCMC is used to find the optimal labelling of latent variables. Promising performance is shown, and the amount of domain-tailoring for this kind of strategy seems to be slightly less than in the SCFG case. However, the size of the search space and intractability of the inference could make the methodology costly to apply in complex domains.

Inductive Logic Programming (ILP) is another technique that has been applied to the event recognition problem. In ILP, the logical rules that best explain a set of training data are learnt from background knowledge and labelled examples. Needham *et al*. [55] demonstrate a system that is capable of learning the rules to several simple card games by observing short passages of play. In these games however, there are no complex temporal relations between events. The system's only awareness of time is through a predicate named 'successor' which indicates when one event follows another, meaning that extensions would be required for application in more complex domains. Dubba *et al*. [21] apply ILP to the task of event learning in the aircraft servicing domain. In that work, relational data is created from motion tracking by abstraction to spatial relations between pairs of objects, with temporal relations between interactions modelled through Allen's interval calculus [2]. Positive training examples are specified through 'Diectic Supervision' , which involves supplying the spatial and temporal extent of events within the video volume. ILP is then used to find the ruleset that best explains most of the positive examples whilst minimizing the number of false positives induced by the rule. Sridhar *et al*. [72] similarly abstract from tracking to relational data, but in this approach the learning is almost entirely unsupervised. A video sequence is represented as a graph which links tracks to *episodes* and links episodes with temporal relations. An episode is defined as an interval of time over which a spatial relation between two objects holds. The primitive relations are linked through a graph structure.

Note that the research of Dubba *et al*. was also carried out on the Co-Friend dataset, which is the dataset of primary focus in this thesis and it is therefore a relevant benchmark.

2.2.3 Motion Patterns and Topic Models

A sub-branch of activity recognition is work on learning of *Motion Patterns*. Motion Patterns are typically described as spatial segments of the image that have a high degree of local similarity of speed and flow direction within the segment and otherwise outside [66].

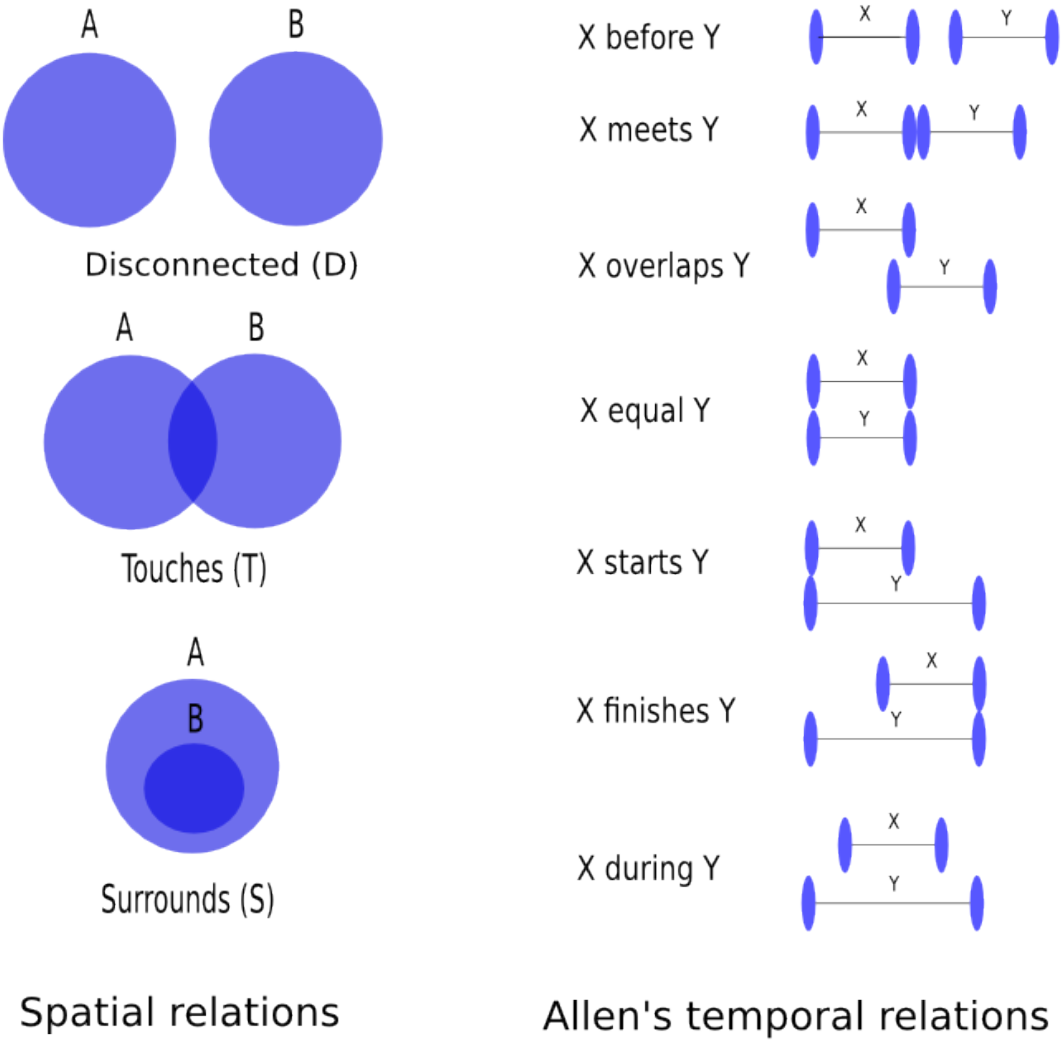


Figure 2.7: Spatial and temporal relations used in Dubba *et al.* [21].

Some of they key motivations behind trajectory-based event recognition is provided by Johnson and Hogg [36]. In this early work, trajectories obtained from a tracker are broken into sequences of motion vectors, which are quantised so that they might be used as a means to detect typical and abnormal trajectories or events. This work is extended by Stauffer and Grimson [74], who similarly cluster motion vectors then represent video sequences as multisets over these words. They then perform hierarchical clustering using co-occurrence statistics. The clusters produced are shown to correspond to meaningful

categories such as morning/evening rush hour, pedestrian traffic etc.

The later work of Hu *et al*. [34] follows a similar methodology, only the motion patterns are represented by chains of Gaussian distributions, which they argue is a more intuitive statistical formulation that thus allows anomoly detection through the application of probabilistic inference.

The limitations of trajectory-based approaches however are summarised in [88]. The chief questions when considering the suitability of a trajectory-based approach to event detection for a particular domain are whether sufficiently reliable tracking is feasible and whether sufficient information about the events of interest is encoded in the trajectories alone. For many domains, including the specific case of detecting servicing events on an airport apron, neither of these criteria can be said to be met by the current state of the art. This comment is further justified in Chapter 3.

The methodologies so far discussed in this section employ motion trackers to obtain trajectories, however this is not the case for all research in the area. Recently Saleemi *et al*. [66] described a method which uses dense optical flow as input to their learning. They first chunk a video into one second-long clips, extract features $X = (x, y, \rho, \theta)$, where (x, y) is position, ρ is magnitude and θ is direction of flow. They 'marginalize' time in each clip then perform k-means clustering on the 4D data collected from many clips and use the resulting labelled data to train a Gaussian mixture model. A clip is then represented as N Gaussian mixture components. A longer video is treated as an undirected graph with the nodes being mixture components. Nodes are connected if they belong to proximal clips and one component is 'reachable' from another if the parameters of the second Gaussian are close to what can be predicted based on a linear prediction from the position and motion of the first component. Connected components of the graph represent distinct motion pattern instances that occurred through the video. KL divergence is then computed for all GMMs by Monte Carlo sampling and GMMs are thus grouped. The approach is tested on three datasets; one is 15 minute-long traffic scene, the second is the MIT traffic

dataset used in [83] which consists of 90 minutes of traffic data, and a final short video sequence consisting of just 250 frames of a scene containing dense pedestrian movement. The results are not quantified in the paper, but the conjecture is that they do correspond to patterns that would be meaningful to a human observer.

The methodology of Stauffer and Grimson discussed earlier closely resembles Topic Models, which have been used extensively in the information retrieval literature [9]. Several recent works have used them explicitly for human action recognition.

Niebles, Wang and Fei Fei [56] is one notable example. They use Dollar's feature detector and gradient descriptors calculated from a cuboid exactly as in [20] and k-means is once again applied to get a finite vocabulary. They then apply two different latent topic models borrowed from document analysis- probabilstic Latent Semantic Analysis (pLSA) and Latent Dirichlet Allocation (LDA) to learn action categories in an unsupervised fashion. These two models are presented diagramatically in Figure 2.8. One advantage of these methods is that localization is easier with this kind of model than SVM as it is easy to identify the spatio-temporal words which are most strongly correlated with a particular action category. A weakness of the method is the strong assumption that all words are independent of one another given the topic and similarly that all topics are independent given the document. They show close to state of the art results on the KTH dataset; though unfortunately for admirers of Bayesian methods they find the simpler pLSA method to do better empirically. They go on to evaluate performance when localizing multiple actions in a single clip, which is simply done by taking all action categories 'significantly induced' (thresholded under certain restriction) by $P(z|w,d)$ - the probability of topic,z, given words, w, and document, d. k-means then performed (with k = number of topics discovered) to cluster interest points based on position before the clusters are assigned to actions based on voting from features within. Another task they tackle, not covered by any of the aforementioned papers, is that of finding several actions performed by the same actor within one video clip, with what is essentially a sliding window detector.

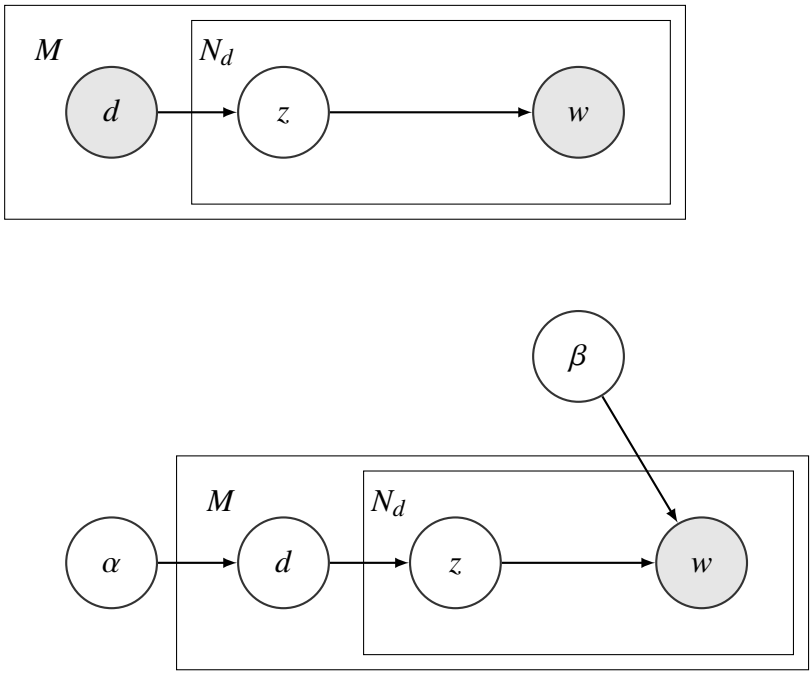


Figure 2.8: Top: Graphical representation of probabilistic Latent Semantic Analysis model. d represents a document, z is a (latent) topic and w is a word.M is the total number of documents and N_d is the number of topics in a given document. Bottom: Latent Dirichlet Allocation model. All common variable names take the same meaning, but note that in LDA, d is not observed, rather a Dirichlet prior is placed on it, with hyperparameter α. An additional multinomial prior β is placed on w.

Wang *et al*. [83] also implement a variant of LDA, before performing further experimentation with Hierarchical Dirichlet Processes (HDPs). The HDP model is completely parameter free since the Dirichlet Process prior does not even require that the number of topics is specified. This is particularly important when applying the method in an unsupervised fashion. An extra layer to the model also exists when compared to the work of Niebles *et al*., meaning that rather than just latent topics, there are latent *atomic activities* and *interactions* (which are a level higher up the hierarchy). The validity of the method is shown by experimentation on a synthetic dataset. A topic model is specified and used to generate a bag of images. HDP then successfully recovers the parameters of the model from the data without any supervision. There are a series of experiments then done on a traffic dataset, with lots of effort made to prove that the topics extracted by the LDA and HDP are semantically meaningful.

As demonstrated, the works in this area are generally focused on the task of unsupervised learning of patterns which doesn't quite fit the context of this thesis where one wishes to detect a set of events from a fixed vocabulary within structured sequences. However, some of the events in the *cofriend* dataset are so semantically simple that they can essentially be considered to be a particular vehicle moving along a particular trajectory, where absolute position is important. For these kinds of events, approaches based on patterns of motion might be easily applicable and more appropriate than the action recognition methods; which are geared more towards recognizing the articulated motion of humans.

2.2.4 Part-based models

The primary focus of this thesis is the recognition of events in structured scenarios. In this setting, an event can actually be considered to be a constituent part of a larger scenario. As mentioned earlier, the local-feature based approach to action recognition drew inspiration from the object detection literature which had already shown similar methods to be

effective in that domain. One of the most active areas of research in object detection is that of part-based models. The Pictorial Structure Model (PSM) was introduced by Fischler and Elschlager in 1973 [29]. Their idea is that an object can be represented as a set of rigid components held together by 'springs'. The springs joining components constrain the relative movement by incurring a cost depending on how much they 'stretched'. They formulate the following objective function for object localization:

$$G(X) = \sum_{i=1}^{p} \sum_{j=1}^{p} g_{ij}(x_i, x_j) \tag{2.12}$$

Where p is the number of parts in the model, $X = x_1, .., x_p$ is a vector giving the position of each part within the image. When $i \neq j$, $g_{ij}(x_i, x_j)$ represents the spring loss incurred under the colocation of parts i and j. In cases where this relationship is uninformative, this would simply be zero. When $i = j$, $g_{ij}(x_i, x_j)$ is defined to be a local appearance term which measures how well a particular part matches the image at a given location. They define a dynamic programming procedure for finding an approximate solution to the localization problem. Many later studies built on this work, most notably Felzenszwalb and Huttenlocher in [27], which introduces a probabilistic formulation for the PSM and thus suggests some Bayesian methods for efficient training and inference. This formulation will be studied in more detail in Chapter 4 where I draw parallels with my model for scenario recognition.

Kumar *et al*. extend the Pictoral Structure Model in [42]. Their approach involves using an appearance model for each part which consists of two elements; shape and texture. The training involves building a set of exemplars for shape and a set of two-component GMMs for texture (which makes sense given the target domain is cows). The likelihood of a given part being in a particular patch is taken to be Gaussian over the minimum truncated chamfer distance between the edge outline and all the exemplars for that part, multiplied by the texture likelihood obtained by reference to the learnt GMMs. For implementation

efficiency, the texture is only referenced at candidate locations deemed sufficiently likely with respect to the outline.

Unlike in Felzenszwalb's formalism, the structural prior is a fully connected graph structure. Their experiments show that this improves performance versus a tree structured prior. Another key change they make to the PSM which is not emphasized in the paper is that the pairwise likelihood function is just a 'top-hat' function, meaning zero in valid configurations and a constant value otherwise. Experimetation with fully connected models was carried out in this thesis and Chapter 5 gives an idea of why this change was most likely made. The experiments highlight that if a PSM is treated as being pairwise fully connected, as the number of pairwise relations grows quadratically with the number of parts, then these terms tend to dominate the observation likelihood, since many of the pairwise terms are heavily dependent on one another yet by multiplying them together independence is being assumed.

CRFs are another powerful method which have been used to implement part-based models. Detail on the theory and implementation of CRFs is included in Chapter 5 but for now the focus is on the application. Quattoni *et al*. [62] extract m patches, $\mathbf{x} = \langle x_1, ...x_m \rangle$, from an image with a SIFT detector. They connect these patches with a minimum spanning tree (with distance function between patches just being the Euclidean distance), stored as the edge set E . Each patch has a latent (training data is not labelled at the part level) variable which gives its assignment to an object part, $\mathbf{h} = \langle h_1, .., h_m \rangle$. They then define the following conditional model:

$$P(y, \mathbf{h}|\mathbf{x}, \theta) = \frac{e^{\Psi(y,\mathbf{h},\mathbf{x},\theta)}}{\sum_{y',\mathbf{h}} e^{\Psi(y',\mathbf{h},\mathbf{x},\theta)}}. \tag{2.13}$$

Where

$$\Psi(y, \mathbf{h}, \mathbf{x}, \theta) = \sum_j \phi(x_j) \cdot \theta(h_j) + \sum_j \theta(y, h_j) + \sum_{(j,k)\in E} \theta(y, h_j, h_k) \tag{2.14}$$

Where $\phi(x_j)$ is a feature vector consisting of SIFT and relative location and scale features.

The inner product $\phi(x_j) \cdot \theta(h_j)$ can be interpreted as a measure of compatibility between patch x_j and part label h_j. Each parameter $\theta(y,k)$ can be thought of as a measure of compatibility between a part of type k and label y. Each parameter $\theta(y,k,l)$ measures the compatibility between an edge with labels k, l and overall label y. Inference in the model is performed via Belief Propagation. Experiments focus primarily on the task of detecting various types of vehicle in a standard dataset. Results shown are impressive and there is some indication that the parts learned in the model correspond to meaningful entities such as the wheels of the car. Wang and Mori [84, 85] have applied similar methodology to action recognition, working with the 4-channel person-centric flow fields proposed by Efros *et al*. rather than images.

Another novel part-based approach to action recognition is proposed by Ke, Sukthankar and Herbert [39]. In this, videos are treated as 3D space time volumes. An action is simply represented as a shape within this volume. Recognition of an action is performed by first segmenting a video into *super voxels* then sliding the template for the action throughout the video volume and finding the local maxima of a normalized response function. The response function is calculated as the intersection distance between the template and the optimal subset of supervoxels that are entirely or partially contained within the template. The template for an action is generated from a single example. The template is manually segmented into parts and the model allows the parts to move independently to improve the generalization power of the template. The response function for a model of multiple parts is then a combination of the response of each individual part and the geometric matching score between the parts.

A significantly different part based approach is outlined by Niebles *et al*. [57] in which temporal structure is combined with local features to recognise more complicated events. Here, a single event model consists of bag of word classifiers at different temporal scales, with some additional temporal context for each classifier. The model is learnt with a Latent SVM formulation and an iterative learning algorithm which alternates between

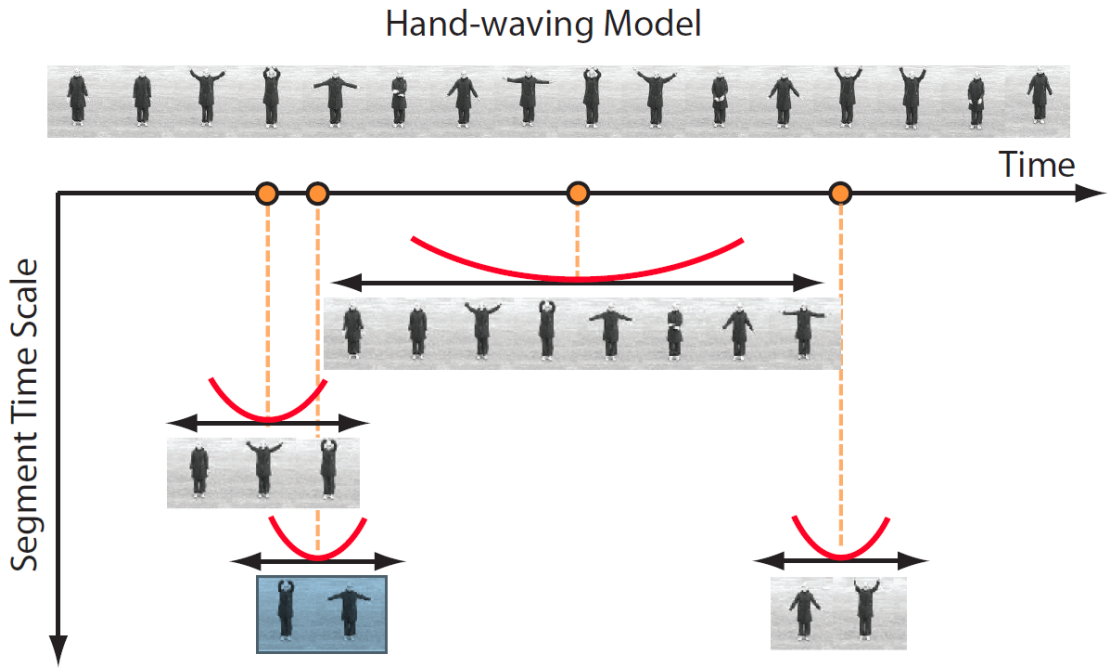


Figure 2.9: Visualization of one of the learnt models from Niebles *et al*. [57]. In this illustration, the horizontal axis represents time. Each row corresponds to a motion segment classifier learned by the model, whose temporal extent is indicated by its vertical location. The appearance of the motion segment is illustrated by a few example frames. The associated dot indicates the anchor position of the motion segment relative to the full sequence. The parameters of the temporal misplacement penalty are represented by the parabola centered at the anchor point.

optimising model parameters and estimating the hidden variables (positions of the motion segment classifiers). Their model is easily understood visually in Figure 2.9.

An alternative approach is presented in [50], where pairwise temporal relationships are appended to local features prior to quantization with the goal of retaining some of the temporal information (intraevent) that would otherwise be lost in a local feature representation.

2.3 Discussion

I have surveyed a broad range of techniques for activity recognition in video. Whilst there exist many powerful methods that have achieved impressive results across various datasets, it is clear that the problem of human action recognition in general unconstrained scenarios is still very much unsolved. No one method exists that beats all others across all datasets; it seems rather that the various methods have differing degrees of robustness to the many different factors that make the problem of activity recognition so challenging:

- occlusion,
- subtlety and variability of motion,
- scale variation,
- viewing angle variation,
- camera motion.

Therefore the 'best' activity recognition system for a given scenario is the one which copes best with those difficulties which are most inherent to that domain. The thrust of this thesis is to create a model for high level scenario recognition which is independent of the workings of the lower level detectors, to allow appropriate ones to be selected for the task in hand: perhaps combining heterogeneous detectors to recognize different events within the same domain. Particularly important in this work is the idea of efficiently exploiting temporal relations between events. None of the previous works in the area are quite satisfactory on this count, either treating the temporal domain in the same way as the spatial domain, or else focusing purely on the ordering rather than expectations over relative timings of events.

Chapter 3

Implementing Low-Level Event Detectors

The primary dataset of concern in this thesis is the Co-Friend dataset. Two important characteristics of this dataset in vision terms are that the camera position is fixed and the position at which events can occur within the scene is very constrained, meaning there is little variation in the scale at which events are observed, and there are no complications caused by camera motion. Occlusion is on the other hand a major issue, since several events are often occurring simultaneously in close proximity, involving multiple protagonists. Some events take place a long way from the camera, meaning the people involved may only be 20 pixels high. These factors render the task of tracking exceedingly difficult. A state-of-the-art tracker [76] was applied to the dataset in early experiments but numerous issues were observed. The most critical of these issues was the inability to maintain object IDs in cluttered scenes. The research of Dubba *et al*. [21], was carried out on this tracking data, however they were only able to report results on a small subset of the

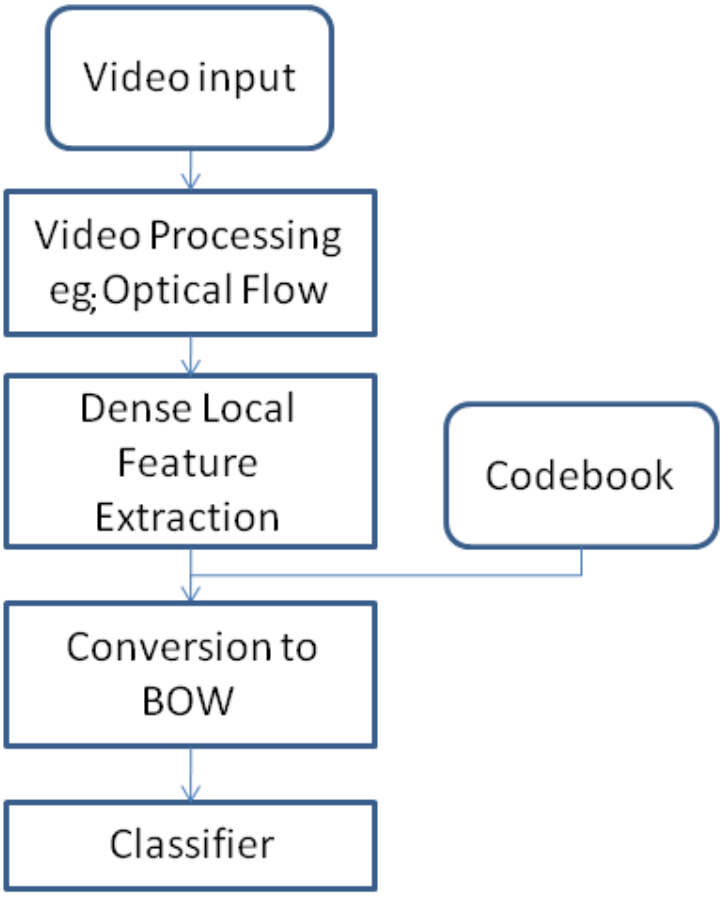


Figure 3.1: Flow diagrams describing the action recognition pipeline implemented in this chapter.

event vocabulary. If a high level event recognition system is to depend on tracking alone, the tracking needs to be stable and capture most or ideally all relevant activity. Whilst noise and missing data can be mitigated to some extent with filters or smoothing, these are things that are would more naturally sit within the tracker, where visual information is still available. As tracking was not the focus of this thesis and existing trackers did not give satisfactory performance on the target domain, alternative methods which did not rely on tracking were preferred for event detection. In the previous chapter, a strong focus was placed on methods using classifiers with local features as detectors. This is the type of method chosen to suit the Co-Friend project.

3.1 Training Event Detectors Based on Histograms of Optical Flow

This chapter describes the implementation of a detection framework based closely on the work of Laptev *et al*. [47]. The detector was implemented from scratch to ensure that all the design choices that would go into the framework were understood. The process did indeed prove to be informative, and whilst Laptev's method is clearly described and thoroughly evaluated, there were some important aspects of the HOF features which could perhaps be enhanced with some further tuning. This chapter begins with a brief description of the method and ends with a thorough evaluation of several HOF parameters which have previously not been tested. Most of these things relate to the normalization scheme for HOF features, which was highlighted by Dalal *et al*. [17] to be an important aspect of the HOG feature. At the time of writing there are no published works which perform this evaluation for the HOF feature.

The detection framework is implemented in Python since the high level functionality and wide range of interfaces to standard libraries allow rapid development. For efficiency, as far as is possible the implementation leverages methods of OpenCV, which has been highly optimized in C++ and 'numpy' [59], which is essentially an interface to the BLAS [7] and LAPACK [3] libraries. BLAS and LAPACK are highly tuned libraries for performing array, vector and linear algebra operations.[1]

3.1.1 HOF Feature Considerations

The first stage in dense HOF/HOG feature extraction for video analysis is to transform the image sequence into a series of 2-channel optical flow or gradient images. This is the only inherent difference between HOG [17] and HOF features, though as already mentioned there are small differences in the typical implementation of the two. In the case

[1] The framework will be made available open-source alongside this thesis.

of HOF, dense optical flow is computed, giving a 2-channel image where one channel is the horizontal motion and the second channel is vertical motion. This 2 channel image is then converted to a D channel image, where D is the number of directions into which motion is to be quantized. In Laptev's original implementation, an additional bin is designated for static pixels and the binning is done in a coarse way where each pixel places a single vote of unit weight into the bin whose direction is closest to the direction of the vector described by the optical flow at that point, with motion below a certain threshold resulting in a vote for the non-motion bin. The argument for this methodology is that the direction of motion alone should provide enough information for action recognition and that optical flow algorithms can be noisy. It is feasible that a coarse method may be more robust to noise but the additional bin required for this approach does seem inelegant and there is no precedent for such a construction in the more mature HOG literature. In this chapter, this coarse binning approach is compared with a bilinear filtering approach which is described as follows. First convert the flow from dx, dy representation to magnitude, m and orientation, θ. Each of the D directional bins is defined by an orientation value ϕ_i. In general, the orientation of any flow vector will fall $\phi_i < \theta \leq \phi_j$. The magnitude of this vector is then shared between bins i and j such that $m_i = \frac{m(\theta-\phi_i)}{\phi_j-\phi_i}$ and $m_j = \frac{m(\phi_j-\theta)}{\phi_j-\phi_i}$. Note that in practice this can be computed efficiently by reference to look-up tables. Once a number of frames have been converted into D channel images, dense HOF features are extracted from this volume. See Figure 3.2 for a diagram of a HOG/HOF feature which clarifies several elements of the terminology that will be used. A HOG/HOF feature corresponds to a cuboid which is termed a block. A block is itself composed of potentially overlapping smaller cuboids referred to as cells. These cells define the regions over which gradient/flow vectors are to be accumulated into histograms over direction. The histograms from all cells within a block are concatenated to produce one feature vector. This vector is of dimensionality $x_b \times y_b \times t_b \times \theta$. The parameters relating to the features are:

- x_b, y_b, t_b are the dimensions of the block (in cells);
- θ is the angular granularity i.e. the number of bins that direction is discretized into;
- O_c is the degree of overlap between cells;
- Z is the normalization scheme (l^1, l^2, etc..) .

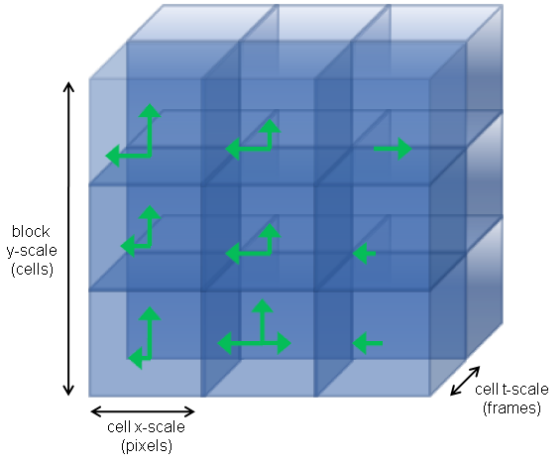


Figure 3.2: Diagram of a HOF feature with a block size of $3 \times 3 \times 2$, with an angular granularity of 4 (i.e. 4 different bins for direction of motion)

Clearly a grid search over this number of parameters would be costly on any non-trivial dataset. Laptev *et al*. [47] perform an evaluation over various settings for most of these parameters on the Hollywood dataset, referring to each of the 24 different settings tried as a *channel* and displaying the most effective channels and combinations of channels. There is no one channel which emerges as clear winner, with different channels proving slightly more effective for different event types. However, in the later experimental paper of Wang *et al*. [82], which compares the effectiveness of various feature point detectors (Harris 3D vs Hessian vs cuboid vs dense sampling) and feature types (HOF vs HOG vs cuboid), the block dimensions are held fixed at $3 \times 3 \times 2$, without any cell overlap, and angular granularity of 4, suggesting that this has been selected as a good general-purpose choice. Note that the cell dimensions x_c, y_c, t_c which were mentioned earlier were omitted from the list of parameters. This is because they are considered to define

the scale of a feature. Since feature point detectors generally have a method for approximating scale and dense sampling is performed over multiple scales, it is not necessary to optimize these parameters prior to deployment of the system.

To enable easy comparison with previous work, and keep the number of different permutations down to a manageable level, all experiments in this chapter are performed with block dimensions fixed at $3 \times 3 \times 2$. The focus in this chapter is to compare normalization strategies. In Laptev's implementation features are l^1 normalized, which due to the unit voting scheme used is equivalent to simply dividing through by the number of pixels in a block. Adopting a bilnear filtering approach for the binning of motion however opens more possibilities for the normalization of the HOF feature vectors. Three normalization schemes are compared: the well known l^1 and l^2 and then a novel scheme we term scale-normalization which is described in Figure 3.3. The normalization constant in scale-normalization is purely dependent on the scale of the feature, preserving rich magnitude information whilst remaining invariant to scale changes. This could potentially prove very useful in distinguishing between events that involve for example, a vehicle stopping suddenly as opposed to a vehicle stopping gradually.

3.1.2 Evaluation of normalization and motion-thresholding

All normalization methods with bilinear versus coarse binning for 3 different angular granularities on the challenging Hollywood 2 dataset are evaluated, using the train/ test divisions provided by the authors. The one stochastic element in the training algorithm is the random initialization of the k-means algorithm used to build the codebook. This problem is mitigated by reinitializing 20 times and taking the best scoring codebook (in terms of information loss over the data used for generation) and thus observe a standard deviation in performance figures $< 0.5\%$ due to codebook initialization.

The results of these experiments are listed in Table 3.2. It seems clear from these results that an angular granularity of 4 is optimal for this dataset; presumably because use

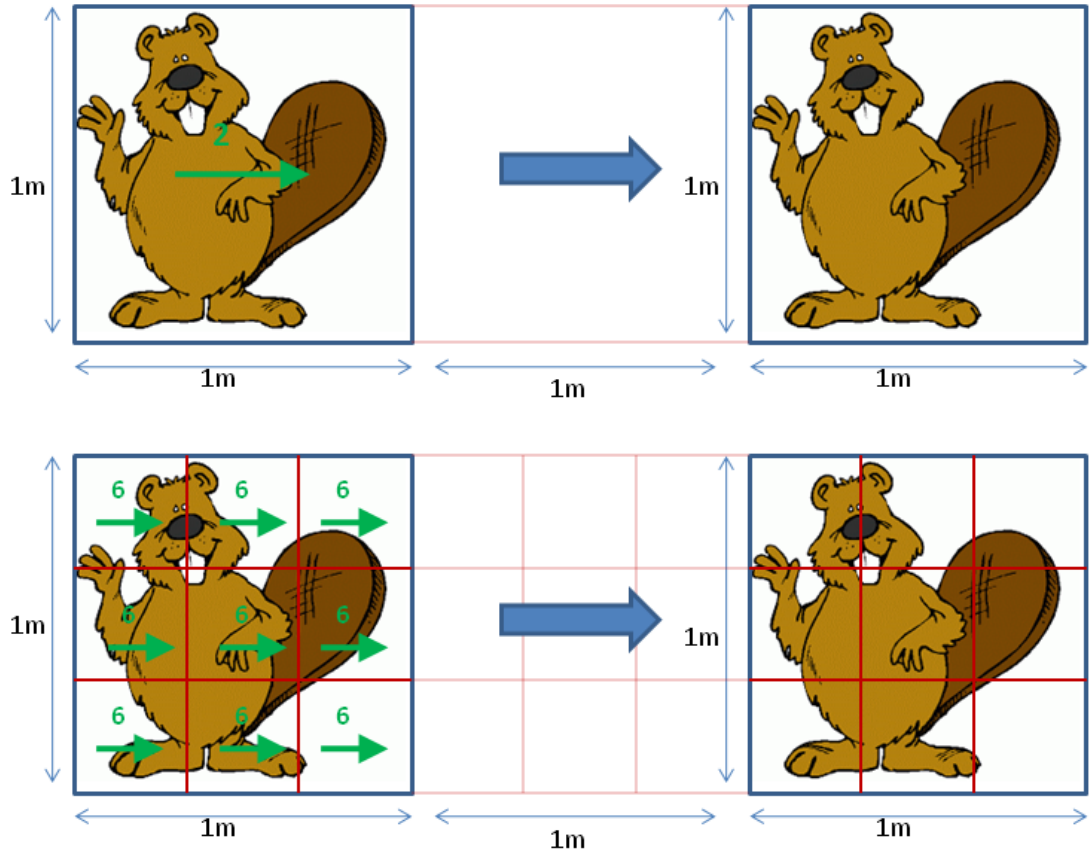


Figure 3.3: Demonstration of feature normalization based only on feature scale. Both rows above show a metre-high, metre-wide bever undergoing a pure translational motion of 2 metres between frames. The top row shows the case where the beaver is captured only by a single pixel. Its translational movement of 2m thus registers an optical flow value of 2 along the x-axis. The bottom row shows the beaver at 3× scale. Now, the beaver is still contained within a single cell but there are $3^2 = 9$ pixels within that cell. Additionally, the optical flow magnitude for each pixel within the cell has increased in proportion to the scale change. It is therefore that to keep scale independance without block normalization, it is necessary to first divide the flow vectors by the scale factor before dividing by the number of pixels in the cell (in this case 3^2).

of only 2 directional bins results in features which are not discriminative enough whilst 6 bins makes features less robust to variations in viewing example for example. Also evident is that bilinear binning of flow magnitudes outperforms coarse binning in almost all cases. l^2 normalization appears to be the most successful normalization strategy, whilst the scale normalization strategy is the biggest disappointment; yielding the worst results

of all.

There are several reasons which could potentially contribute to this result. Firstly, motion magnitude isn't very helpful in distinguishing between the action types covered by the Hollywood dataset. Secondly, not normalizing causes points to be more sparsely spread in the original feature space, causing problems for the k-means codebook generation and making the features less repeatable. Finally, the optical flow algorithm deployed uses a Gaussian weighting term in the region-segmentation procedure used to smooth flow. The scale of the Gaussian can be specified but since dense optical flow calculations are expensive it is only feasible to run this once per image, subsequently extracting at multiple scales from the same flow image. Because of this it could be that without normalization of flow magnitude, the repeatability at different scales might be reduced. This issue was investigated by extracting HOF features with scale normalization over one of the longer clips (2265 frames) from the Hollywood 2 dataset, then upscaling the video to 2× scale and running the feature extractor again with features at 2× scale, using the same codebook. The raw flow obtained for various frames in this clip at these two scales is shown in Figure 3.4. This anecdotal evidence appears to show that significant salient motion is almost identically captured at both scales, though there is difference in the patchier background noise. As a small quantitive verification of this intuition, the features extracted at these two different scales are examined to check how many are identically quantized across different normalization schemes. The figures, given in Table 3.1 appear to validate the concern that the scale-normalized features are less repeatable over different scales than the normalized varieties.

3.1.3 Statistical Significance of Hollywood Performance Figures

The performance figures quoted for these detectors come from a test set comprising 884 samples. The best performing parameter combination (using bilinear binning and l^2 norm) classifies 414 cases correctly (46.8%), whilst the next best combination (coarse binning

Norm	Nwords different	Percentage different
l^2	8080	32
$Scale$	8635	35
$Coarse$	7990	31

Table 3.1: Repeatability of HOF features at 2 different scales in a single clip from the Hollywood 2 dataset. The second scale is obtained by upscaling the original image sequence to double the original size using bilinear interpolation, then re-running optical flow calculations with identical parameters and extracting features at double the spatial scale.

Normalization	Angular Granularity	Binning	Mean Average Precision (%)
l^1	2	bilinear	43.1
l^1	4	bilinear	46.0
l^1	6	bilinear	43.8
l^1	2	coarse	41.5
l^1	4	coarse	45.9
l^1	6	coarse	42.9
l^2	2	bilinear	43.8
l^2	4	bilinear	**46.8**
l^2	6	bilinear	44.8
l^2	2	coarse	42.5
l^2	4	coarse	45.2
l^2	6	coarse	44.8
scale	2	bilinear	41.0
scale	4	bilinear	40.8
scale	6	bilinear	39.9

Table 3.2: Performance figures for parameter combinations on Hollywood2 dataset. The Mean Average Precision is obtained by taking the mean AP across event classes

	NOccurs	l^2 bilinear	l^1 coarse
pole vault	40	62.1	60.2
vault	56	60.0	60.4
clean and jerk	66	76.4	75.3
shot put	63	25.2	27.1
diving springboard 3m	46	92.9	84.3
long jump	46	77.4	77.1
snatch	49	44.2	44.2
basketball layup	50	63.2	60.8
high jump	67	57.1	54.0
javelin throw	25	78.0	77.4
bowling	49	36.2	37.8
diving platform 10m	57	95.1	88.8
discus throw	63	39.9	39.5
hammer throw	46	40.7	39.6
triple jump	21	17.0	20.1
tennis serve	39	37.2	37.0
Overall	783	57.2	55.8

Table 3.3: Performance figures for parameter combinations on the Stanford Olypic Sports dataset. The Mean Average Precision (%) is obtained by taking the Mean Average Precision across event classes

Figure 3.4: Side by side comparison of thresholded dense optical flow extracted with the Farneback OpenCV implementation with identical parameters at original (left) and 2× scale (right). Arrows are plotted proportional in length to the magnitude of flow every 8 pixels in x and y directions at original scale and every 16 pixels at the double scale. Only flow with a euclidean magnitude greater than 2.5 (original scale) and 5 (double scale) is displayed.

with l^1 norm) classifies 406 cases correctly (45.9%). It is not obvious however, whether this 0.9% improvement is conclusive evidence that bilinear binning with l^2 norm is better in general given the size of the test set? One way to model the significance of these figures is with the binomial theorem. The true performance values p_1, p_2 of the two detectors are unknown. This gives a distribution over p_i in the form:

$$p(p_i) = \frac{B(p_i, n, k_i)}{\int_0^1 B(p_j, n, k_i) \mathrm{d}p_j} \tag{3.1}$$

Where B is the binomial distribution

$$B(p,n,k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{3.2}$$

Where k_i is the number of samples correctly classified by classifier i and n is the total number of samples. Therefore the likelihood that the true performance $p_1 > p_2$ is

$$\iint_D p(p_1)p(p_2)\mathrm{d}p_1\mathrm{d}p_2 \qquad D = \{(p_1,p_2) \in \mathbb{R}^2; 0 \geq p_1 \geq 1; p_1 \geq p_2 \geq 1\}, \tag{3.3}$$

which is the region below the line in the graph shown in Figure 3.5. In this case, the probability mass in that region is 0.65 meaning that it is with only 65% certainty that classifier p_1 is better than p_2. Therefore it is not possible to make the statement that bi-linear filtering is superior with confidence. Hence corroboration is needed from a further dataset. The best scoring two normalization schemes on the Hollywood dataset are taken forward for further experimentation on the Stanford Olympic Sports dataset [57]. In the publication which introduces the dataset, it is stated that there are 50 examples for each of the 16 classes of action and that training is performed on 40 samples, testing on 10. A train/test split is supplied with the dataset but within this file, it was found that there were in many cases with fewer samples than expected. For one of the classes there were as few as 4 samples in the test set, meaning any figures produced would be of questionable importance. Therefore the fixed test/train division supplied by Niebles *et al*. was abandoned and a 5-fold cross validation procedure adopted, which explains the difference in performance in the results displayed in Table 3.3 relative to the baseline given in [57]. The bilinear binning with l^2 norm once again performs best overall, classifying 448 cases correctly (47.2%), whilst the coarse binning with l^1 norm classifies 437 cases correctly (55.8%). When combined with the numbers from the Hollywood Dataset, this gives 862 for bilinear l^2 against 843 for coarse l^1 from 1667. Evaluating the integral in Equation 3.3 using these numbers now gives 0.75, meaning it is possible to say with 75% certainty that

bilinear filtering with l^2 norm is more effective than coarse binning. Further experiments would be required to increase the significance further but time constraints and lack of further easily accessible datasets prohibited further work.

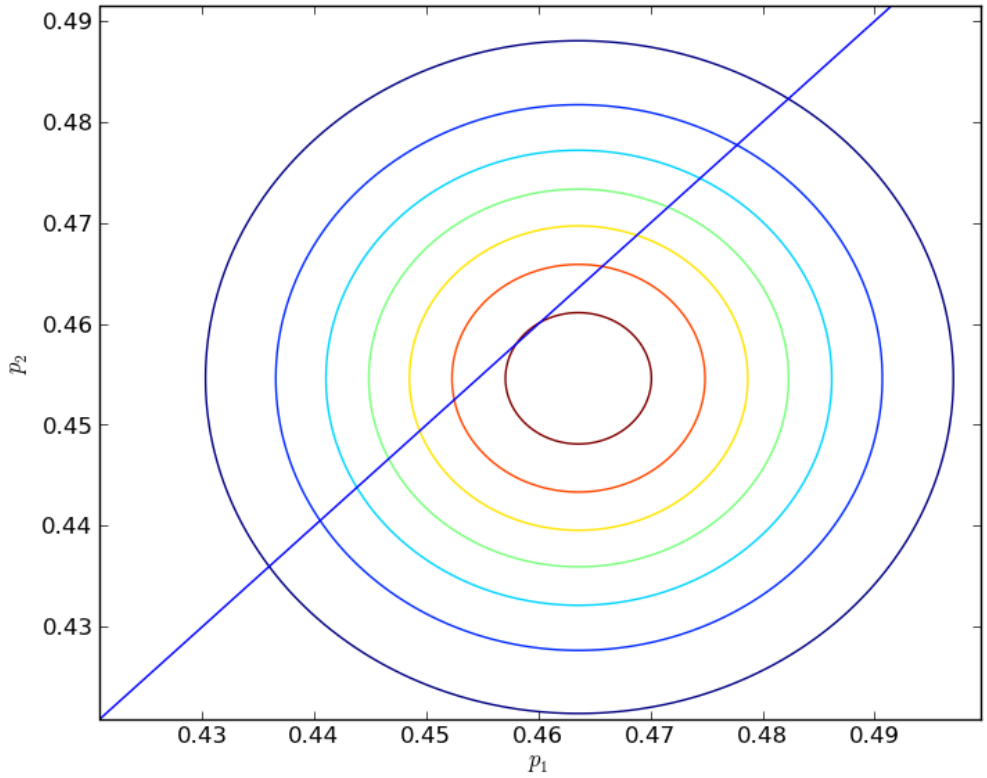


Figure 3.5: Contour plot of the joint probability distribution over the true performance level of two classifiers based on their success on the Hollywood dataset.

3.2 The Co-Friend Dataset

The dataset consists of a collection of 37 recordings of aircraft servicing turnarounds on one apron at Toulouse airport. There is video from between 3 and 5 static cameras for each sequence. Some of the camera angles suffer heavy occlusion once the aircraft is in position. Most servicing activities occur on one side of the aircraft hence the cameras on

the other side are redundant in recognition of these activities. There is one camera position just to the left of the nose of the aircraft which has an excellent vantage point and which recorded all 37 sequences. For this reason, all experiments use this fixed viewpoint. Each sequence is roughly an hour long, recorded at 10 frames per second, starts a few minutes before an aircraft arrives and ends a few minutes after the aircraft departs. As such, each sequence wholly contains a complete aircraft servicing operation.

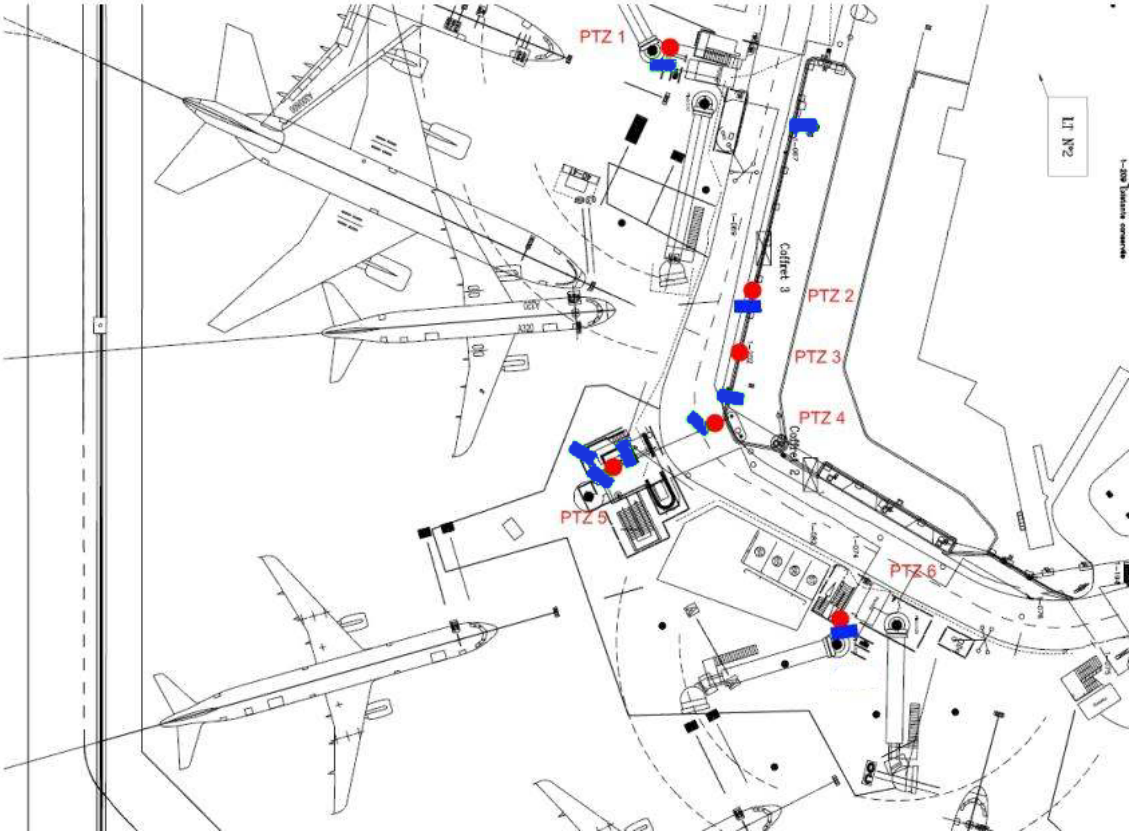


Figure 3.6: Layout of cameras on apron in Toulouse. Static cameras are marked with blue rectangles. Pan Tilt Zoom (PTZ) cameras are marked with red dots. Due to hardware limitations, all cameras were not processed simultaneously.

The event vocabulary comprises 12 servicing event classes which can take place on the apron. Snapshots from the events can be seen in Figures 3.7 and 3.8. A brief description of each event class follows:

- Aircraft Arrival: Starts when an aircraft turns off the road which connects the apron

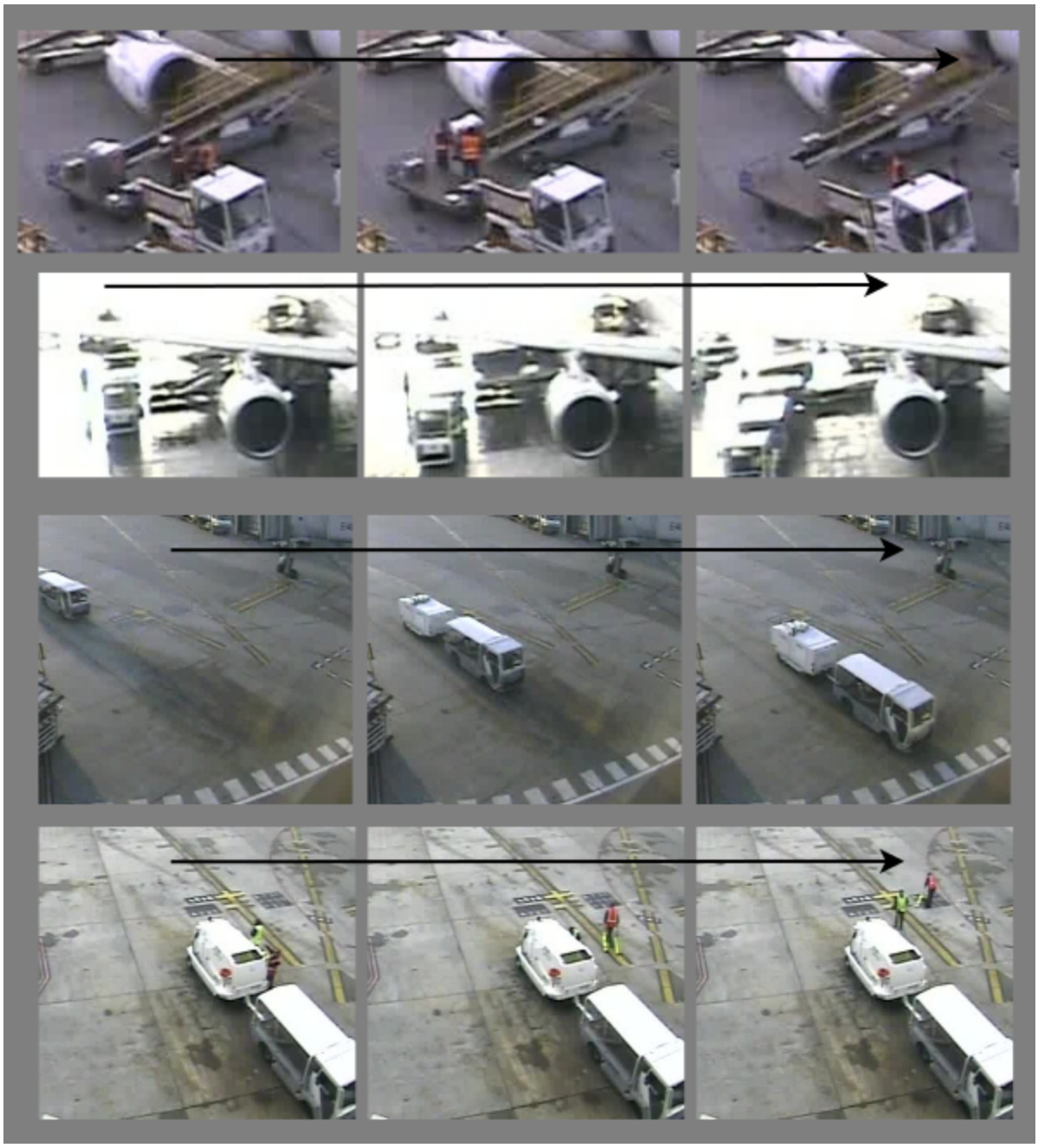


Figure 3.7: Images demonstrating the events in the Co-Friend dataset. The images are cropped to the zones of interest related to each event. From top to bottom, the events are as follows: Front Belt Loader Loading, Rear Container Loading, GPU Positioning, Handler Deposits Chocks.

Figure 3.8: Images demonstrating the events in the Co-Friend dataset. The images are cropped to the zones of interest related to each event. From top to bottom, the events are as follows: Rear Belt Loader Unloading, Aircraft Arrival / Departure, Container Front Loading, Passenger Boarding Bridge Positioning.

areas to the runway and ends when it comes to a complete stop in position on the apron.

- Aircraft Departure: Starts when the Aircraft is moved from its mark by the Push Back and ends when the Aircraft has been pushed completely off the apron.

- GPU Positioning: Starts when the Ground Power Unit becomes visible on the apron and ends when it is completely stopped in position.

- Handler Deposits Chocks: Starts when the Handler takes the Chocks from where they are kept in the rear of the GPU and ends once he has placed them in position.

- PBB Positioning: Starts when the Passenger Boarding Bridge starts moving from its default position and ends once the PBB has completely connected to the aircraft.

- PBB Removing: Starts when PBB disconnects from aircraft and ends when PBB is completely at rest in its default position.

- Belt Loader Unloading: The Belt Loader will normally move into position by the Aircraft's rear luggage hatch very soon after the Aircraft Arrival. The hatch will normally open around this time too. These are necessary prerequisites for the event to start but are not considered to be part of the Belt Loader Loading event. The event starts when a train of empty trolleys halts at the foot of the Belt Loader. Normally at this stage one or two handlers will walk up the belt and disappear into the hatch where they will transfer bags from the hold to the belt (on some occasions these handlers may already be in position). The belt will start moving and a number (between 1 and 4) of luggage handlers will then transfer the luggage that is appearing on the belt and travelling downwards one piece at a time from the belt to the trolleys. The event ends when the stream of luggage stops and the train of trolleys moves away (either because it is full or there are no further bags in the hold).

- Belt Loader Loading: This event almost always occurs after Belt Loader Unloading, so again the Belt Loader is assumed to be in position with the luggage hatch open. The event starts when a train of trolleys laden with luggage halts at the foot of the already-positioned Belt Loader. Normally at this stage one or two handlers will walk up the belt and disappear into the hatch where they will stack the luggage that comes up the belt (on some occasions these handlers may already be in position). A number (between 1 and 4) of luggage handlers will then move the luggage one piece at a time from the trolleys to the belt. The luggage then travels up the belt and disappears into the hatch (where further handlers are waiting) . The event ends when the train of trolleys has been cleared of luggage and is moved away from the foot of the loader

- Container Front Unloading: Starts when the ramp on the loader rises up to the cargo door. One or two large containers will then be pushed out onto the ramp, before the ramp is lowered. The containers will then be pushed off the ramp onto trolleys. The event ends when the trolleys are pulled away from the foot of the loader.

- Container Front Loading: Starts when a train of trolleys laden with containers arrives at the foot of the loader. One or two containers will be pushed onto the ramp. The ramp will be raised and the containers pushed through the cargo door. The event ends when the ramp has returned to its lowest point.

- Container Rear (Un)Loading: These events are identical to the Front Unloading/Loading events, but they take place at the rear cargo door.

Some of the events on the apron are very simple. Events falling into this category would be the Aircraft Arrival/Departure, the Passenger Bridge Positioning and the Ground Power Unit Positioning, which all involve a single agent and a distinctive pattern of motion. In contrast are events such as Belt Loader Loading, which involve a variable number of participants and have greatly increased variability in duration and associated motion.

An additional challenge in the dataset is the variable weather conditions: several turnarounds include rain and one takes place in snow. Lighting conditions also vary, with two turnarounds in the glare of the early morning sun, three taking place at sunset and one taking place after dark. There are a few attributes of the dataset however which make things easier. Firstly, as the airport apron is a highly regulated environment, there isn't too much incidental activity. If the Aircraft Arrival is delayed, the ground staff might be observed milling around trying to entertain themselves whilst they wait and sometimes equipment not directly relevant to an ongoing turnaround may be parked in designated waiting areas around the edges of the apron but in the main most of the activity that occurs on the apron is relevant to the turnaround.

Several of these events involve very slow and gradual movements, whilst in others objects accelerate rapidly, which could pose difficulties in selecting appropriate interest point detection for feature extraction. To handle the variation, it might be possible to tune interest point detectors for each event but here dense sampling is simply used in all cases, leaving the non-linear classifier to distinguish the features which are relevant.

The ground truth for the dataset consists simply of start and end times for all event instances across the 37 sequences. Initially, 7 sequences were annotated by a member of operational staff from the airport. The remaining 30 sequences were annotated by researchers on the project adhering to the same event specifications as summarized previously in this chapter. The airport apron is divided into a number of technical areas, which align with certain parts of the aircraft. These technical areas are static (and are actually in some cases painted on the ground) rather than relative to the position of the aircraft. The zones are aligned with parts of the plane simply because each apron is designed to take a fixed set of different aircraft types and the aircraft is parked in precisely the same location after every landing. Due to the highly-regulated nature of the environment, all events in the vocabulary can be associated with one of these zones. This association was given in documentation made available by the airline involved in the project. These zones

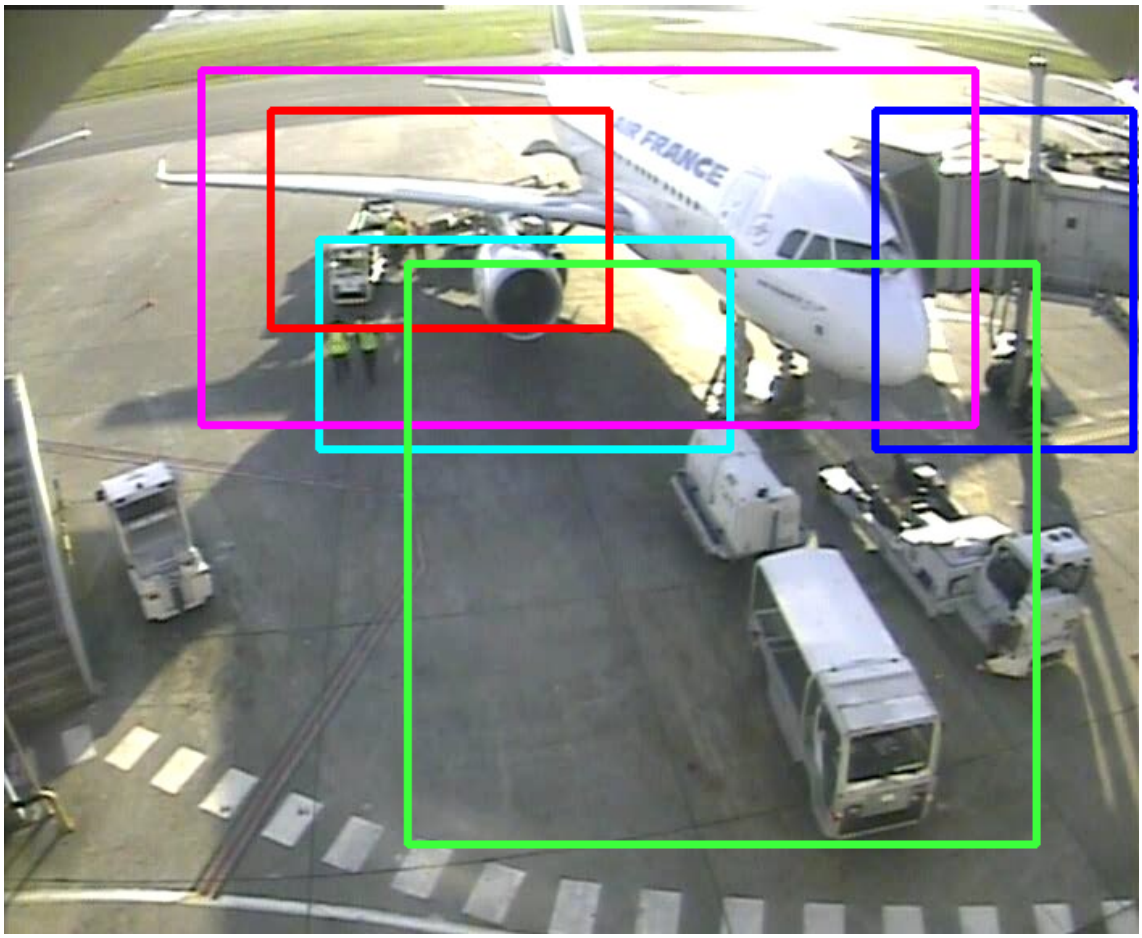


Figure 3.9: Zones of interest within which dense local features are extracted for event detection.

are exploited as follows in this project: The zone boundaries in world co-ordinates are back-projected to the image plane using camera calibration information. The minimum enclosing axis-aligned bounding box that contains this back projection is then taken to represent each zone in the image plane. Detectors for each event type then use densely extracted local features within the relevant zone. This was a simple and obvious way to eliminate a fair amount of noise in the detectors for free, however they are not critical to method. Experiments were performed initially without zones of interest and results were just a couple of percentage points worse in terms of Mean Average Precision for most events. The zones used are depicted in Figure 3.9. Note that the zones are overlapping and that the correspondence between zone and event is not 1-1 since several events share

the same zone. One zone corresponds to the resting place of the aircraft on the apron. This zone is associated with aircraft arrival and departure. There are zones corresponding to the front and rear hatches of the aircraft which are associated with the loading/unloading events. Another zone represents the technical area around the nose of the plane and is associated with the GPU Positioning and Handler Deposits Chocks events. The final zone covers the area around the aircraft's front door and is associated with the PBB Positioning/ Removing events.

3.2.1 Extracting and storing features

Based on the experiments earlier in this chapter HOF features were selected with block dimensions $3 \times 3 \times 2$ and an angular granularity of 4 using l^2 normalization and bilinear binning. Since the scale of the dataset is fixed (with fixed camera and events happening in roughly the same positions) for efficiency, features are extracted at a single scale, with cell dimensions $18 \times 18 \times 10$, sampling densely with an overlap of 50%. The cell size of 18 pixels means that the height and width of a block is around the same size as a pedestrian standing near the nose of the plane. A codebook of 4000 words is used, which was trained on the Hollywood dataset. Since the feature extractor runs at between 3-5 frames per second (depending on the size of the zones of interest) and the dataset consists of 1.7 million frames, for convenience in these experiments a high performance computing facility is used to do the feature extraction, saving the features to file to be used in later experiments. For each zone, within each sequence feature extraction yields a matrix of dimension $4000 \times T$, where T is the sequence length divided by the sampling rate (in this case every 10 frames). By summing through time, a cumulative histogram is generated, which allows the histogram for any sub-sequence to be calculated with just one column subtraction.

3.2.2 Training the classifier for detection

The challenge presented by the Co-Friend dataset is slightly different to that of the action classification datasets such as Hollywood and KTH. Those datasets are entirely composed of clips which fall into one of N classes between which it is necessary to differentiate and the numbers of instances in each class is relatively balanced. Therefore since the size of the datasets is manageable, once the division is made between test and training sets the entire set of data can be used for training the classifier. In Co-Friend , there are 37 extended sequences which in total contain between 10 and 40 instances of each event type which have been annotated. These define the positive examples. It is then necessary to collect negative examples from elsewhere within the 37 hours worth of footage in the dataset. Since the events have variable duration, it is necessary to detect over multiple possible durations as well as midpoint therefore the negative samples should have durations across the ranges to be tested. At detection time, it is deemed sufficient to search ±2 standard deviations from the mean duration for each event type, which thus leads to the sampling of negative intervals by sampling midpoint uniformly over entire sequence length and sampling duration $\sim \mathcal{N}(\mu_m, \sigma_m)$, where μ_m and σ_m are mean and standard deviation over duration for event type m.

It is possible to access a vast number of negative samples if it is permitted to extract them from overlapping intervals, as shown in Figure 3.10. To obtain a balanced classifier, positive and negative examples are weighted proportional to the numbers in each category so that if there are 40 positive examples and 9960 negative samples each positive example is worth 9960/40 whilst each negative sample is worth one. The larger the number of samples used however, the more time that will be required for training and potentially for prediction (depending on how many support vectors are retained) so it is not desirable to use more data than is required. The number of samples to use is decided empirically by the following method. First 10,000 random samples are drawn from across all 37 sequences to act as the test set, keeping the data from each sequence separate. To evaluate

the performance with N training samples, N random samples are drawn from the 37 sequences and 5-fold cross validation performed on the test set, applying weighted scoring as described above; the folding ensures that there is no overlap between train and test data by separating based on sequence. This procedure is repeated 5 times for several N across 4 different event types. The results of these experiments are presented in Figure 3.11. From these plots, it is clear that performance improves rapidly up to around 1000 samples, then continues to improve slowly up to around 10,000 samples where it appears to saturate. Note this effect is less pronounced on the graph for the Aircraft Arrival event since the event is so easily detected that with only 100 samples, the event is classified with 99.96% accuracy. This means there are just a handful of difficult examples in the test set which makes the performance increase in relation to the number of samples used appear much less smooth.

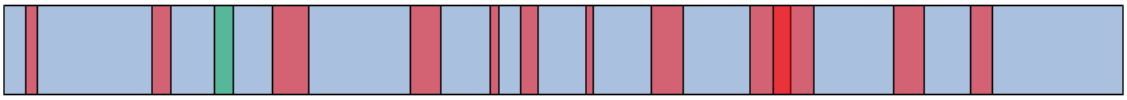


Figure 3.10: Drawing samples from video sequence to build classification training data. Green samples represent positives, red samples negatives

3.2.3 Detection performance

Having trained the classifiers, their detection performance is evaluated. For each event in each sequence, histograms corresponding to every possible midpoint ($t \in T$) and all durations $d \in \{\mu_m - 2\sigma_m, .., \mu_m + 2\sigma_m\}$ are passed to the corresponding classifier. Note that midpoint and duration are used rather than start and end time as this representation makes inference simpler when the temporal structure model is applied in the following chapter. The probabilities returned from the classifier are stored in a matrix of dimension $T \times 4\sigma_m$. To give an idea of how the detectors for the various events perform, some of these matrices are plotted as heat maps in Figures 3.12 and 3.13. Ground truth instances

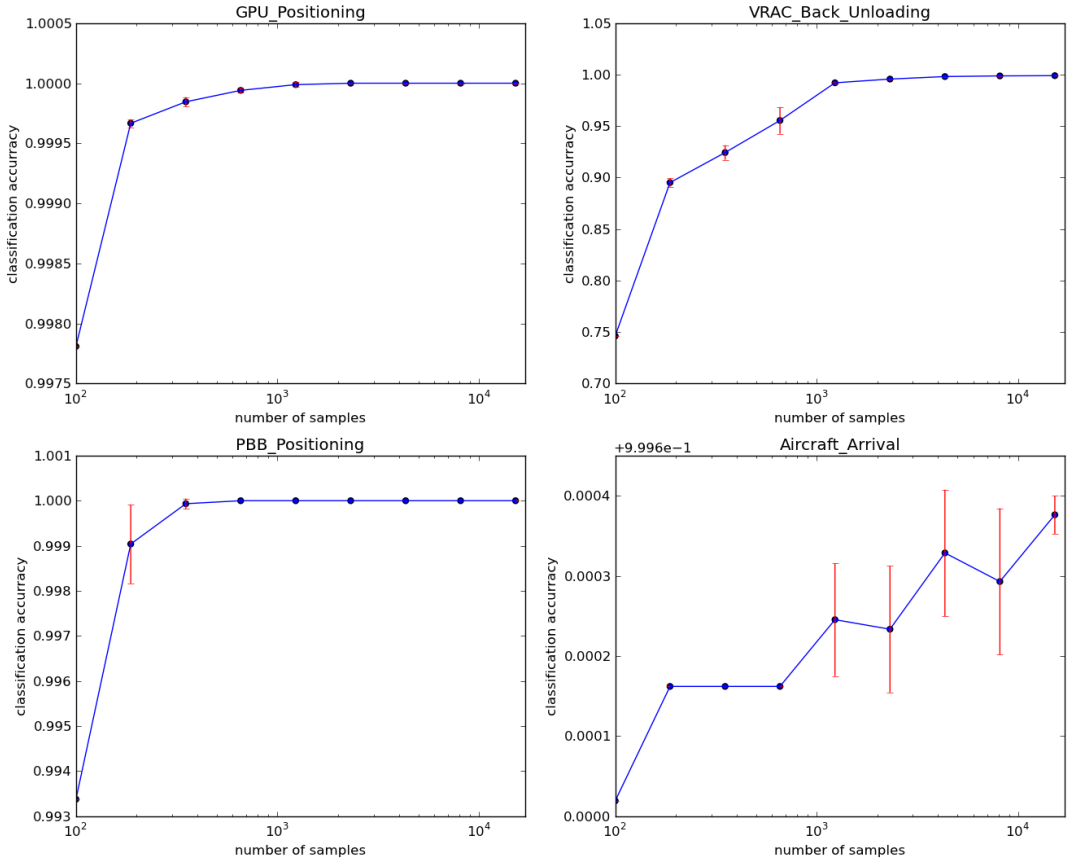


Figure 3.11: Event-wise classification performance versus number of samples used for SVM training. The error bars show standard deviation across 5 runs.

are marked white triangles.

It's clear from these plots that some events are much easier to detect than others. As one might expect, Aircraft Arrival and Aircraft Departure are detected almost perfectly, with spikes of very strong response coinciding with ground truth instances and very low response outside of these intervals. Note that these spikes appear as vertical bands meaning that the detector is less good at determining the true duration of the event; all intervals with a high relative overlap with the ground truth induce some response. This effect is evident across all event types to varying degrees. It is worth noting that the scale of the duration axis is different for each event to reflect the differing variance in event duration.

Aircraft Arrival for example is a fairly short event with a fairly consistent duration therefore we only have to search over a handful of durations compared to the much longer and more variable duration for VRAC Back Loading. The problems exhibited by the Handler Deposits Chocks and GPU Postioning events are quite similar; there are quite sharp distinct peaks, but many of them are false positives due to the fact that there is lots of unrelated noisy motion on the apron which resembles these two events. GPUs drive through the scene heading towards other aprons and may follow a similar pattern of motion to those parking in the scene. Ground staff often loiter around the nose of the plane and it is understandable how this could induce similar HOF features to the handler placing the chocks.

The performance of the thresholded independent detectors is reported in Table 3.4. In all experiments, the leave-one-out protocol was followed, training on 36 recordings and testing on the remaining one. The experiment is repeated with each sequence as the test sequence. For a detection to be considered a hit, detections are required to have a relative overlap (RO) > 30% with ground truth. Where RO between two intervals A,B is defined as

$$RO(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{3.4}$$

Increasing the overlap level required had little effect on most events up to around the 60% mark, after which it degraded rapidly. This is likely due to the independent detectors distinguishing only the main significant motion associated with each event, and not necessarily the less distinctive but semantically important start and end points. Two figures are presented for each event; the Average Precision (AP) and the Equal Error Rate (EER). The AP is calculated according to the methodology preferred in the Pascal VOC Challenge 2011 [24]. This involves computing a precision-recall curve with precision monotonically decreasing, by setting precision for recall r to the maximum precision ob-

tained for any recall $r' \geq r$, and sampling the curve at all unique recall values. AP is calculated by numerical integration of this curve. No approximation is needed since the curve is piecewise constant. The Equal Error Rate is calculated by taking the point on the curve at which precision and recall are equal.

Both of these figures are of interest since AP gives a strong measure of the detector's performance across the whole range of recall levels. This is the standard measure used in object detection literature and also in some works relating to low level action recognition. AP isn't seen in use so much in high level activity recognition. This is partly because works in this area haven't been as intensely results-oriented and also because in many high level systems, it is not the case that a single threshold can be varied which will influence precision and recall across the board. In the following chapters, an effort is made to extract AP on an event-by-event basis, yet given the interdependence of events in the scenario recognition context, these numbers seem slightly artificial. Many of the decisions which must be made in building a scenario recognition system would be very different dependant on whether the goal was to squeeze an extra bit of precision at 100% recall or else to maximize the recall at 100% precision. Since the stated goal of the Co-Friend project was to build a system which would be helpful to airlines for part automating record-keeping on aircraft servicing operations, there was no cost function supplied to weight false positives versus false negatives. The Equal Error Rate was therefore selected as a reasonable point on the precision-recall curve to attempt to maximise.

Event	Occurs	AP (%)	EER (%)
Aircraft Arrival	37	98	95
Aircraft Departure	37	100	100
Passenger Bridge Positioning	37	70	63
Passenger Bridge Parking	37	91	79
Ground Power Unit Positioning	37	87	82
Handler Deposits Chocks	40	36	33
Container Front Door Loading	13	79	65
Container Front Door Unloading	14	72	67
Container Rear Door Loading	27	51	55
Container Rear Door Unloading	25	75	60
Belt Loader Loading	36	46	48
Belt Loader Unloading	25	80	83

Table 3.4: Co-Friend performance figures in terms of Average Precision and Equal Error Rate.

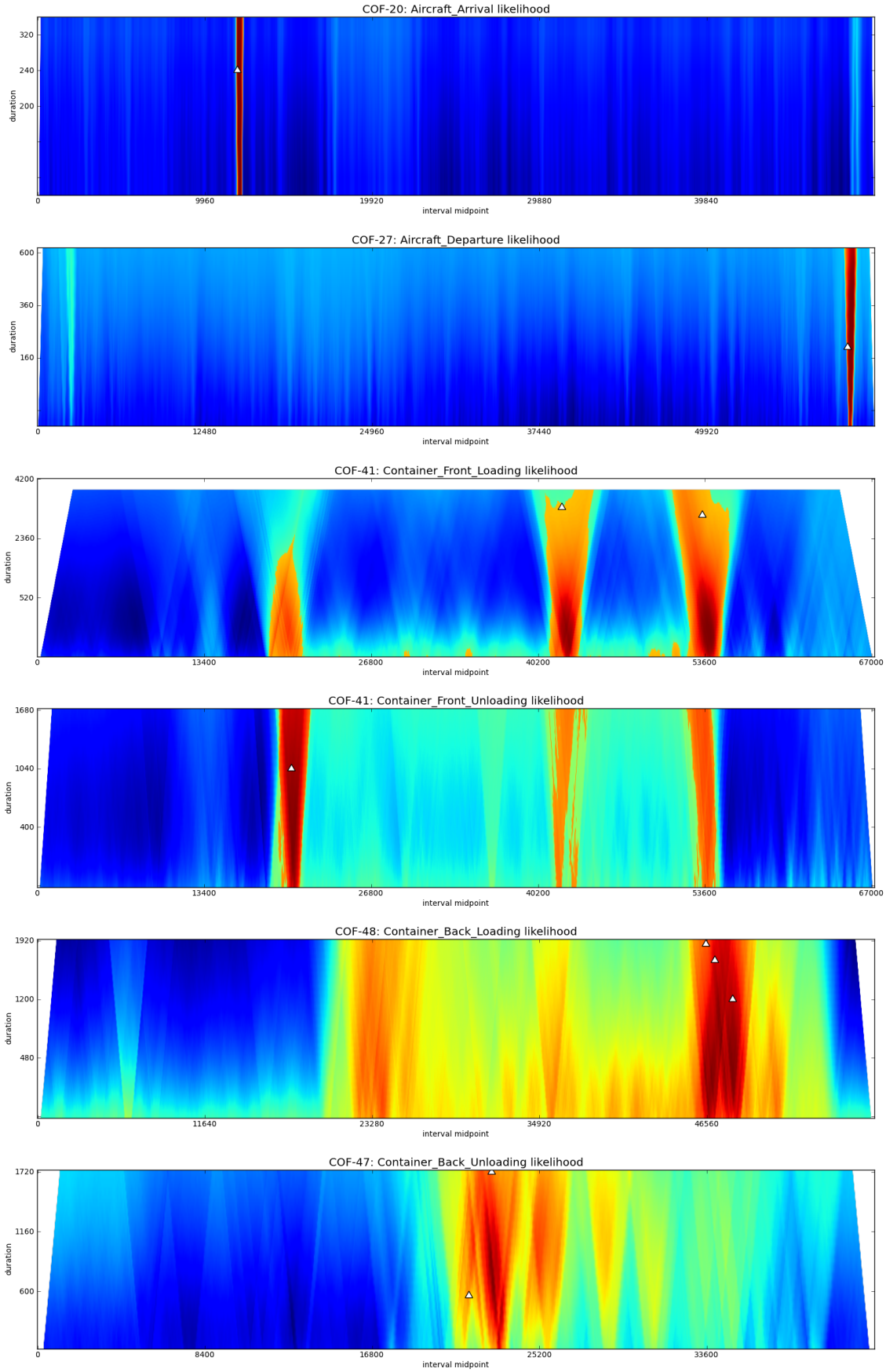


Figure 3.12: Heat map showing probabilities output from independent event classifiers at all candidate intervals for various events and sequences, with probability going from blue (lowest) to red (highest). Ground truth is marked by tiny triangles. The vertical axis is duration (which is scaled relative to the standard deviation of duration of each event type). The title for each the graph contains the name of the event and the identifier of the sequence from which it was taken. Note that the samples shown have been chosen from different sequences as no single sequence contains instances of every event type.

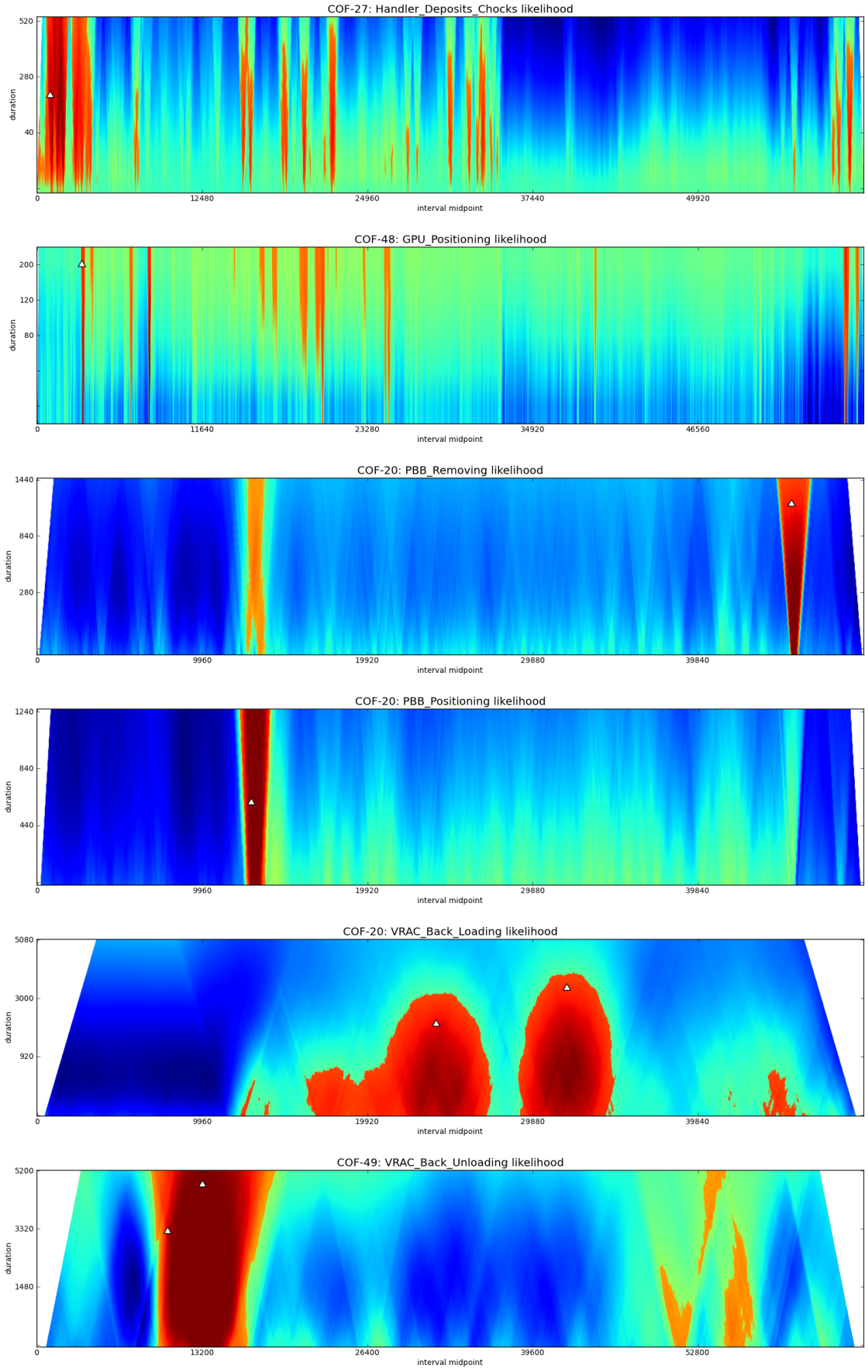


Figure 3.13: Additional heat maps of probabilities output from independent event classifiers at all candidate intervals for various events. A trapezoid shape is noticable in events with long durations as only intervals which are wholly contained within the sequence are evaluated.

3.3 Summary

In this chapter the implementation of a state of the art detector for human action recognition has been described. By experimentation on standard datasets it has been verified that that a bilinear binning strategy for HOF features outperforms the coarse binning strategy used by all previous works. Experiments were also performed in retaining flow magnitude information, but it was found that performance was reduced versus l^1 and l^2 block normalization. The application of the detectors to the aircraft servicing domain was described and the effect of increasing the number of samples drawn from the video dataset for training the classifiers was evaluated. Performance appeared to saturate with ten thousand negative samples. The detector performance was near perfect for some of the simplest and visually distinctive events in the Co-Friend domain such as aircraft arrival, but was much less reliable for other events. In the folowing chapters, these detectors can serve as both a baseline and low-level input to a high level scenario model which exploits temporal relationships between events for the purpose of boosting performance.

Chapter 4

Tree-Structured Temporal Structure Models

In the previous chapter it was demonstrated that a state of the art event detector based on low level features performs strongly on detecting several types of event and less strongly on others. Whilst it might be possible to overcome this problem by tailoring a method to fit each individual event in the target domain, this would make the method problem specific. The focus of this chapter is to find a method for boosting performance in a more transferrable manner through exploitation of the temporal structure that exists between events in many domains. In the previous chapter, the Co-Friend dataset was introduced and it is this dataset which is used for experimentation henceforth. The model would generalize well, particularly to industrial process environments such as production lines or warehouses, where activity is structured and periodic. In the following few pages, the aircraft servicing scenario and the Co-Friend dataset are analyzed, but many of the characteristics and limitations of the Co-Friend dataset would likely apply to many real-world datasets, where training data is often difficult and expensive to obtain. Whilst in video processing and data terms the Co-Friend dataset is quite large (with > 40 hours footage), at a scenario level it is smaller than one would like, containing only 37 turnarounds. It

would have been desirable to obtain more to increase the significance of the results and the robustness of the learnt temporal models but doing so was impossible as it required further cooperation from the airport, whose resource allocation on the project was limited. On the other hand, this limitation can also be seen in a positive light. If it is possible to construct a temporal model which is proven to be effective even in complex scenarios with relatively small amounts of training data, then this would make it all the more useful.

The matrix in Figure 4.1 gives co-occurrence counts on event types in the Co-Friend dataset on a pairwise basis across the 37 sequences. Observe that all pairs of different event types co-occur in at least one sequence, but for some pairs the number of co-occurrences is as low as 1. Additionally, there is a strict ordering on only a small subset of these events (most events occur before Aircraft Departure) but in the main there is no particular sequence to the events. The timings of the different events appear to be correlated rather than directly causal. Figure 4.2 shows the strict ordering constraints, where they exist. Finally, it is quite common for events to overlap in time. This factor in particular causes problems for state-based representations such as an HMM for which that means an explosion in the state-space.

In this chapter a model that is more closely related to Pictoral Structure Models (PSMs) is constructed. PSMs were introduced by Fischler and Elschlager [29] in 1973. More recently, Felzenszwalb and Huttenlocher [27] showed that the original formalization of PSMs as an energy minimization problem could be derived elegantly through Bayesian probability theory and it is their notation that is followed in this chapter. The PSM for an object can be described most simply as a collection of parts with connections between certain parts. The structure of the PSM is thus stored naturally as an undirected graph, where the vertices correspond to the parts and there is an edge for each pair of connected parts. In structured environments, it is possible to consider a *scenario* to be a collection of *events*, some of which are (temporally) related.

In applications of PSMs, the connections are almost always considered on a pairwise

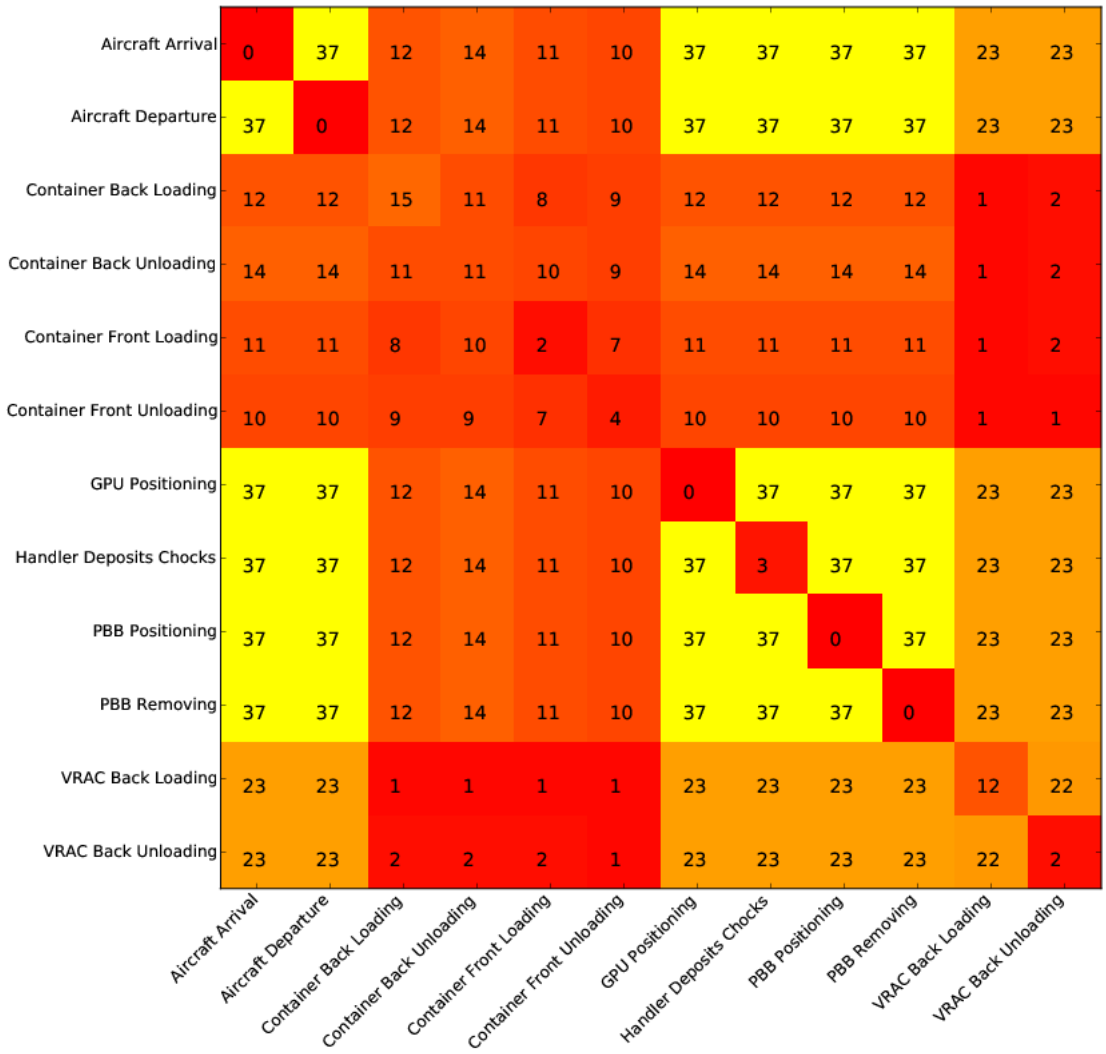


Figure 4.1: Matrix showing number of sequences in which events of a particular type co-occur in the Co-Friend dataset. It is clear that even when looking for co-occurrence only on a pairwise basis, the dataset appears quite sparse. Note that the lead diagonal gives the number of turnarounds in which more than one instance of that event type can be observed.

basis, though this simplification isn't intrinsic to their definition. In the scenario recognition case, this assumption could be particularly beneficial as it could reduce the demands on the amount of data required to train the model. This is based on the assumption that the training data consists of a number of sequences, each of which contains a subset of the events from an event vocabulary. Therefore the higher-order the relationship one is

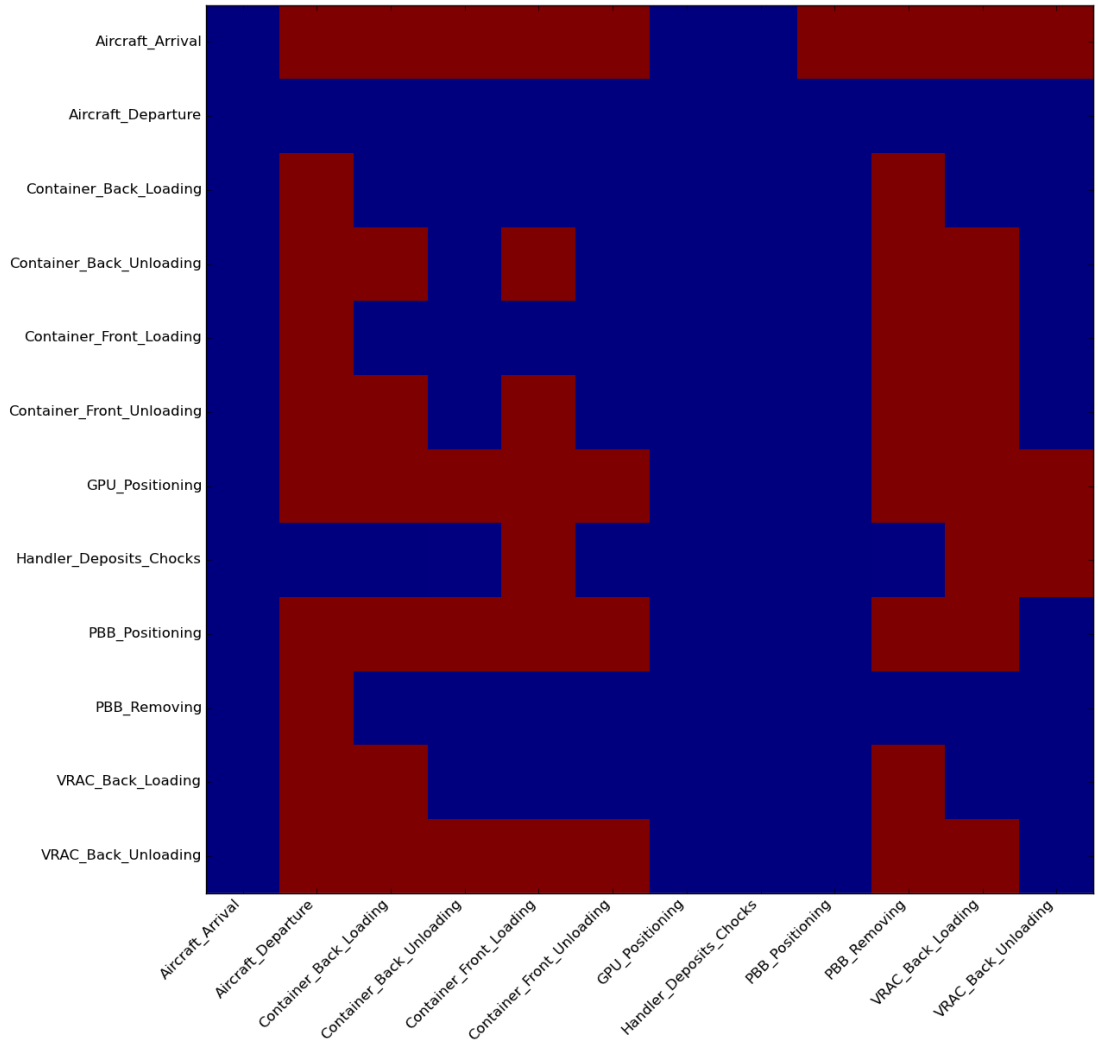


Figure 4.2: Matrix indicating where event pairs are strictly ordered. An entry is coloured red if instances of the event type given by that row always precedes instances of events of type given by column.

attempting to model, the fewer the instances of co-occurrence one will have to learn from in the dataset and the higher the dimension of the distribution to be modelled. PSMs are normally used as object detectors so that for an image, candidate regions are proposed by a simple sliding window or perhaps a simple coarse classifier. The most likely configuration of parts within each window is found through optimization of the posterior likelihood composed of the product between part appearance likelihood terms and part configuration priors. The posterior likelihood in this optimal configuration then gives the probability

that the candidate window contains the object. In the object detection problem for which PSMs are most commonly applied, the most important output is simply this probability, rather than the configuration of the parts; the parts are just a means to an end. PSMs have also been used successfully for human pose estimation, which more closely mirrors the focus in the scenario recognition situation, where the parts of the model, namely the events, are of more concern than the scenario which is already determined by the domain. The scenario recognition task however, does have an extra degree of difficulty in that it is not known *apriori* how many occurrences of each event to expect within a video sequence. This issue exists to some extent in object detection since parts may be occluded. In [26] , the problem isn't tackled directly though by learning multiple models per object class, the self-occluding parts issue is mitigated by having different part sets and configurations for diffferent viewpoints. In pose estimation, the task is normally defined as producing the best possible estimate for a fixed number of joint locations, whether or not they are visible. Annotators of datasets are typically asked to mark locations of occluded joints with their 'best guess', though this precedent is questioned in [37] , where a large dataset is assembled using Amazon Mechanical Turk and annotators are instructed to only annotate visible joints. Johnson and Everingham's training procedure is thus designed to cope with joint locations that are missing due to occlusion, which ensures their spatial prior is not muddied by inaccurate and inconsistent guessed joint locations. At runtime however, estimates for all joint locations are produced and no effort is made to distinguish between visible and occluded parts. There have been some works however, whose primary focus is occlusion. The problem of primary concern in Sigal and Black [70] is that PSMs for human pose recognition are seriously affected by self-occlusion and the similarity in appearance of limbs. In their terminology, the spatial prior is 'fighting' the observation likelihood, which they acknowledge is necessarily independent across parts. In cases where one leg is occluded, the likelihood will strongly encourage both legs to be localized to the same image region. Their mechanism for dealing with this is through two additional

binary hidden variables for each part at each pixel, to indicate whether a part is occluded by or occludes others at this pixel. There is a depth ordering over the limbs which is fixed for a given viewpoint, which allows these additional variables to be incorporated into the message passing optimization without too much increase of computational complexity. This strategy seems effective, yet quite specifically targets the problem of self-occlusion and it is difficult to see a natural extension to the scenario recognition domain; where an event being present or not within a sequence is not as a result of being occluded by another event. Eichner and Ferrari [23] tackle the problem differently, introducing an extra 'occlusion' state for each body part, to enable the optimization to proceed in an otherwise familiar fashion. The main novelty in this work is in the way the energy for the occluded state is calculated through an occlusion prediction system which combines the simple likelihood of any given part being occluded by frequency of occlusion in the training set, with more complicated terms dependent on the colocation of other people and limbs. The problem with this work is that the additional complexity makes the optimization much more difficult and a series of domain- specific constraints, such as the requirement that heads should always be visible, must be introduced in order to regain tractibility. In this chapter, a new approach to dealing with the problem of having an unknown number of instances of each event is introduced, which is efficient and avoids any domain-specific constraints.

4.1 Notation and problem description

Given a previously unseen video sequence of length T, the task is to determine the temporal midpoint, $t \in \mathcal{T} = \{1,..,T\}$, duration, $d \in \mathcal{D} = \{\mu_c - 2\sigma_c,..,\mu_c + 2\sigma_c\}$, and event class, $c \in \{1,...,M\}$, of an unknown number, N, of event *instances*, from a fixed vocabulary of M event *classes*, where μ_c, σ_c are mean and standard deviation of duration of events of type c. This is depicted in Figure 4.3 (right). The scenario recognition problem over a sequence

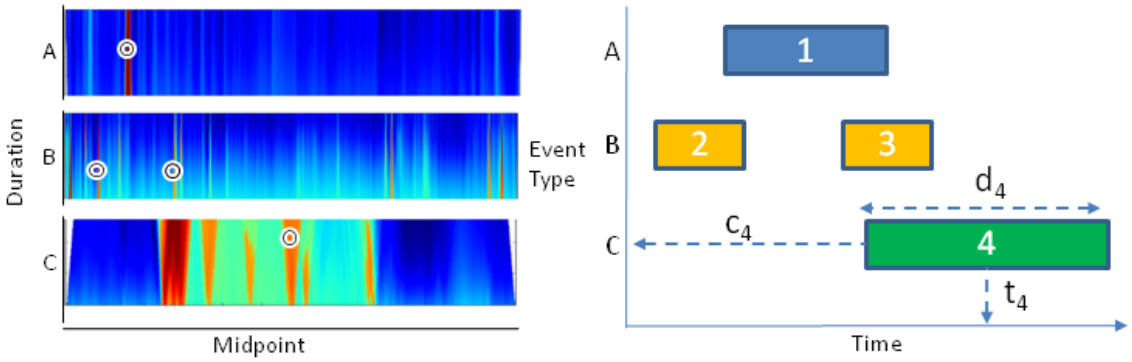


Figure 4.3: Left: Example response from 3 independent probabilistic event detectors. The detectors give a response for each possible interval in a sequence, with colour indicating the probability (red being high). Four event instances are shown localized on these likelihood streams. Right: Illustration of corresponding sequence scenario. The scenario is fully specified by event class vector $C = (c_1, ..., c_4)$ and temporal information $Y = ((t_1, d_1), ..., (t_4, d_4))$.

is challenging since it entails determining the correct number of event instances, N, then optimizing over these $3N$ random variables. There are several possible approaches to dealing with the problem of unknown N and these are addressed later. For now the problem is mitigated by introducing a'Null' state for each (t, d) so that $(t, d) \in (\mathcal{T} \times \mathcal{D}) \cup Null$ as a mechanism for 'switching off' detections without having to dispose of the corresponding random variables entirely. To simplify matters, for the moment the assumption is that there is at most one event of each type in each sequence. Again, the problem of multiple instances of one event type within a sequence is addressed later. Under these assumptions, it is now possible to fix the number of event instances in the scenario optimization to be M; fixing $C = (c_1, c_2, ..., c_M) = (1, 2, ..., M)$ and leaving $Y = ((t_1, d_1), ..., (t_M, d_M))$ to be optimised conditioned on C and the video sequence, X.

A probabilistic event detector for each event class has already been trained, and it is assumed that these detectors will be run in parallel to evaluate every possible midpoint and duration over the discrete temporal domain at detection time. Adopting a Bayesian perspective, the output of the independent event detectors can be treated as the likelihood of the relevant chunk of the observed video sequence, X, arising as a result of the given

event scenario, which is termed $p(X|C,Y,\theta)$, where θ is the set of parameters that define the model. Therefore $p(X|c_i,(t_i,d_i),\theta)$ is known for all possible $(t_i,d_i) \in \mathcal{T}^2$. The probability of the event being *Null* is defined as proportional to the complement of the maximum of the probability of it having occurred at any particular timestep. This enforces a suitable penalty for writing off the output of an event detector as noise and allows for the insertion of a parameter, α_{c_i}, into the model for each event class in order to vary the sensitivity of the detector.

$$p(X|(t_i,d_i) = Null, c_i) = \alpha_{c_i}(1 - \max_{t_i,d_i}(p(X|(t_i,d_i) \in \mathcal{T}^2, c_i))) \tag{4.1}$$

It is then possible to concentrate on optimizing the posterior

$$p(Y|X,C,\theta) \propto p(X|C,Y,\theta)p(Y|C,\theta). \tag{4.2}$$

The distribution $p(Y|C,\theta)$ denotes the prior probability of a scenario given its event set and the model. By analogy to Pictoral Structure Models [27], this prior is defined to be a tree-structured Markov Random Field (MRF). The general form for the joint distribution of such a prior can be written

$$p(Y|C,\theta) = \frac{\prod_{(i,j)\in E} p((t_i,d_i),(t_j,d_j)|c_i,c_j,\theta)}{\prod_{i\in V} p((t_i,d_i)|c_i,\theta)} \tag{4.3}$$

where E is the edge set and V the vertices of the MRF. The absolute timing of any event in isolation is presumed to be unhelpful, so therefore $p((t_i,d_i)|c_i,\theta)$ is set to be uniform, so that Equation 4.3 simplifies to.

$$p(Y|C,\theta) \propto \prod_{(i,j)\in E} p((t_i,d_i),(t_j,d_j)|c_i,c_j,\theta) \tag{4.4}$$

This prior has the following desirable characteristics. Firstly the tree structure ensures

that the configuration is to some extent globally consistent since all random variables are (indirectly) connected. Secondly there is the dependence only on pairwise co-occurrences to help cope with sparse training data. Finally, the tree-structure allows exact inference on the prior, meaning globally optimal solutions may be quickly obtained.

4.2 Training a Temporal Structure Model

Let the dataset consist of m video sequences $\{X^1, X^2, ..., X^m\}$ with corresponding temporal annotations $\{Y^1, Y^2, ..., Y^m\}$ and event sets $\{C^1, C^2, ..., C^m\}$. The training examples are to be used to obtain estimates for the model parameters θ, which for the first time are made more explicit; $\theta = (A, E, T)$. A is termed the appearance parameters, which comprise all the parameters relating to the classifiers trained in the previous chapter. E is the set of connections referenced in Equation 4.3 and T is a set of probability distributions over all possible pairs of event types.

4.2.1 Learning Inter-Event Priors

The first component of the prior that is learnt is the set of $\frac{M(M+1)}{2}$ pairwise probability distributions, $\phi_{c_i,c_j}(t_i, t_j)$, which are defined to be probability distributions on the time differences between midpoints of instances of each possible pair of event classes (including the cases where $c_i = c_j$) appearing in the same sequence. These distributions will be combined with another term relating to co-ocurrence statistics to obtain $p((t_i, d_i), (t_j, d_j)|c_i, c_j, \theta)$. Note that temporal midpoints are assumed independent of durations for simplicity. In domains where the duration of events is small relative to the time between them, the information loss through this assumption will be small. The other assumption made is that the absolute timings of the events are not useful; so rather than learn a 2 dimensional distribution over all possible pairs of midpoints, a univariate distribution over time differences is learnt, then reference at test time to get $\phi_{c_i,c_j}(t_i, t_j)$ for all $t_i, t_j \in \mathcal{T}$. This procedure

is illustrated in Figure 4.4.

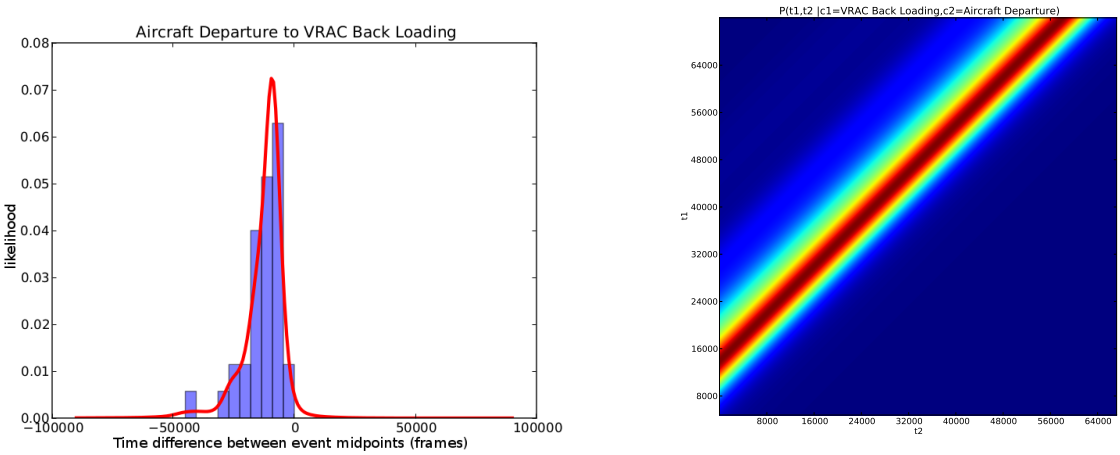


Figure 4.4: left :A learnt distribution over time difference (in frames) between two events evaluated over the range {-90000,...,90000}. right: Joint prior over two temporal midpoints generated by referencing the univariate distribution on the left, then renormalizing.

Initial experiments involved fitting a single Gaussian to the training data. The probability density function of a univariate gaussian distribution is given by

$$p(x) = \mathcal{N}(x|\mu,\sigma) = (2\pi\sigma)^{-1/2}\exp(-\frac{1}{2}(x-\mu)^2\sigma^{-1}), \tag{4.5}$$

where the parameter μ is the mean and σ is the variance.

However, it was observed that the single Gaussian is not a good approximator for many of the distributions in the test domain. In some cases, a distribution is observed that is very skewed, perhaps indicating that one event is supposed to occur soon after another; it can never happen before, yet sometimes it does happen much later. In other cases, there may be two or more distinct modes.

4.2.1.1 Gaussian Mixture Models

A Gaussian Mixture Model (GMM) with a sufficient number of components can model both skewed and multimodal distributions. A GMM is a weighted linear superposition of

Gaussians of the form

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{4.6}$$

GMMs are typically trained using the Expectation Maximization (EM) algorithm [5]. This involves the introduction of a set of k-dimensional binary latent variables $\mathbf{z}$. For each data point, x_n there is then an indicator variable z_n, which will be zero across all but one of the k dimensions indicating which component from the model was responsible for generating the point x_n. Optimization then proceeds by taking an initial guess at parameter values $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\pi}$ then iterating between the expectation step where z is updated according to $\mathbf{z}* = p(\mathbf{z}|\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})$ and the maximization step where the log-likelihood $log(p(\mathbf{X}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}))$ is maximized with respect to each of the parameters. The algorithm guarantees convergence to a local optimum and generally performs quite satisfactorally on large datasets. One of the main problems with this training procedure is that the problem is ill-posed in that the log-likelihood function can go to infinity if a Gaussian component becomes centred on a single point and its covariance is allowed to shrink to zero. This problem, and that of local maxima can be mitigated by running the algorithm multiple times with random initializations. Another issue however is that there is no automatic determination of the appropriate number of components, K, in this procedure; it must be prescribed. This could be selected based on domain knowledge, by trial and error, or by first running some kind of clustering procedure. Since it is desirable to avoid the requirement of any domain knowledge, it is convenient to simply try several different values of K. A concern is that frequentist (Maximum Likelihood) training methods are liable to overfit in the presence of limited training data. For this reason, Bayesian approaches to GMM training were also applied, which it is hoped might be more resistant to this effect without requiring the addition of arbitrary regularization terms.

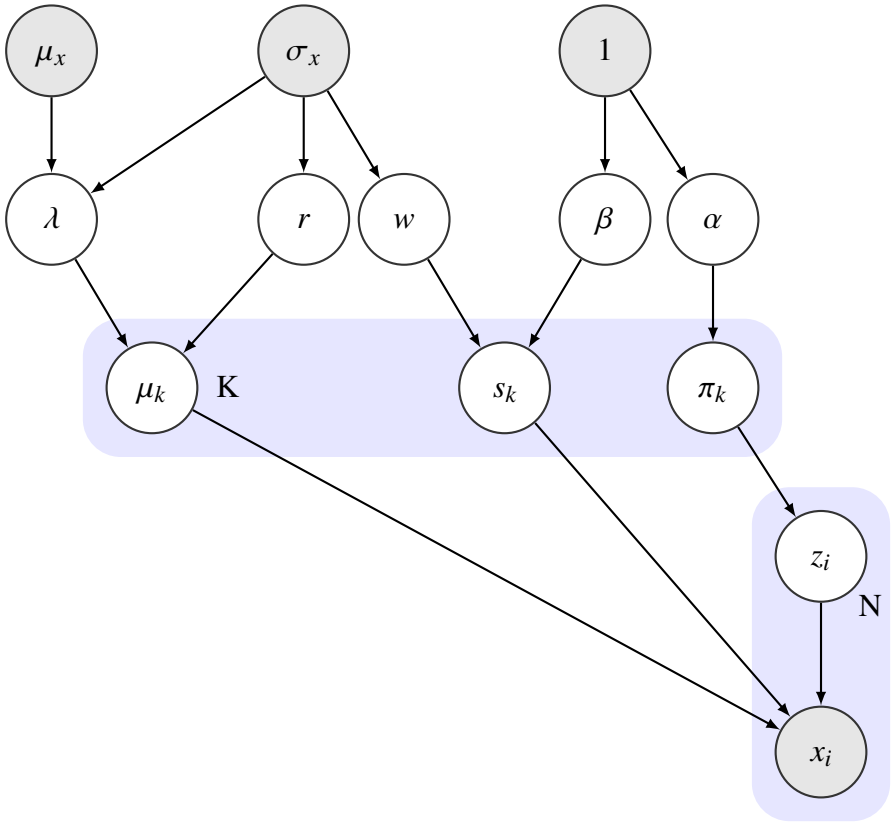


Figure 4.5: Graphical representation of Gaussian Mixture Model with Dirichlet Process priors. μ_k, s_k, π_k represent the mean, precision (inverse variance) and weight of the Gaussian components, of which there are K. For each of the N observations, x_i there is an indicator variable z_i, that indicates which component was responsible for generating x_i. The latent variables on the upper level are termed the hyperparameters, α is the concentration parameter of a symmetric dirichlet process prior on π_k. λ and r are the parameters of a Gaussian prior on m_k. w and β are the parameters of an Inverse Gamma prior on s_k. Note that distributions are also placed on the hyperparameters, though the parameters of this level are constants. μ_x and σ_x are the mean and variance of the whole dataset. See Section 4.2.1.2 for a full explanation.

4.2.1.2 Bayesian Training of Gaussian Mixture Models

The method of Rasmussen [64], which gives a thorough Bayesian treatment to the training of GMMs, was implemented. This approach places a symmetric Dirichlet Process prior over the mixture weights of the model, as well as 'non-informative' conjugate priors over the mean and variance parameters. Direct optimization with respect to the parameters is not possible, but conditional distributions for all the parameters are derived to facilitate Gibbs sampling. Gibbs sampling is a frequently used method for approximating a complex joint distribution by repeatedly resampling subsets of the variables based on their conditional distributions [48]. The model is depicted in Figure 4.5. The symbols used in the diagram are consistent with those used in the original publication, [64], to which the reader should refer for a full description of the model. The discussion of the model here is restricted to some key observations. Note that the component means, μ_k are given Gaussian priors:

$$p(\mu_j|\lambda, r) \sim \mathcal{N}(\lambda, r^{-1}) \tag{4.7}$$

whose mean λ and precision r, are hyperparameters common to all components. The hyperparameters themselves are given 'vague' normal and gamma priors:

$$p(\lambda) \sim \mathcal{N}(\mu_x, \sigma_x^2), \qquad p(r) \sim \mathcal{G}(1, \sigma_x^2) \propto r^{-1/2} \exp(-r\sigma_x^2/2) \tag{4.8}$$

where μ_y and σ_y^2 are the overall mean and variance of the training set. Having priors which are dependant on the data is not normal practice in Bayesian statistics, but Rasmussen [64] argues that it is reasonable in this context since it is equivalent to recentering and normalizing all the data and setting zero mean, unit variance priors. Experiments are included later using both this data-driven prior, and a broader prior reflecting a prior assumption that the timings of two events are independent unless enough evidence is observed to suggest otherwise.

In Figure 4.5, and throughout Rasmussen's derivation of the conditional probabilities

the number of components, K, is considered to be a fixed finite quantity. For all model variables except the indicators, the conditional posteriors in the infinite limit are simply obtained by substituting K for the number of classes that have data associated with them. Derivation of the conditional posterior of the indicators, z_i, in the infinite limit is slightly more involved but ultimately the dependence on K indicator variables in the prior is removed by application of the Dirichlet integral. This way, the number of components is simultaneously optimized with the other parameters in the model through Gibbs sampling. Allowing 50 iterations for burn-in, 20 samples are drawn evenly spread over 1000 iterations. Spreading the draw of the samples is to ensure that they can be considered to be independent draws from the posterior. These samples are then combined with equal weighting. The result of this procedure is a GMM with a large number of components which are highly-overlapped. The overhead of evaluating these PDFs can be eliminated at execution time by the use of look-up tables if required.

The model was coded in Python in this work, due to personal preference, but it became apparent during implementation that this would be a poor choice of language if speed were required. In many applications, it is possible to get performance in Python comparable to C through extensive use of matrix and vector operations with numeric libraries. However, it is difficult to vectorise operations in a Gibbs sampler since the procedure is by its nature iterative. As the method was only in this case used to do offline training of univariate distributions with small amounts of data, the performance isn't a problem. With a large multidimensional dataset however, this would not be the case. Even with a heavily optimized implementation, the method is much slower than the EM training procedure.

4.2.1.3 Simple Histogram Approximation

Given all the complexity of the Bayesian GMM training procedure, its worth should be justified against simpler procedures. The GMM trained with EM provides one such benchmark. Here one more method is proposed which is even simpler. Here, a histogram

of time differences is taken, with a bin size equal to the resolution at which optimization will be performed (for most of these experiments 40 frames, which at 10 fps gives an accuracy of ±2 seconds). This distribution is then convolved with a Gaussian filter to smooth the effects of overfitting from limited data. There is just one parameter in this strategy, σ, which is the variance of the Gaussian filter. Experiments were performed with three different values of σ, $\sigma \in \{10, 50, 200\}$ to give a reasonable range on the level of blur applied. The notable difference with this versus the methods previously discussed in this section is that the distribution learned is non-parametric. For some message passing algorithms, this would be unacceptable since it is required that local probabilities are of a standard parametric form in order to facilitate fast inference. However all the experiments in this thesis involve Belief Propagation over a discrete domain, therefore the pairwise distributions can be arbitrary with no loss of performance.

4.2.1.4 Evaluation of Methods for Learning Temporal Distributions

To evaluate the various forms of distribution, leave-one-out training and testing is performed over the 37 turnarounds. Various measures of success are studied for a number of event pairs which form a representative subset of the $M(M+1)/2$ event pairs in the dataset. The first measure of interest is the overall likelihood of the training data with respect to the model, $p(\mathbf{X}|\theta) = \prod_i^N p(x_i|\theta_{-i})$, where $\mathbf{X} = \{x_1, .., x_N\}$ is the set of all time differences observed between two particular event types in the dataset and θ_{-i} represents the distribution trained on $\mathbf{X}_{-i}$, which omits point x_i. By itself, however this measure may be problematic since it can be dominated by one or two anomalous points. A more useful measure could be the median value of $p(x_i|\theta_{-i})$. These two statistics are extracted for several different distributions across several event pairs and the results are presented in Table 4.1. Note that the median and the overall likelihood do not necessarily agree.

Event Pair	$H_{\sigma=10}$		$H_{\sigma=50}$		$H_{\sigma=200}$		DP_{data}		DP_{broad}		$ML_{k=2}$		$ML_{k=6}$	
	Sum	Md	Sum	Md	Sum	Md	Sum	Md	Sum	Md	Sum	Md	Sum	Md
Air Arr ⇔ GPU Pos	-232	**−5.5**	**-224**	−5.8	-239	−6.4	-225	−5.8	-245	−6.6	-225	−5.6	-227	−5.6
Air Arr ⇔ Handler Chocks	$-\infty$	**−5.5**	-403	−5.6	-270	−6.4	**-250**	−5.6	-261	−6.3	-297	−5.7	-393	−5.5
PBB Pos ⇔ PBB Park	-1013	−6.8	-286	**−6.7**	-263	−6.8	**-261**	**−6.7**	-266	−7.0	-262	**−6.7**	-266	−6.8
Air Dep ⇔ PBB Park	-159	**−4.0**	-183	−4.9	-230	−6.2	**-156**	**−4.0**	-241	−6.6	-162	−4.1	-162	−4.2
GPU Pos ⇔ VRAC Load	-634	**−6.4**	-253	−6.8	-246	−6.8	**-244**	−6.6	-250	−6.9	-246	−6.6	-258	−6.7
GPU Pos ⇔ VRAC Unl	-413	−5.8	-157	−5.7	-160	−6.3	**-151**	−5.7	-169	−6.8	-183	**−5.6**	-166	**−5.6**
Air Arr ⇔ Cont Frnt Ld	-305	**−6.3**	-94	−6.6	**-91**	−6.8	**-91**	−6.5	-95	−7.2	-116	−6.6	-121	−6.7
Air Arr ⇔ Air Dep	-955	−7.0	-281	**−6.7**	-262	−6.8	**-259**	**−6.7**	-264	−6.9	-260	−6.8	-264	−6.8

Table 4.1: Evaluating several schemes for learning the distribution over time differences between event pairs. $H_{\sigma=s}$ represents histogram convolved with a Gaussian of $\sigma = s$. DP_{data} represents the GMM trained under Rasmussen's method, with Dirichlet priors and data-driven priors over the hyperparameters. DP_{broad} is as DP_{data} but with the prior over the hyperparameters broad and fixed. $ML_{k=i}$ represents GMM trained according to Maximum Likelihood using EM, with i components. Numbers given are 'Sum', which corresponds to $\sum_i^N log(p(x_i|\theta_{-i}))$ and 'Md', which is the median of $log(p(x_i|\theta_{-i}))$. Clearly the histogram with minimal smoothing performs best in terms of median score, though it is almost always very bad in overall terms since it does not generalize to all cases. The DP_{data} model performs well in almost all cases on both measures. The ML models are not far behind, but do appear to generalize slightly less well.

In fact, there is not a single simple best measure of the suitability of a distribution for these purposes, rather it is strongly dependant on the nature of the output from the independent event detectors for the event types involved. For example, if two independant event detectors were to work with 90% precision and recall, a temporal model with a very high median likelihood wouldn't necessarily be expected to boost performance if for 10% of examples its predictions were very bad. Conversely, if the performance of the detectors was much lower, then a model with high overall likelihood, but a low median (corresponding to a vague, high-entropy distribution) would be likely to only give modest improvement, whilst one with a high median but low overall likelihood could be more informative. The following test was thus devised. When evaluating the distribution θ, between events of type A and B, the focus is instead on how θ might boost the performance of detection of event A given the correct localization of event B. This is achieved by iterating over the instances of event A in the dataset, so for $i \in 1,..,N_A$ the observation likelihood $p(X|(t_i,d_i),c_i = A)$, for event A at all possible $t_i,d_i \in \mathcal{T}$ is maximized with respect to d_i, resulting in a 1D distribution over possible midpoints. $\phi_{c_i,c_j}(t_i,t_j = GT)$ (trained on all datapoints except those from turnaround containing i) is treated as the prior and multiplied with the likelihood function over all $t_i \in \mathcal{T}$. The resulting posterior is then thresholded in the same manner as the likelihood function and the results compared based on results obtained at the target recall level for each event as defined in Section 3.2.3. The difference in precision of the likelihood and posterior at the target recall rate is termed as the Potential Performance Improvement (PPI) associated with that temporal relationship. The precision-recall curves for a number of events, along with plots of the distributions are presented in Figures 4.6, 4.7, 4.8. The EER figures are presented for these events in Figure 4.9 as this marks the target recall selected for the Co-Friend project. The AP is also included for comparison. With the smoothed histogram distributions, the distribution with the greatest degree of smoothing proves to be the most effective in the first case, meaning the cases with a smaller degree of smoothing are overfitted. In the second case,

the order is reversed, with the broad smoothing not making the most of the low entropy distribution. This trend is mirrored in the Bayesian GMMs trained with Rasmussen's method. The data prior permits a slight overfit in the first case but is better calibrated to capture the lower entropy distributions in the second and third cases. The maximum likelihood trained GMMs perform well in the first two cases but seem to overfit in the final case. The single Gaussian performs reasonably well in all cases, though it is never the best performing distribution. On the basis of these results, it is clear that no one method for estimating the distribution is best in all cases. Even giving a thorough Bayesian treatment to training of the GMM doesn't entirely remove the dependence on an appropriate prior when data is scarce. Therefore the proposed solution is to use leave-one-out training and evaluation of several possible distribution types as above, selecting the best one in terms of Potential Performance Improvement for use in the final model.

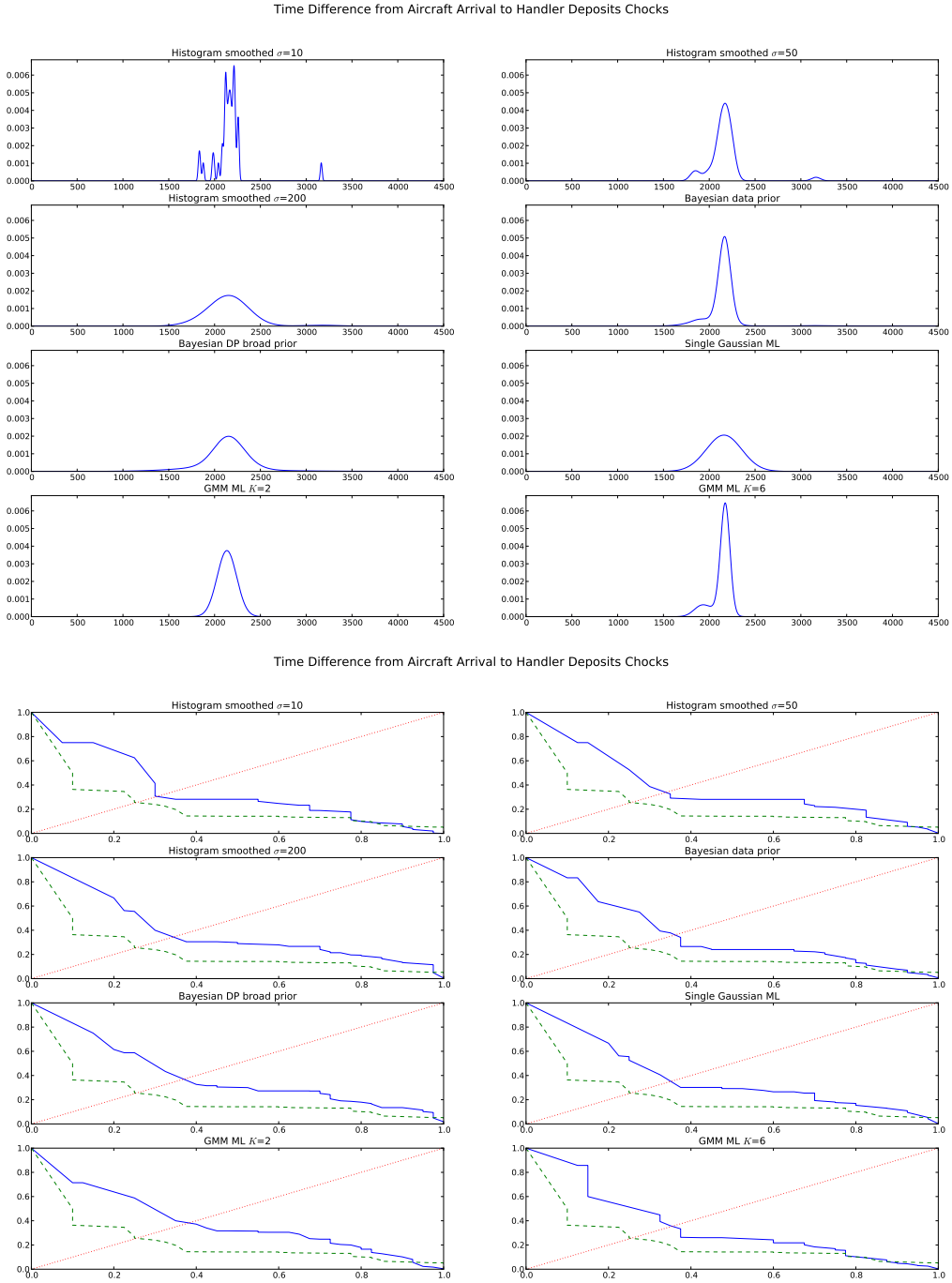


Figure 4.6: Top: Distributions learnt over time difference from Aircraft Arrival to Handler Deposits Chocks. Note that the histogram with minimal smoothing in the top left gives the clearest indication of the actual distribution of the training data. Bottom: Corresponding precision-recall curves when the posterior probability of Handler Deposits Chocks given correct localization of Aircraft Arrival, with different distributions as the prior. The solid blue line is the posterior and the dotted green line is the likelihood. This detector is the weakest in the Co-Friend dataset and therefore stands to gain most from a temporal relationship, with all pairwise distributions offering significant improvements. The only exception is the drop-off of performance at 100% recall for all the temporal models apart from the single Gaussian due to a single anomalous instance at a time difference of 3200. The posterior performs better than the likelihood across all distributions in terms of Average Precision (area under curve) and Equal Error Rate (point of intersection with dotted red line).

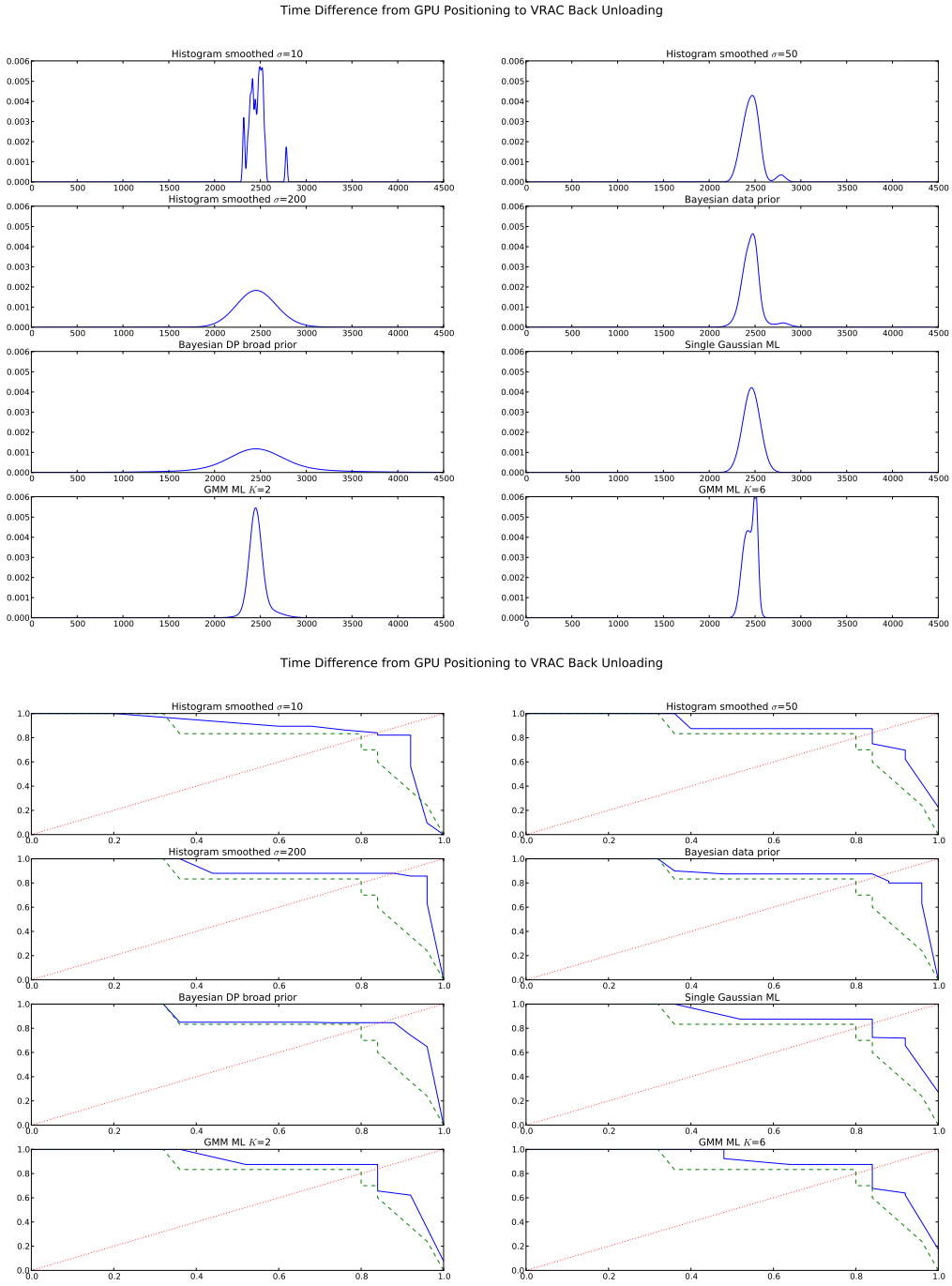


Figure 4.7: Top: Distributions learnt over time difference from GPU Positioning to VRAC Back Unloading. Bottom: Corresponding precision-recall graphs.

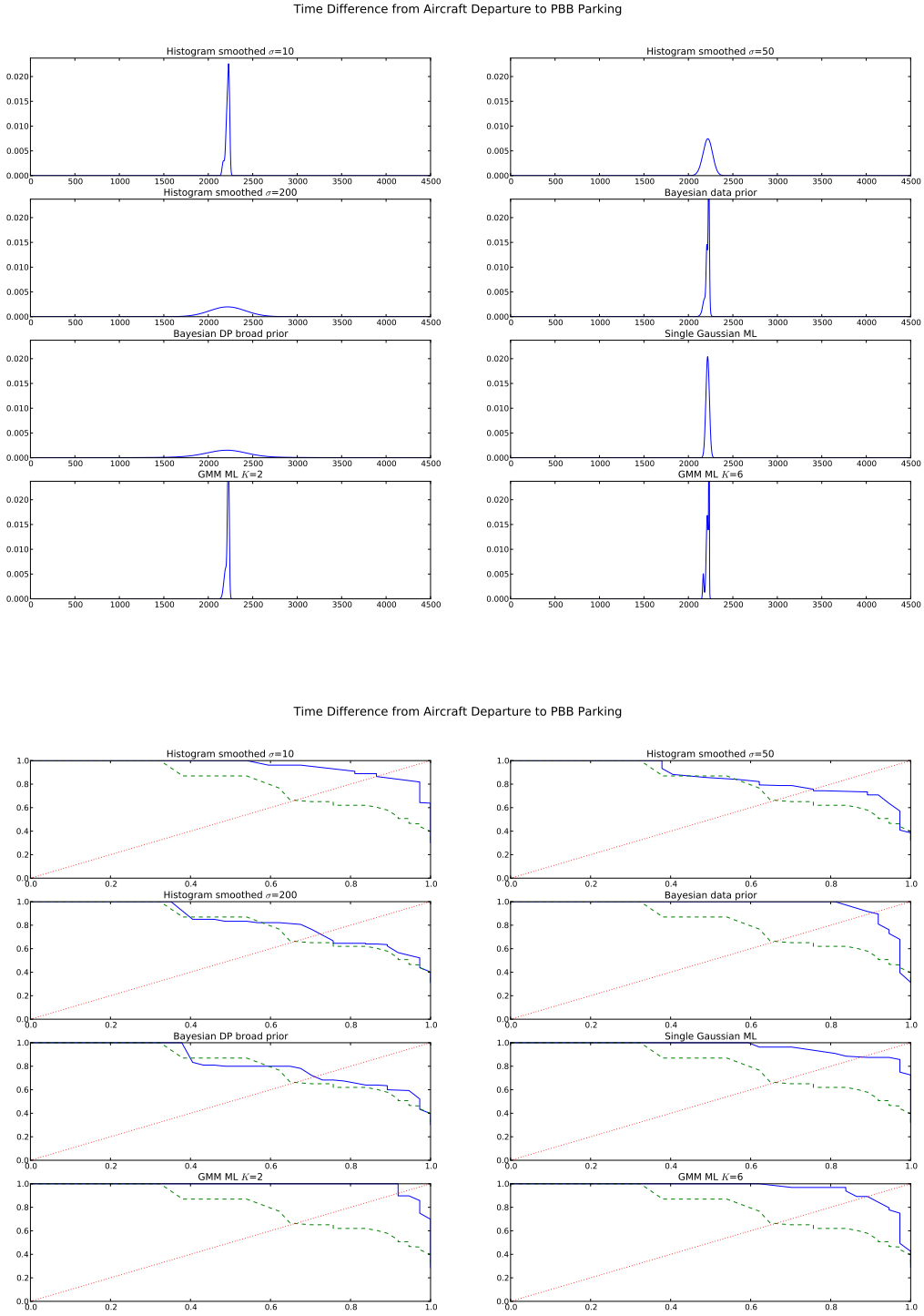
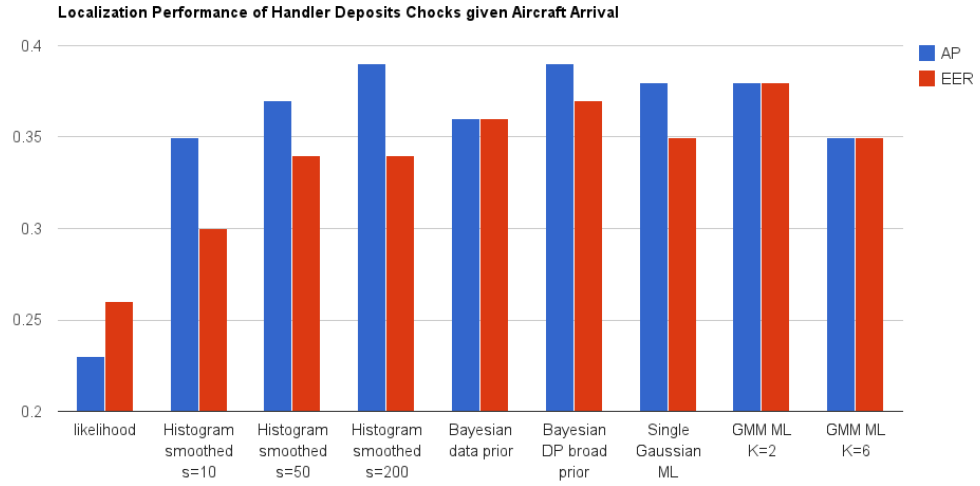
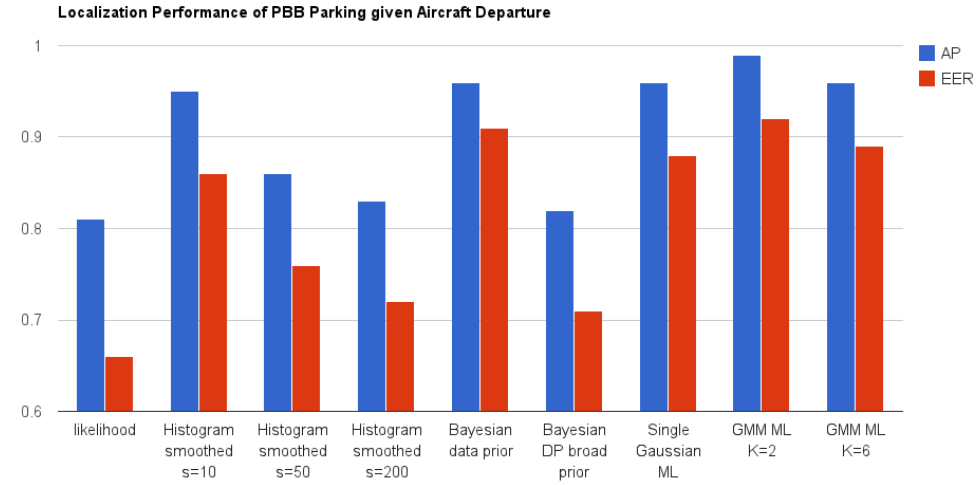


Figure 4.8: Top: Distributions learnt over time difference from Aircraft Departure to Passenger Boarding Bridge Parking. These two events are clearly strongly correlated, resulting in tightly-fitted distributions. Bottom: Corresponding precision-recall curves for posterior probabilities on PBB Parking given Aircraft Departure. Large gains in both AP and EER are evident among the tighter distributions.





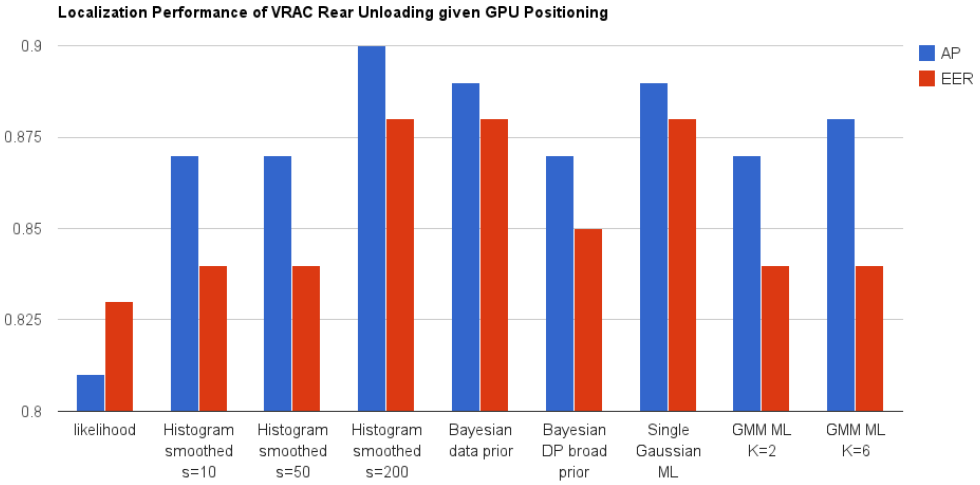


Figure 4.9: AP and EER figures for the distribution over time differences for several event pairs.

4.2.1.5 Joint Probability For *Null* States

Note that thus far, $\phi_{c_i,c_j}(t_i,t_j)$ has only been defined for cases where $t_i,t_j \in \mathcal{T}$. In the previous section *Null* states were introduced for the (t,d) variables. The cases where $t_i = Null$ and $t_j \in \mathcal{T}$ (or vice versa), or both $t_i,t_j = Null$ must also be considered. If either or both events are *Null*, clearly it is not possible to show any preference based on time differences. Therefore these cases should be of uniform probability. That uniform value is chosen to be T^{-2}, which is 1 over the size of the state space over which $\phi_{c_i,c_j}(t_i,t_j)$ for cases where $t_i,t_j \in \mathcal{T}$ was normalized. This ensures preference isn't shown to the *Null* states just because the number of states in which they feature is smaller than $\mathcal{T}^2$

4.2.1.6 Incorporating Co-occurrence Statistics

In a turnaround containing N events of various types, the number of instances of a given event type, i, is denoted by $\mathbf{N}_i = \{N_{i,0},..,N_{i,k},$ which takes a 1-of-$(K_{c_i}+1)$ representation as a binary $K_{c_i}+1$ dimensional vector. i.e.

$$N_{i,k} = \begin{cases} 1 & \text{for } k = \bar{n}_i, \\ 0 & \text{for } k \neq \bar{n}_i \end{cases} \quad ; \qquad \bar{n}_i = \sum_{n=1}^{N} \mathbf{1}_{c_n=i} \tag{4.9}$$

where $\mathbf{1}$ is the indicator function. If the joint distribution $p(\mathbf{N_i},\mathbf{N_j})$ could be learnt for all event type pairs $(i,j) \in \{1,..,M\}^2$, then this could be referenced and combined with $\phi_{c_i,c_j}(t_i,t_j)$ to give $p(t_i,t_j|c_i,c_j)$ as follows:

$$p(t_i,t_j|c_i,c_j) = \begin{cases} \phi_{c_i,c_j}(t_i,t_j)\sum_{k=1}^{K}\sum_{l=1}^{K} p(N_{i,k}=1,N_{j,l}=1) & t_i,t_j \in \mathcal{T} \\ \phi_{c_i,c_j}(t_i,t_j)p(N_{i,0}=1,N_{j,0}=1) & t_i = Null, t_j = Null \\ \phi_{c_i,c_j}(t_i,t_j)\sum_{k=1}^{K} p(N_{i,0}=1,N_{j,k}=1) & t_i = Null, t_j \in \mathcal{T} \\ \phi_{c_i,c_j}(t_i,t_j)\sum_{k=1}^{K} p(N_{i,k}=1,N_{j,0}=1) & t_j = Null, t_i \in \mathcal{T} \end{cases} \tag{4.10}$$

To estimate this probability from the training set, a number of options could be considered. Firstly, one might consider assuming independence of $\mathbf{N_i}$ and $\mathbf{N_j}$, so that $p(\mathbf{N_i}, \mathbf{N_j}) = p(\mathbf{N_i})p(\mathbf{N_j})$. Learning two univariate distributions requires less data then learning a bivariate distribution (since state space is $2(K+1)$ as opposed to $(K+1)^2$. However, this assumption would almost certainly result in loss of useful information. For the temporal prior to be useful, the timings of events need to be correlated to some degree. It is difficult to think of many scenarios where timings of two events are correlated yet the presence of one event doest not influence the probability of the other occurring. In the Co-Friend dataset, the maximum number of occurrences of any event is 4 and many events only occur once per turnaround. This means that it is therefore feasible to learn a categoric distribution over the joint state space for $p(\mathbf{N_i}, \mathbf{N_j})$. Maximum Likelihood training of a categoric distribution is done by simply counting the number of occurrences in each possible state; then dividing by the number of samples to give a normalized distribution. In domains where the number of instances in each sequence is less constrained, it might be necessary to impose approprite Dirichlet priors (though this would require a number of hyper parameters since it is not likely that a symmetric Dirichlet prior would be suitable) to reduce the danger of overfit. Another alternative would be to switch from learning a categoric distribution to learning parameters for a Poisson distribution. The Poisson distribution expresses the probability of a given number of events occurring in a fixed interval of time if these events occur with a known average rate and independently of the time since the last event. There are numerous examples of multivariate Possion distributions being applied in the literature (see for example [38]) and whilst exact parameterizations differ, the central idea is the same; there exists a covariance term as well as independent occurrence rates which captures the dependence between the dimensions.

4.2.2 Determining The Structure Of The MRF Prior

The second component of the temporal prior which must be learnt is the edge set, E, of the MRF. The procedure used is similar to that in Pictoral Structure Models [27]. An $N \times N$ matrix, $\mathbf{I}$, is constructed where $I(i, j)$ contains a measure inversely related to the usefulness of the link between i and j. Kruskal's algorithm [41] is then applied to $\mathbf{I}$ to obtain the desired minimum spanning tree. The potential weakness in this method is that the 'quality' of each relationship is measured by a value equivalent to the 'Sum' term that was studied in 4.1 and proven on a pairwise basis to be an unreliable indicator of the most useful distribution in terms of improving final performance of the system. Two alternatives are suggested here. The first, simplest method is to use the potential performance improvement at a particular sensitivity (for example the Equal Error Rate) in the manner used to produce the numbers in Figure 4.9.

The second option is to make the structure assignment dynamic (i.e. decided at recognition time), permitting the formulation of a data-dependent measure with which to rank potential edges in the MRF. As the prior is required to be a tree-structured MRF, and the variables within Y are discrete, the MAP solution to Equation 4.2 may be obtained directly through the max-sum Belief Propagation (BP) algorithm [6]. Simplifying the prior to a tree structure will result in the loss of some information so it is important that the edges retained are the most informative. BP is a message passing algorithm and in this context, it is reasonable to assume the most useful messages will be those of lowest entropy. Therefore, to determine the structure of the tree, the informativeness of each pairwise connection is assessed with the following equation.

$$I(i, j) = H[p(X|(t_i, d_i), c_i, \theta) \sum_{t_j} p(t_i, t_j|c_i, c_j, \theta)] \tag{4.11}$$

which is equivalent to the entropy of the initial message that would be passed from t_i to t_j (shown in figure 4.10). Note that this measure factors in the entropy of the detectors'

responses over the sequence, meaning event instances with noisy observation likelihoods over a particular sequence are less likely to be connected to one another. Calculating I for each pair of event instances gives an $N \times N$ matrix, which is made symmetric by setting $I(i,j) \rightarrow \max(I(i,j), I(j,i))$.

All three connection strategies described here are evaluated against one another in Section 4.4

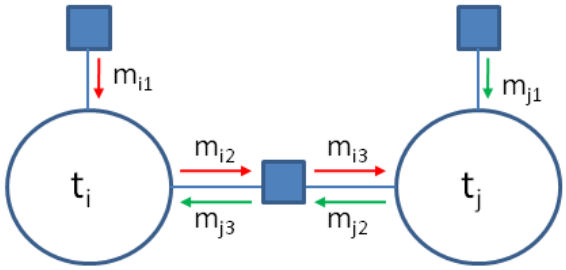


Figure 4.10: Depiction of the message passing scenario between two temporal midpoint random variables. The usefulness of the link between i and j is defined to be inversely proportional to the lower of the two entropies $H[m_{j3}]$,$H[m_{i3}]$.

4.2.3 Scenario Recognition with Multiple Instances

In Section 4.1 the idea of a *Null* state was introduced as a convenient way of dealing with the prior uncertainty over whether a particular event occurs within a sequence. Here a method is described for coping with the scenario recognition problem in its full generality: where multiple instances of the same event class may occur.

One obvious solution might be to set a hard upper limit on the number of instances of each particular class which could occur within a scenario, then to initialize the appropriate random variables before simply finding the MAP solution as in the single instance case. There is a problem with this approach since the observation likelihood for repeat instances of the same event class is taken from the same detector. For this reason, it is required that instances of the same event class be non-overlapping so that multiple instances don't

point to the same activity. However, this is an impossible constraint to impose with a tree structured prior (as demonstrated in Figure 4.11).

Another option would be to treat the number of instances of each event type, $\mathbf{N}_i$, as random variables and formulate appropriate priors and likelihood functions for them. There is no way this could be achieved whilst retaining the tree structure of the model since the likelihood function for $\mathbf{N}_i$ would have to depend on all instances where $c_j = i$. Losing the tree structure would mean loss of speed and guarantees of globally optimal solutions. A sampling approach would most probably be required which would be prohibitively expensive given that the observation likelihood probabilities taken from the independent event detectors are not of any convenient parameteric form.

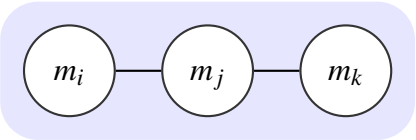


Figure 4.11: Three instances of event m. Clearly with more than 2 instances of m it is impossible for them to be fully connected in a tree structure. Here the instances are shown connected in a chain, which is insufficient since i and k are independent of one another given j, meaning the prior will not prevent them taking the same value.

To solve the problem, a greedy procedure was formulated which is fast and allows the independence constraints to be circumvented. First exact optimization of Equation 4.2 is performed, allowing exactly one instance of each event class. For events which had instances localized successfully ($(t_i, d_i) \neq Null$), additional instances are created for the next stage of optimization. To prevent overlap, the previously localized events are fixed by transforming their observation likelihood function to a Dirac delta function at the point at which they were localized in the previous stage of optimization and set the observation likelihood of the new instance to zero at all (t_i, d_i) which would overlap the previously localized instance. The tree-shaped prior is then restructured to include the new nodes, and optimization is performed again. The process is repeated until there is a *Null* instance for each event class.

Recall that in Equation 4.10, the joint distribution over the number of instances $p(\mathbf{N}_{c_i}, \mathbf{N}_{c_j})$ was referenced in order to weight the probability of either t_i, t_j being 'Null'. The assumption in that equation was that instances i, j were the only potential instances of c_i, c_j in the turnaround. Clearly in the general case, this would not be the case and since all instances of each event type are not connected in the tree structured prior, it seems that the necessary information is not available to appropriately reference $p(\mathbf{N}_{c_i}, \mathbf{N}_{c_j})$. In fact, the iterative optimization procedure opens the possibility here to further work around the independence assumptions. An additional variable o_i, is defined for each instance which denotes the 'order' of the i^{th} instance. $o_i = 2$ would indicate that instance i is the 2^{nd} occurrence of class c_i. Because of the nature of the iterative optimization, it is reasonable to assume that if an instance has an order n then $n-1$ instances have already been successfully localized. Therefore Equation 4.10 can be modified to

$$p(t_i, t_j | c_i, c_j, o_i, o_j) = \begin{cases} \phi_{c_i,c_j}(t_i, t_j) \sum_{k=o_i}^{K} \sum_{l=o_j}^{K} p(N_{i,k} = 1, N_{j,l} = 1) & t_i, t_j \in \mathcal{T} \\ \phi_{c_i,c_j}(t_i, t_j) p(N_{i,0} = 1, N_{j,0} = 1) & t_i = Null, t_j = Null \\ \phi_{c_i,c_j}(t_i, t_j) \sum_{k=o_j}^{K} p(N_{i,0} = 1, N_{j,k} = 1) & t_i = Null, t_j \in \mathcal{T} \\ \phi_{c_i,c_j}(t_i, t_j) \sum_{k=o_i}^{K} p(N_{i,k} = 1, N_{j,0} = 1) & t_j = Null, t_i \in \mathcal{T} \end{cases} \tag{4.12}$$

It is acknowledged that there are limitations to this procedure. For example, if it was observed that there were always either zero or exactly n instances of a particular event type, this subtlety would be ignored since under this scheme, it is assumed that observing $n-1$ instances ($n > 1$) is at least as likely under the prior as observing n due to the sum from o_i to K. This deficiency is not considered to be too problematic since such relationships are likely to be relatively rare and in any case, the prior would not reduce the chance of detecting the n^{th} sample.

4.3 Summary of Method

This chapter has stepped through, in detail, several disparate elements. The entire pipeline is summarized here to give a clear picture of how it is applied.

- Learning
    1. Train independent event detectors for all events,
    2. Learn pairwise distributions over time differences using several different models,
    3. Decide on a target recall rate for detectors,
    4. Select most useful models based on potential performance improvement at chosen recall rate.
- Recognition
    1. Initialize scenario with single instance of each event type,
    2. Determine optimal tree structure,
    3. Maximize with respect to instance timings,
    4. Create additional instances for events that do not yet have a *Null* instance,
    5. Repeat steps 2-4 until there exists a *Null* instance for each event type.

4.4 Evaluation

The comparison of the independent detector with three variants of the temporal model can be found in Table 4.2.

There are three different sets of results in Table 4.2 to represent the three connection strategies covered in Section 4.2.2. 'Dynamic' represents the dynamic, entropy-based structure selection. 'PPI' uses the Potential Performance Improvement as the measure,

whilst 'PSM' is the quality measure used for Pictoral Structure Models in [27]. Comparing the three methods is not straightforward since a cost function was not provided for the Co-Friend dataset. An attempt was made to calibrate sensitivity to around the Equal Error Rate through a limited line search over the α_{c_i} variables from Equation 4.1 for each model. It was not possible however to get an exact match in all cases given the dataset size and interdependence of the α_{c_i} values. Therefore it is a case of trying to compare points in the 2D precision-recall space. Since precision-recall curves had been generated for the independent event detectors, it was decided that these should be used as benchmarks for performance. Given a precision, recall score for a given connection strategy, the performance improvement is calculated by taking the precision of the independent detector at the same level of recall. The models can be compared based on the performance improvement numbers, or by reference to Figure 4.12. Its clear that for almost all events, all variants of the temporal model perform significantly better than the independent detectors. Note that the events which cause most problems to all variants of the temporal model are the Container Rear Door Loading/Unloading events. There are a number of reasons for this. Firstly, these are the events for which there are fewest occurrences in the data, meaning the temporal relations learned are most likely to be overfitted. Secondly, it is unfortunate that the false positives for this event tend to be induced by activity which actually belongs to a Belt Loader Unloading event; and the two events tend to occur at similar times within the turnaround. Therefore most of the noise from the detector also satisfies the temporal model.

It's not possible to say conclusively that any of the connection strategies is best. The PPI-based connection strategy achieves the best results overall, but only just, and there are individual events on which it is beaten. This is likely to largely be an effect of a small dataset size. What is more informative than the numbers is actually an examination of the connection structures which arise from the different methods and the issues and advantages of each.

The most significant issue that the PPI and PSM models are both theoretically susceptible to is that an event instance may be highly connected (i.e. act as a hub linking several other nodes), despite its detector registering very little response for a sequence. This results in very uninformative messages being passed through this node, meaning the neighbours for whom that node is the sole neighbour are effectively disconnected. See Figure 4.13 for an example of this effect. The dynamic structure selection procedure protects against this, since the high entropy of the intial messages passed out of such a node would result in low scores; meaning the node is very unlikely to be selected for multiple connections. In the sequences on which PPI performs badly, this effect is often observed. Though there is nothing in the PSM connection strategy to prevent this effect, it can seldom be observed in the Co-Friend dataset. This is because the events which are least likely to occur in any given sequence are those with the fewest occurrences across the dataset and hence those for which the temporal relations are most often overfitted, giving them a low score under the PSM connection strategy. In a larger dataset, this synergy would not exist, making PSMs just as susceptible to the issue as PPI. A weakness of the dynamic structure selection procedure is that since it prefers low entropy messages, it is more likely to select the overfitted distributions (which tend to be strongly peaked). Similarly, the PPI measure is also more prone to overfit since it involves estimating the usefulness of links with an N-fold cross validation procedure which further reduces the amount of available data for training. From these observations, it would seem that PSM connection strategy is particularly useful in situations when there is a shortage of data, wheras the PPI and dynamic models are likely to be more powerful in larger datasets.

Event Class	N	Dynamic			PPI			PSM		
		rec (%)	pre (%)	imp (%)	rec (%)	pre (%)	imp (%)	rec (%)	pre (%)	imp (%)
Aircraft Arrival	37	100	100	51	100	100	51	100	100	51
Aircraft Departure	37	100	100	0	100	100	0	100	100	0
Passenger Bridge Positioning	37	95	95	56	100	100	66	100	100	66
Passenger Bridge Parking	37	100	100	53	97	97	38	97	97	38
Ground Power Unit Positioning	37	86	86	4	86	86	4	86	86	4
Handler Deposits Chocks	40	55	59	34	52	57	31	55	59	34
Container Rear Door Loading	13	67	18	-20	56	54	-2	70	18	-17
Container Rear Door Unloading	14	80	22	-29	20	31	-69	68	16	-44
Container Front Door Loading	27	54	47	-31	77	77	14	92	57	0
Container Front Door Unloading	25	79	69	12	43	100	31	71	62	0
Belt Loader Loading	36	74	52	18	60	60	14	60	70	24
Belt Loader Unloading	25	80	65	-19	84	75	5	72	86	2
Mean Improvement				18			22			20

Table 4.2: Numbers of occurrences of each event class in dataset and performance of detection techniques.

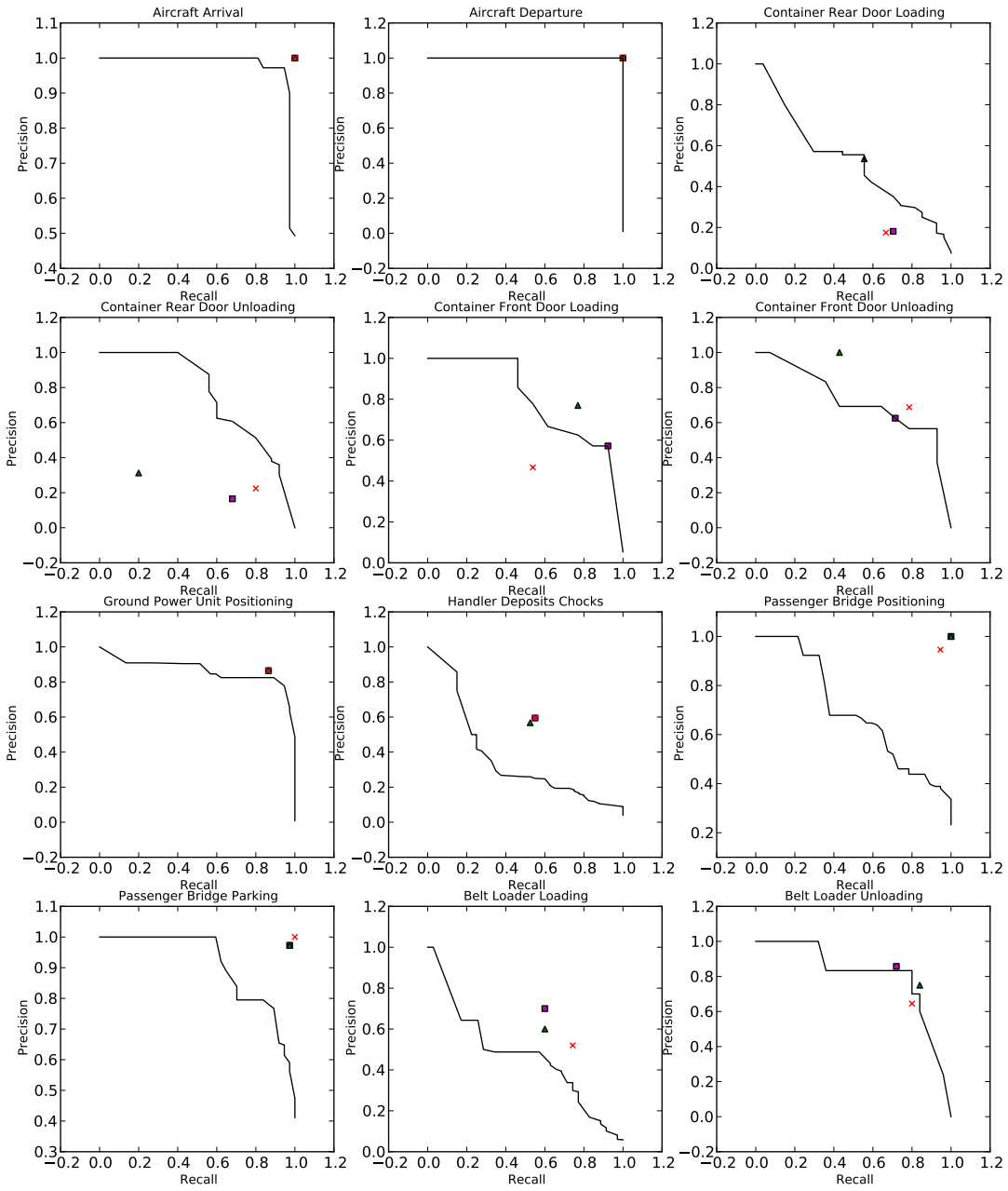


Figure 4.12: Performance of Temporal Structure Models across event types. The precision recall curves plotted are the performance of the independant detector for each event. The performance of 3 different variants of the temporal structure model based on 3 different connection strategies are plotted as single points. The square represents Pictoral Structure Model style connection, the triangle represents connection based on Potential Performance Improvement and the cross represents dynamic connection. Note that interdependence between events makes it impossible to generate precision-recall curves using Temporal Structure Models.

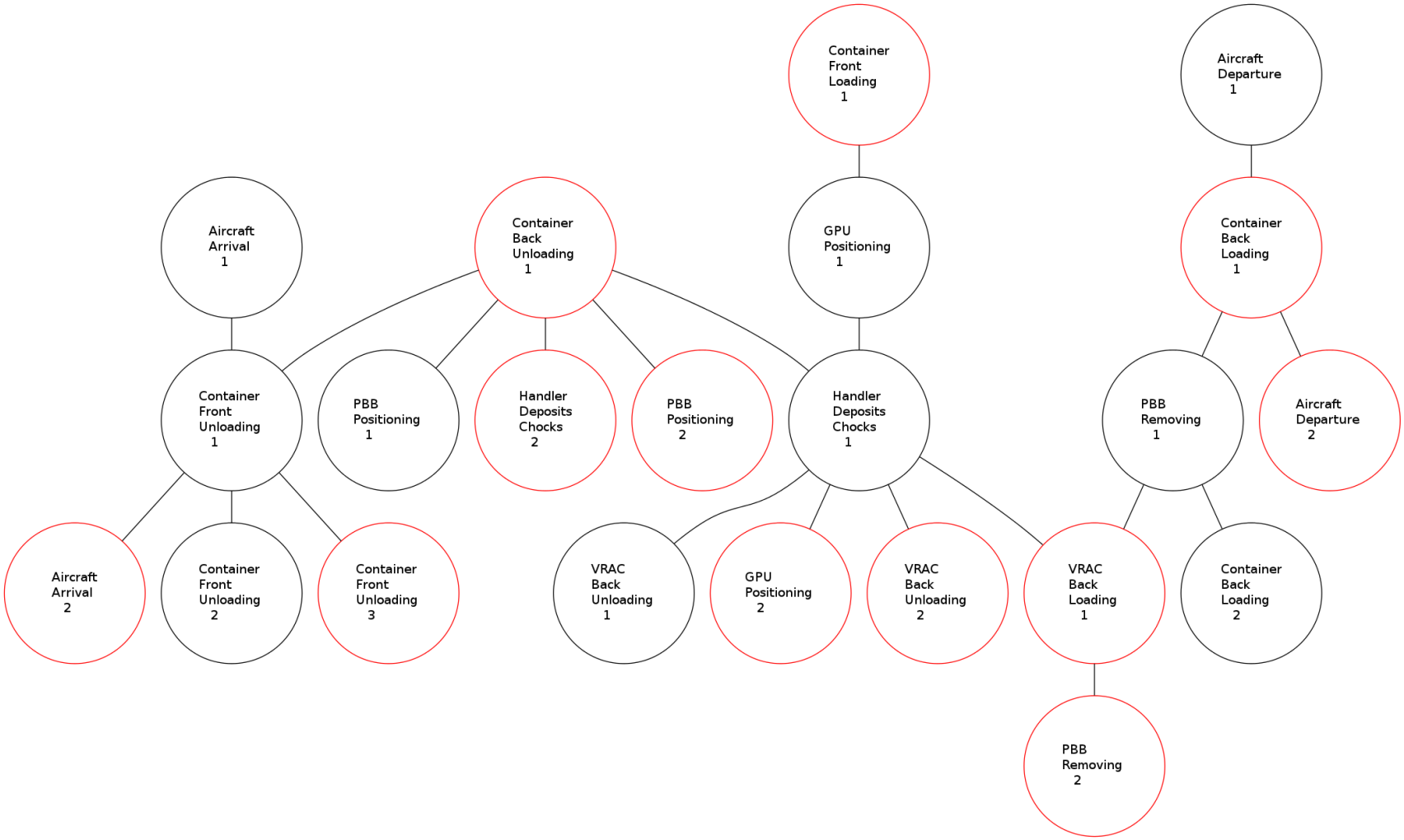


Figure 4.13: Graph showing connectivity of the final state of the MRF prior after optimization of turnaround COF-7. Red circles indicate *Null* event instances. The number beneath the event type is the order of that node i.e. Aircraft Arrival 2 is the 2^{nd} occurrence of Aircraft Arrival, meaning that only one instance of Aircraft Arrival was detected.

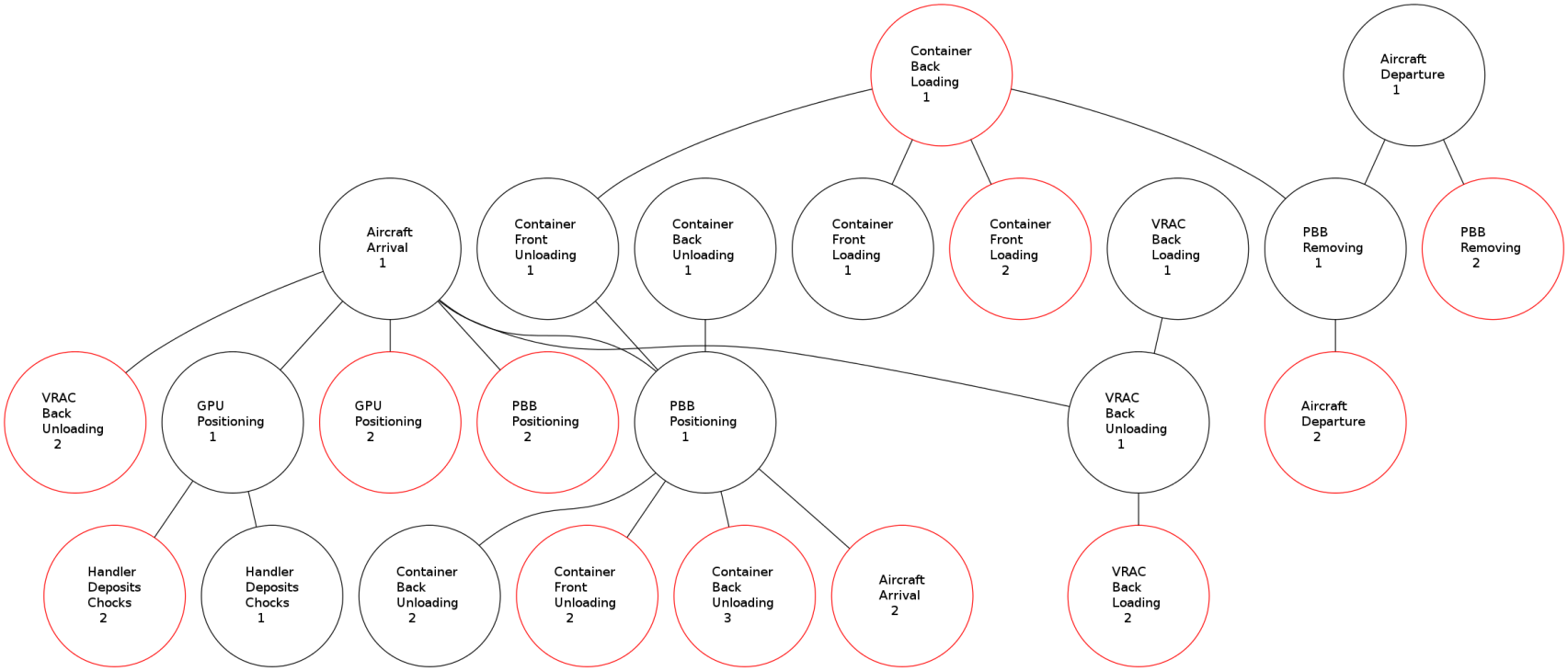


Figure 4.14: Graph showing connectivity of the final state of the MRF prior after optimization of turnaround COF-7 with PSM structure selection. Red circles indicate *Null* event instances. The number beneath the event type is the order of that node i.e. Aircraft Arrival 2 is the 2^{nd} occurrence of Aircraft Arrival and as it is red, the instance is null, meaning that only one instance of Aircraft Arrival was detected.

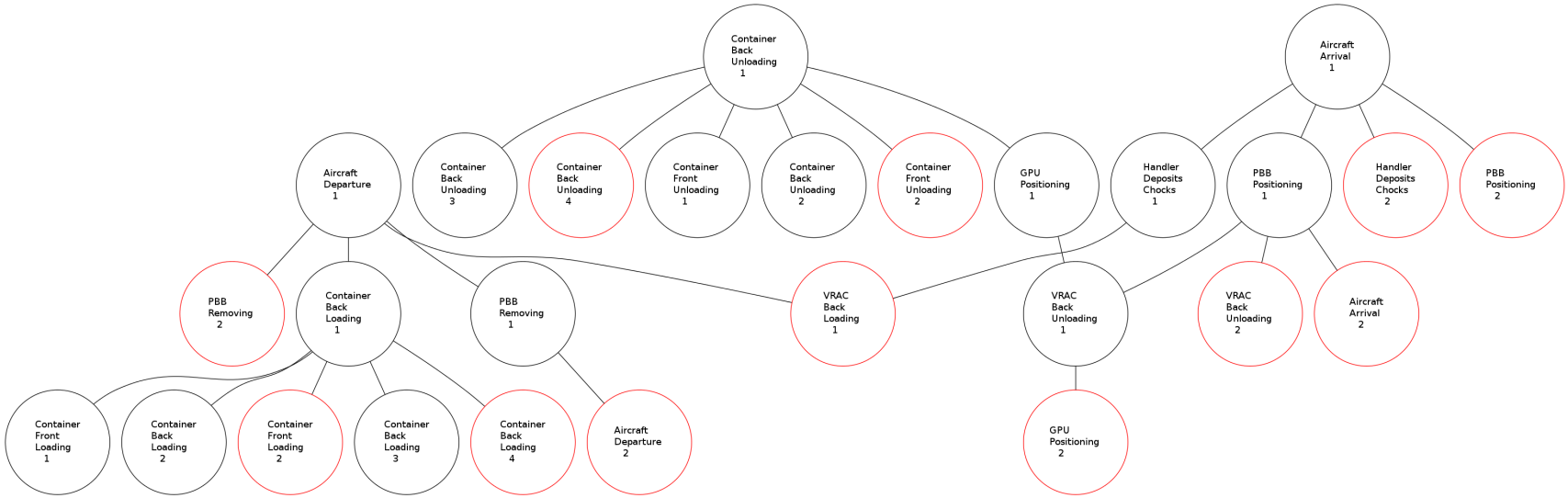


Figure 4.15: Graph showing connectivity of the final state of the MRF prior after optimization of turnaround COF-7 with Dynamic structure selection. Red circles indicate *Null* event instances. The number beneath the event type is the order of that node i.e. Aircraft Arrival 2 is the 2^{nd} occurrence of Aircraft Arrival and as it is red, the instance is null, meaning that only one instance of Aircraft Arrival was detected.

In this chapter, a novel method for exploiting temporal structure in scenarios to significantly improve over the performance of independent event detectors was introduced. The method is flexible; allowing event detectors to be trained independently and new/different detectors to be 'plugged in' as the observation likelihood term. Several methods for learning pairwise temporal relations between different event types were suggested and a new metric introduced for evaluating distributions learned to represent these relationships, which reflects the context in which they are to be deployed. An efficient algorithm for recognition was presented which can incorporate a novel, dynamic method of structure learning.

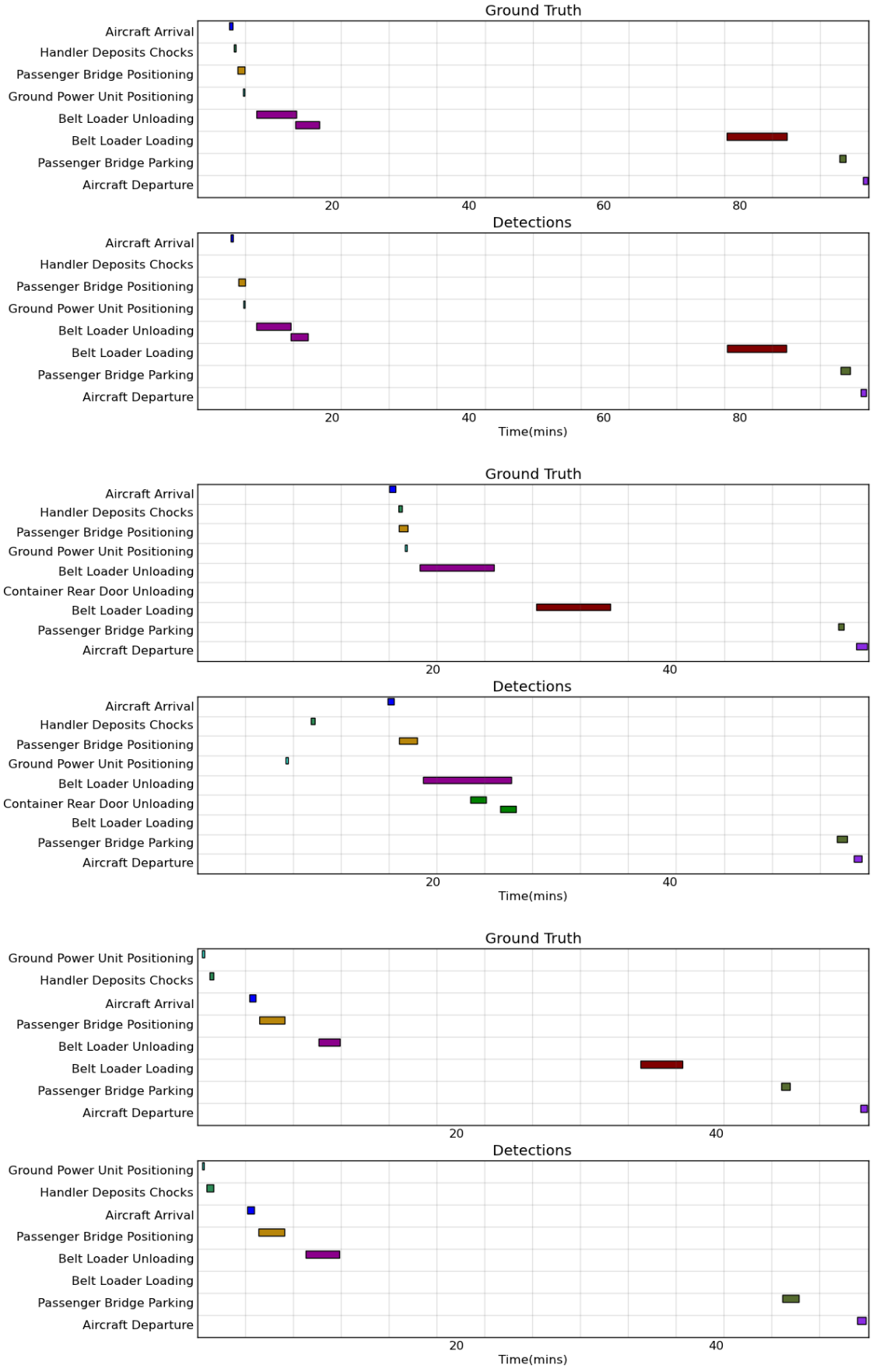


Figure 4.16: Gantt charts showing detection results against Ground Truth for a number of turnarounds.

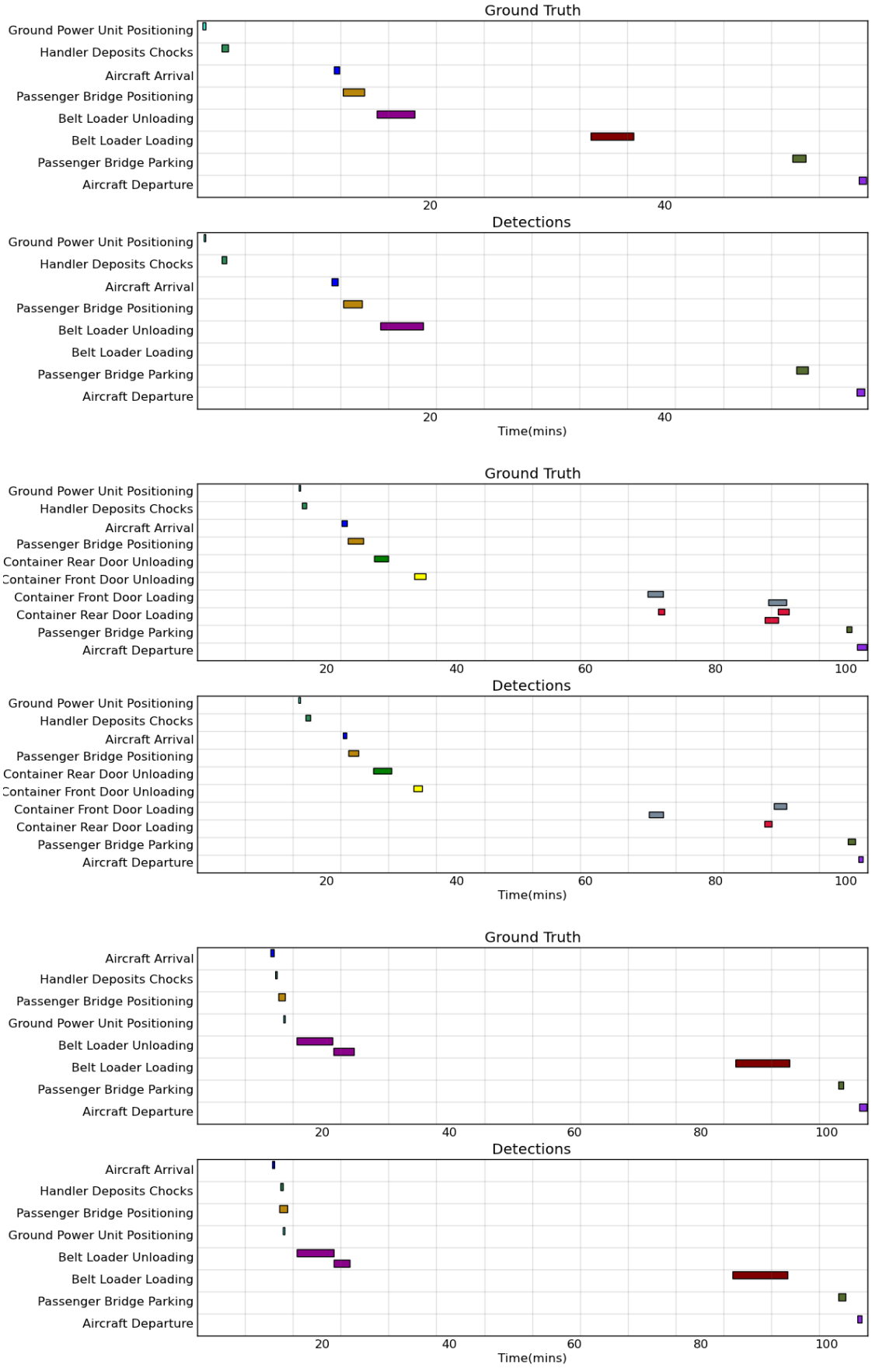


Figure 4.17: Gantt charts showing detection results against Ground Truth for a number of turnarounds.

Chapter 5

Imposing Temporal Constraints With Conditional Random Fields

In the previous chapter, the output of independently trained event detectors was given a direct probabilistic interpretation by treating them as $p(X|C,Y,\theta)$. A potential problem with this approach is that in treating the outputs of these heterogeneous detectors as genuine probabilities, all are consequently weighted equally in the global reasoning over the scenario. It is desired that no assumptions should be made over the type of classifier to be used in the framework. In this thesis, the SVM has been favoured, the output of which is not directly probabilistic, rather the distance from the decision plane is returned. The method of Platt [61] is used to map to a probability by means of a sigmoid function; the parameters of which are also determined by the data. For the model in the previous chapter to function well, with detections for all events around the Equal Error Rate, the probabilities from the independent event detectors must be well-calibrated. In the probabilistic formulation there is no facility to weight the influence of the temporal prior relative to the observations, which might be disadvantageous. By treating the individual classifier probabilities and learnt temporal relations as feature functions within a Conditional Random Field, this limitation can be addressed. This chapter describes how the model described

in the previous chapter may be reformulated as a CRF.

5.1 General Conditional Random Field Formulation

The majority of the literature on CRFs focuses on the special case of linear chain CRFs [43]. This is due in part to the model's strong simliarity to the HMM and its easy applicability to sequential data. Of interest here however, are those of a more general tree structure as deployed in the MRF prior in the previous chapter. The notation used in this chapter mirrors that used in Sutton and McCallum's [75] CRF tutorial.

A Conditional Random Field is the name for a very broad class of discriminative model, typically parameterized as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{\psi_A \in G} \exp\left\{ \sum_{k=1}^{K(A)} \lambda_{Ak} f_{Ak}(\mathbf{y}_A, \mathbf{x}_A) \right\}, \tag{5.1}$$

where G is a factor graph over Y, and $p(\mathbf{y}|\mathbf{x})$ factorizes according to G. $F = \{\psi_A\}$ is the set of factors in G, and each factor takes the exponential family form. The motivation for using this log-linear functional form has origins in information theory, as explained eloquently in [4]. In brief, the idea is that given some statistics about a set of data, the best model to fit that data is the one which fits these statistics with maximum entropy. This form allows multiple feature functions and multiple weighting parameters for every clique within G. In practical applications, one often sees extensive parameter tying. For example, the weights used for factors $\psi_t(y_t, y_{t-1}, \mathbf{x})$ are the same for each timestep in a linear chain CRF. In more general cases, parameter tying can be formalized through *clique templates*. Each clique template $C_p \in C$ is a set of factors which has a corresponding set of sufficient statistics $\{f_{pk}(\mathbf{x}_p, \mathbf{y}_p)\}$ and parameters $\theta_p \in \mathbb{R}^{K(A)}$. Thus the CRF can be written

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in C} \prod_{\psi_c \in C_p} \psi_c(\mathbf{x}, \mathbf{y}; \theta_p), \tag{5.2}$$

where each factor is parameterized as

$$\psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p) = \exp\left\{\sum_{k=1}^{K(p)} \lambda_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c)\right\}, \tag{5.3}$$

and the normalization function is

$$Z(x) = \sum_{\mathbf{y}} \prod_{C_p \in C} \prod_{\psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p). \tag{5.4}$$

Note that the normalization function, Z, requires summing over the entire state space of $\mathbf{t}$, a space of size $(T+1)^N$, i.e. exponential in the number of event instances. This makes CRFs intractible for general graphs and restricts their usage to small problems or restricted structural forms which allow calculation of Z in a more efficient manner. For this reason in the following section, a tree structure is once again used in the CRF formulation.

5.2 Tree-Structured Conditional Random Field Formulation

To formulate a CRF which is analagous to the model described in Chapter 4, clique templates are defined for each individual event type, consisting of a single function, g, (the output of the independent event detector) and a corresponding weighting parameter, λ. $\mathbf{t_i}$ represents all the t variables within clique i.

$$\psi_i = \exp\{\lambda_i g_i(\mathbf{t_i}, X)\}; \qquad i \in \{1, .., M\} \tag{5.5}$$

Next a clique template is defined for each event pair consisting of a single function

(the pairwise probability from the previous chapter) and corresponding weight

$$\psi_i = \exp\{\lambda_i f_i(\mathbf{t_i}, X)\}; \qquad i \in \{M+1, .., M+|\mathbb{E}|\} \tag{5.6}$$

where $\mathbb{E}$ is the set of all event pairs *for which a connection exists in the training set*. This reveals a limitation in the CRF formulation compared to the simpler model. In the previous chapter, the graph structure in each sequence was decided at optimisation time based on a usefulness measure of the pairwise relationships. This measure was permitted to be dynamic (as in the suggested entropy-based measure). Therefore, when approaching an unseen sequence, the structure of the graph would not be known until the detector output was observed. In most sequences Event A might be connected to Event B, yet in another sequence Event A might be connected to Event C since both Event A and Event B had very high entropy observation likelihood. This would not cause a problem, since training of pairwise relations was done entirely independently, so the relationship between A and C would have been trained even if it would not have been used in scenario-level optimization over the training set. In the CRF case, whilst the independently trained pairwise relations will be retained as functions, the weights relating to each pairwise function must be optimised at a scenario level. The structure of the training scenarios is therefore crucial to the CRF training. If one desired to use the dynamic structure assignment, a very large dataset would be required in order to guarantee that all pairwise relations would be observed in the training data.

5.2.1 CRF Training

Since it is assumed the clique functions have all been trained as described in Section 4.2.1, all that remains in training is to optimize the $M + |\mathbb{E}|$ weights. The CRF is assembled for training by iterating over all the sequences in the training set. For each event type, $m \in \{1, .., M\}$ in each each sequence $s \in \{1, .., S\}$, $N_{m,s} + 1$ random variables, t_i, with

corresponding event-class indicators, c_i are created, where $N_{m,s}$ is the number of occurrences of event m in sequence s. Note that the extra instance corresponds to the 'switched off' instance of each type which would result from a successful optimization using the method described in Section 4.2.3. Each t_i has a factor relating to its corresponding observation likelihood- all instances of the same type across all sequences share the same weighting parameter. The static PPI measure defined in Chapter 4 is used as before in order to determine the structure for each sequence. The sequences can be looked at as disjoint components within one large graph, G. The structure of G then determines how the pairwise clique templates are applied. All pairs connected in the minimum spanning tree have a factor linking them. Again, pairs of the same type have the same clique template and hence the same weight associated with their factor. This results in the following objective function.

$$p(\mathbf{t}|\mathbf{X}) = \frac{1}{Z}\prod_{i=1}^{N}\exp\left(\lambda_{c_i}g_{c_i}(t_i,X)\right)\prod_{(j,k)\in G}\exp\left(\mu_{c_j,c_k}f_{c_j,c_k}(t_j,t_k)\right) \tag{5.7}$$

where the double-subscripted parameter μ has been introduced to simplify notation. It may still be convenient to refer to the set of parameters as λ, which is possible if it is explained that there is a one to one mapping between $\mu_{c_j,c_k}; (c_j,c_k) \in \mathbb{E}$ and $\lambda_i; i \in \{M+1,..,M+|\mathbb{E}|\}$. Due to numerical precision issues, it is more convenient to work with the log-likelihood

$$\mathcal{L}(\lambda) = \sum_{i=1}^{N}\lambda_{c_i}g_{c_i}(t_i,X)\sum_{(j,k)\in G}\mu_{c_j,c_k}f_{c_j,c_k}(t_j,t_k) - \log Z \tag{5.8}$$

and its derivatives

$$\frac{\partial \mathcal{L}}{\partial \lambda_a} = \sum_{i=1}^{N} \mathbf{1}_{c_i=a} \lambda_{c_i} g_{c_i}(t_i, X) - \sum_{i=1}^{N} \sum_{t'_i} \mathbf{1}_{c_i=i} \lambda_{c_i} g_{c_i}(t'_i, X) p(t'_i|X); \qquad a \in \{1, .., M\} \tag{5.9}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_a} = \sum_{(j,k)\in G} \mu_{c_j,c_k} f_{c_j,c_k}(t_j, t_k) - \sum_{(j,k)\in G} \sum_{t'_j,t'_k} \mu_{c_j,c_k} f_{c_j,c_k}(t'_j, t'_k) p(t'_j, t'_k|X) \quad a \in \{M+1, .., M+|\mathbb{E}|\} \tag{5.10}$$

Note that the normalizing constant Z requires summing over all possible settings of the **t** variables. Since G is tree-structured, this computation can be done in linear rather than exponential time through the sum-product Belief Propagation algorithm. BP can also be used for efficient computation of the conditional probabilities used in the calculating the partial derivatives as described in Figure 5.1.

Since the log-likelihood function is concave with respect to the λ and it is possible to calculate the derviative of the log-likelihood function with respect to λ, it can be optimised with any gradient-based method. In [28], several flavours of Conjugate Gradient algorithm are compared with L-BFGS and Generalized Iterative Scaling. L-BFGS and preconditioned Conjugate Gradient algorithms come out on top of this comparison. Since an L-BFGS implementation is readily available in *Scipy*, this is the method deployed in this work.

As in all previous experiments with the Co-Friend dataset, a leave-one-out training procedure is followed. In an initial run, optimization failed to converge in most cases after the permitted 100 iterations (which took roughly 10 hours). Also the weights arising from across the leave-one-out procedure were quite variable, with lots of extreme values. This was a clear indication of overfitting, meaning regularization would be necessary.

Regularization is often used in CRF training, to prevent overfit and encourage sparseness in features. The most common heuristic is to introduce the following penalty to the

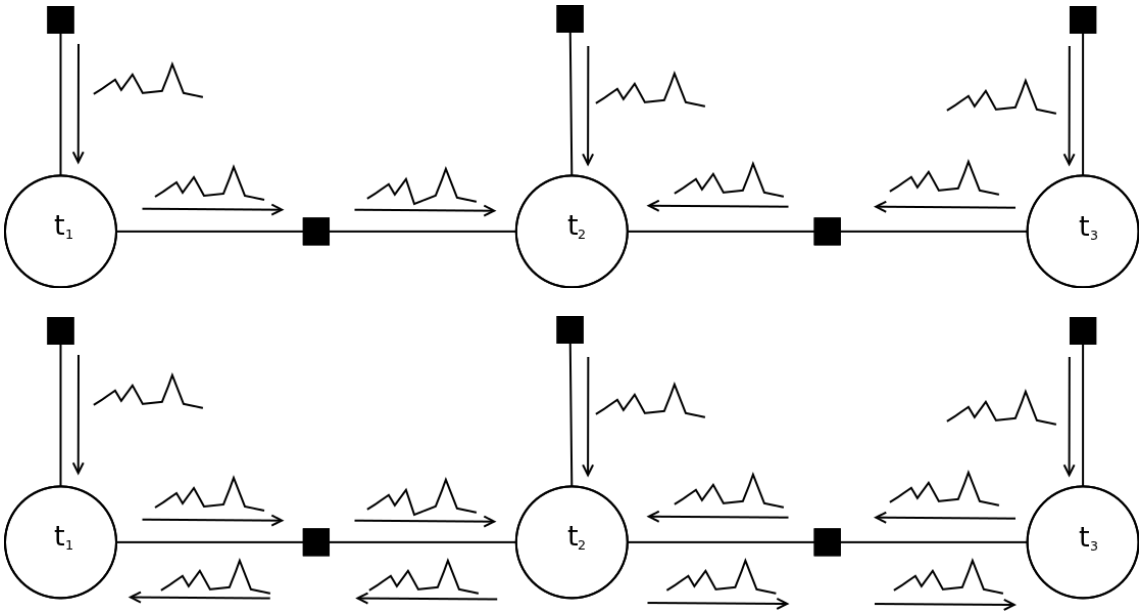


Figure 5.1: Illustration of the message-passing scenario for tree-structured CRF optimization. Top: For computation of the normalizing coefficient Z, messages are passed inwards from leaves into the root (in this case t_2). $\log Z$ is calculated at the root by summing all incoming messages then taking log sum of the exponential of that resultant vector (i.e. summing over t_2). For computation of conditional probabilities over other nodes, it is necessary for the root to transmit messages back out to the leaves. These messages can then be retained in memory and referenced when required. The conditional probability over any single t variable given all others is obtained by simply summing all the messages coming into that node. The joint conditional $p(t_j, t_k|\mathbf{t}_{\neg j,k}, \mathbf{X}, \lambda)$ over any pair of t variables is obtained by projecting the messages from t_j, t_k onto their adjoining factor.

log-likelihood function

$$-\sum_{i=1}^{M+|E|} \frac{\lambda_k^2}{2\sigma^2} \tag{5.11}$$

which corresponds to placing a Gaussian prior on the weight vector λ with zero mean and covariance $\sigma^2\mathbf{I}$. When $\frac{\partial \mathcal{L}}{\partial \lambda}$ is calculated for the updated Log likelihood function, the result is as before but with the additional penalty term $\frac{-\lambda_k}{\sigma^2}$. Optimization takes around 8 hours to complete, requiring on average 75 Iterations to converge. Most of this computation time can be attributed to the aproximately 1500 function evaluations and 700 gradient evaluations required. Several values of sigma were tried, with a value of 0.5 yielding the best results.

The weights assigned to observations from the various event detectors are displayed in Figure 5.2. The weights are generally larger on the detectors which peform bettter, with the exception of the Handler Deposits Chocks event, which recieves the highest weighting despite being one of the least reliable detectors. This is most likely because whilst the detector does fire the occasional false positive, these false positives are very sharply peaked and the detector is otherwise very well behaved. In several of the more effective detectors such as the Passenger Bridge Positioning, the event takes place over a much longer time period and any intervals which overlap the ground truth to a reasonable degree display some level response in the detector. Whilst this noise isn't problematic at detection time where one can apply local maxima constraints, it causes the detector to be downweighted in the CRF training, since training penalizes any response over negative intervals.

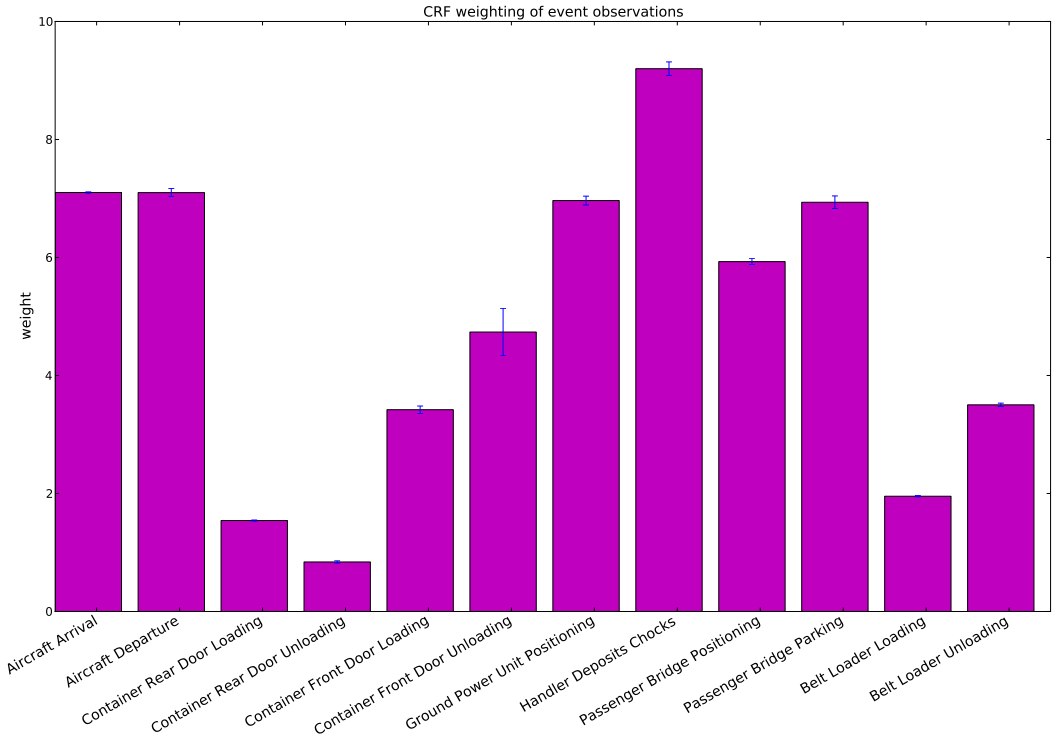


Figure 5.2: Weights obtained from CRF training. Error bars are generated by leave-one out training over 37 sequences. The small error bars indicate the stability of the parameters after the introduction of the regularization term.

Event Class	N	TSM			CRF		
		rec (%)	pre (%)	imp (%)	rec (%)	pre (%)	imp (%)
Aircraft Arrival	37	100	100	51	100	100	51
Aircraft Departure	37	100	100	0	100	100	0
Passenger Bridge Positioning	37	100	100	66	100	100	66
Passenger Bridge Parking	37	97	97	38	97	97	38
Ground Power Unit Positioning	37	86	86	4	86	85	3
Handler Deposits Chocks	40	52	57	31	57	63	38
Container Rear Door Loading	13	56	54	-2	56	59	3
Container Rear Door Unloading	14	20	31	-69	32	52	-5
Container Front Door Loading	27	77	77	14	69	73	8
Container Front Door Unloading	25	43	100	31	45	74	5
Belt Loader Loading	36	60	60	14	60	60	14
Belt Loader Unloading	25	84	75	5	84	73	3
Mean Improvement				22			23

Table 5.1: Numbers of occurrences of each event class in dataset and performance of detection techniques.

5.3 Evaluation

Sequence-level optimization for the CRF is carried out in exactly the same way as in the previous Chapter. The only change is that instead of the Belief Propagation stage involving the summation of the log of the pairwise and observation likelihoods, it now involves summation of the weighted functions.

Table 5.1 compares results obtained with the CRF model versus the best Temporal Structure Model variant (with connections based on PPI) from the previous chapter. It appears that the CRF marginally outperforms the simpler model, with a mean improvement of 23

5.4 Future Directions

Conditional Random Fields are a powerful and very flexible class of model. The most natural way to apply them in this thesis seemed to be to use them to extend the tree structured

models which were already proving to be effective. However, it would be interesting to experiment with different levels of connectivity. Once obvious thing to try would be to set up the graph as pairwise fully connected as in [42]. Note that there is a big problem with this pairwise fully connected architecture on a theoretical level. In the tree structured case, the assumption is being made that a node is temporally dependant only on its neighbours. This inevitably results in the loss of some information, which is not ideal, but at least by retaining the most informative relations this loss can be minimised. In a pairwise fully connected architecture, where the pairwise relations were trained exactly as before, the problem would not just be loss of information, but also huge amounts of redundancy of information. If event A and event B always happen very close to one another, with event C coming some time after that; and if A and B were observed, there would then be two very similar messages about C coming from A and B being treated as independent pieces of information. The effect of this in a probabilistic framework would be that the temporal relations in larger graphs would dominate the observation likelihoods. It is likely for that reason that in [42], 'top-hat' functions are used rather than a peaked distribution like a Gaussian. The pairwise relations then just essentially define an allowable configuration space rather than attempting to convey richer information. By treating the pairwise functions as features in a CRF however, redundancy should not be a problem. In part of speech tagging for example, [69] millions of features are created- many of which are redundant, and the redundant information is downweighted by CRF optimization. The reason why this experiment was not straightforward to try was that a fully-connected structure would mean that Belief Propagation could no longer be relied on to provide exact inference or computation of the normalization constant. Loopy BP [54] could potentially be an option; though it is a computationally expensive procedure and given that training times for the CRF were already in the order of hours may not be practical. Another interesting option would be to try and apply Expectation Propagation (EP) [52] to the task. EP has recently gained much attention in the literature, especially since its deployment in Microsoft's In-

fer.NET [51]. EP is extremely fast since the inference is done analytically with conjugate exponential family distributions. The difficulty with application of EP to this problem would be that EP requires a step where the posterior probability over observation density multiplied by a prior must be approximated with a distribution of the same form as the prior. Minka has derived this projection for some well known distributions [52], but it is not clear how it would be done in general, and represents a significant research challenge in itself.

Chapter 6

Conclusion

The purpose of this thesis was to explore the state of the art in action recognition systems and to propose a generic architecture applicable in scenarios where temporal relations between events could provide additional information. This chapter summarizes how these goals were achieved and highlights the contributions of this thesis in three significant areas before ending with some general remarks and suggestions for future directions in the field.

6.1 Improving the Histogram of Optical Flow feature

Though widely cited in recent literature, the HOF feature is still a relatively new development and as such there are very few implementations in existence, with most researchers using the binaries made available by their creator, Laptev [47]. As such, not every aspect of the features has been extensively evaluated. The aspect focused on in this thesis is the means by which the features are normalized. Laptev uses a coarse binning strategy where he creates an additional bin for non-motion, then thresholds motion before placing a vote of a single unit magnitude into the relevant bin at each pixel. It was shown with thorough evaluation on two well-known datasets that this strategy can be outperformed by one which retains richer motion information by the abolition of the non-motion bin and retention of magnitudes using bilinear filtering and l^2-normalization.

6.2 A novel formulation for the event detection task

Taking inspiration from the success of Pictoral Structure Models in the field of object detection, a similar framework was suggested that could be employed to the task of event detection within structured scenarios, casting the midpoints of different events as the random variables to be optimized. Several differences were highlighted between the object detection and event detection tasks, which required extensions to the core PSM model. A number of methods for training the distributions over the pairwise temporal relations used in the model were considered, ranging from simple smoothed histograms to Gaussian Mixture Models trained with a complex Bayesian procedure. Two new metrics were introduced for learning structure in the MRF prior of the model, and evaluated versus the one used in PSMs. The first metric was given the name Potential Performance Improvement, as it was not a goodness-of-fit measure but instead a measure directly focused on its usefulness for detection at a target level of recall. The second measure was based on entropy of the pairwise distributions combined with the detector response at run time, made possible by doing the structure learning dynamically. Though the difference in performance between these metrics was not significant on the target dataset, some intuition was gained from the results about which may be most appropriate in different scenarios. Experimentation showed that each connection strategy had slightly different strengths, meaning that one can imagine circumstances under which any one of them could be preferable. In future work further experiments on other datasets would be desirable to fully determine the impact of dynamic structure learning. Finally, an efficient iterative algorithm was introduced to optimize for turnarounds with unkown numbers of event instances, the design of which allowed rich information to be incorporated without invalidating the independence assumptions required to keep the computation tractable. Results were achieved well above the strong baseline provided by the independent event detectors.

6.3 An extension to the Temporal Structure Model using Conditional Random Fields

In the final chapter, it was demonstrated how the Temporal Structure Model defined in the previous chapter could be extended within a Conditional Random Field formulation. The practical implications of training a CRF model for such an application were discussed and some results presented which demonstrated a modest improvement over the simpler model. A number of potential extensions to the model were suggested as avenues for future work.

6.4 Final Remarks

The review of existing high level action recognition methodologies revealed a literature that is relatively disconnected. Many interesting methods have been presented yet very few are directly comparable. In contrast, research in the related field of object detection has been driven forward in recent times by intense competition on large freely-available datasets. Competetive results on such a dataset are expected for new publications in that area. This makes it quite obvious which methods are currently at the forefront, at the cost of creating an environment that some might argue makes innovation more difficult as it is easier to squeeze additional performance through the incremental improvement of existing methods. A similar kind of community is developing in the area of low-level action recognition, around datasets such as KTH Actions and Hollywood2. Unfortunately, nothing similar is yet on the horizon for high-level action recognition. The result is a literature which seems a little more 'horses for courses'. Datasets for the high-level problem are expensive to gather, to annotate, distribute and process. Consisting of approximately forty hours of video, the Co-Friend dataset used in this thesis is larger than most and required thousands of hours of CPU time for training and experimentation. Yet on the other hand,

if it is viewed on a scenario level, the dataset is relatively small, something that became particularly apparent when trying to attach any significance to the results achieved in the CRF experiments. Could it really be argued that a dataset is large when it contains fewer than forty instances of a complex multi-agent scenario where the number of occurrences, the ordering and the timing of the constituent events are all variable? Unfortunately it seems unavoidable that practical concerns will continue to limit the size of datasets in this area in the immediate future, yet benchmarks are nevertheless important. In particular it is important for researchers to modularize the evaluation of high-level methods as much as possible in order to allow others to see to what degree the various components of a high level system are responsible for the overall performance. This was done to as great an extent as possible within this thesis by starting with low-level detectors whose performance is well understood, then using this as a benchmark and reference point from which to compare several varieties of model. Something which also needs to come with any future datasets is a common evaluation metric. In the literature surrounding the detection of simple events, it is customary to give performance figures in terms of Average Precision on a per-event basis and to plot precision-recall or ROC curves. It became clear in this work that generating these figures would simply not be possible for many higher-level systems due to interdependencies between the sensitivities of detectors relating to the different events. The solution to this problem, employed in this work, was to gauge performance in terms of precision-recall relative to a precision-recall curve generated by a lower level method. This seemed an appropriately flexible metric given the relatively loose definition of good performance inherent in the task. However, if one desired to produce a canonical dataset for activity recognition, it would be preferable to define with the dataset a cost function (perhaps at the event level) giving weight to false positives and false negatives, encapsulating performance in terms of a single figure. When a dataset in the area does gain the support of the community, it will be interesting to see how this influences the direction of research. It is likely that in the short term at least, whilst motion

tracking techniques remain imperfect, works that leverage dectors based on local features paired with interest point detectors or dense extraction will outperform methods that use tracking to abstract away from the video. If that prediction proves correct, the main challenges in this work of how to incorporate response from different detectors, how to make the independence assumptions required for tractability whilst retaining maximum useful information and how to evaluate each element of a complex system, will be encountered by many. Hopefully then, this thesis is well timed to be of use to other researchers in the area.

Bibliography

[1] S. Ali and M. Shah. Human action recognition in videos using kinematic features and multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):288 –303, Feb. 2010.

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

[3] E. Anderson, Z. Bai, and C. Bischof. *LAPACK Users'Guide*, volume 9. Society for Industrial Mathematics, 1999.

[4] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22:39–71, 1996.

[5] J.A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4:126, 1998.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[7] L.S. Blackford, A. Petitet, R. Pozo, K. Remington, R.C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, et al. An updated set of basic linear algebra

subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.

[8] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *The Tenth IEEE International Conference on Computer Vision (ICCV'05)*, pages 1395–1402, 2005.

[9] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[10] A.F. Bobick and J.W. Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267, 2001.

[11] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Computer Vision and Pattern Recognition, 1997. Proceedings. IEEE Computer Society Conference on*, pages 994 –999, 1997.

[12] M. Bregonzio, S. Gong, and T. Xiang. Recognising action as clouds of space-time interest points. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1948–1955. IEEE, 2009.

[13] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] M.T. Chan, A. Hoogs, J. Schmiederer, and M. Petersen. Detecting rare events in video using semantic primitives with hmm. In *Pattern Recognition. ICPR 2004. Proceedings of the 17th International Conference on*, volume 4, pages 150 – 154 Vol.4, 2004.

[15] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, pages 2:27:1–27:27, 2011.

[16] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge Univ Pr, 2000.

[17] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886 –893 vol. 1, june 2005.

[18] D. Damen and D. Hogg. Recognizing linked events: Searching the space of feasible explanations. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009*, pages 927–934, 2009.

[19] J.W. Davis and A.F. Bobick. The representation and recognition of action using temporal templates. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 928–934, 1997.

[20] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72, 2005.

[21] K.S.R. Dubba, A.G. Cohn, and D.C. Hogg. Event model learning from complex videos using ilp. In *Proceeding of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 93–98. IOS Press, 2010.

[22] A.A. Efros, A.C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 726 –733 vol.2, 2003.

[23] M. Eichner and V. Ferrari. We are family: Joint pose estimation of multiple persons. *Computer Vision–ECCV 2010*, pages 228–242, 2010.

[24] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[25] A. Fathi and G. Mori. Action recognition by learning mid-level motion features. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[26] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[27] P.F. Felzenszwalb and D.P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[28] J.R. Finkel, A. Kleeman, and C.D. Manning. Efficient, feature-based, conditional random field parsing. *Proceedings of ACL-08: HLT*, pages 959–967, 2008.

[29] M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *Computers, IEEE Transactions on*, 100(1):67–92, 1973.

[30] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[31] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[32] S. Hongeng and R. Nevatia. Multi-agent event recognition. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 84–91. IEEE, 2001.

[33] S. Hongeng, R. Nevatia, and F. Bremond. Video-based event recognition: activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding*, 96(2):129–162, 2004.

[34] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1450–1464, 2006.

[35] Yuri A. Ivanov and Aaron F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[36] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8):609–615, 1996.

[37] S. Johnson and M. Everingham. Learning effective human pose estimation from inaccurate annotation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1465–1472. IEEE, 2011.

[38] D. Karlis. An em algorithm for multivariate poisson distribution and related models. *Journal of Applied Statistics*, 30(1):63–77, 2003.

[39] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *IEEE International Conference on Computer Vision*, volume 23, pages 38–41. Citeseer, 2007.

[40] Yan Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 166 – 173 Vol. 1, oct. 2005.

[41] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[42] P. Kumar, P. Torr, and A. Zisserman. Extending pictorial structures for object recognition. *British Machine Vision Conference*, 2004.

[43] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Machine Learning*, pages 282–289. Citeseer, 2001.

[44] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123, 2005.

[45] I. Laptev, B. Caputo, C. Schüldt, and T. Lindeberg. Local velocity-adapted motion events for spatio-temporal recognition. *Computer Vision and Image Understanding*, 108(3):207–229, 2007.

[46] I. Laptev and T. Lindeberg. Space-time interest points. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 432 –439 vol.1, 2003.

[47] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008.

[48] D.J.C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge Univ Pr, 2003.

[49] D. Makris, T. Ellis, and J. Black. Intelligent Visual Surveillance: Towards Cognitive Vision Systems. *The Open Cybernetics & Systemics Journal*, 2(1):219–229, 2008.

[50] P. Matikainen, M. Hebert, and R. Sukthankar. Representing pairwise spatial and temporal relations for action recognition. *Computer Vision–ECCV 2010*, pages 508–521, 2010.

[51] T. Minka, J. Winn, J. Guiver, and A. Kannan. Infer .net 2.3, 2009. Microsoft Research Cambridge.

[52] T.P. Minka. Expectation propagation for approximate Bayesian inference. In *Uncertainty in Artificial Intelligence*, volume 17, pages 362–369. Citeseer, 2001.

[53] C.Z. Mooney and R.D. Duval. *Bootstrapping: A nonparametric approach to statistical inference*, volume 95. Sage Publications, Incorporated, 1993.

[54] K.P. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

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

[56] J.C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008.

[57] Juan Carlos Niebles, Chih-Wei Chen, , and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *Proceedings of the 12th European Conference of Computer Vision (ECCV)*, 2010.

[58] F. Niu and M. Abdel-Mottaleb. Hmm-based segmentation and recognition of human activities from video sequences. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 804–807. IEEE, 2005.

[59] T.E. Oliphant. *A Guide to NumPy*, volume 1. Trelgol Publishing, 2006.

[60] Nuria Oliver, Ashutosh Garg, and Eric Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and*

Image Understanding, 96(2):163 – 180, 2004. Special Issue on Event Detection in Video.

[61] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[62] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *In NIPS*, pages 1097–1104. MIT Press, 2004.

[63] H. Ragheb, S. Velastin, P. Remagnino, and T. Ellis. Human action recognition using robust power spectrum features. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 753–756. IEEE, 2008.

[64] C.E. Rasmussen. The infinite Gaussian mixture model. *Advances in neural information processing systems*, 12:554–560, 2000.

[65] C.E. Rasmussen. Gaussian processes in machine learning. *Advanced Lectures on Machine Learning*, pages 63–71, 2004.

[66] I. Saleemi, L. Hartung, and M. Shah. Scene understanding by statistical modeling of motion patterns. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2069 –2076, 2010.

[67] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[68] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3-Volume 03*, page 36. IEEE Computer Society, 2004.

[69] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for*

Computational Linguistics on Human Language Technology-Volume 1, pages 134–141. Association for Computational Linguistics, 2003.

[70] L. Sigal and M.J. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2041–2048. IEEE, 2006.

[71] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional models for contextual human motion recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1808–1815. IEEE, 2005.

[72] M. Sridhar, A.G. Cohn, and D.C. Hogg. Unsupervised learning of event classes from video. In *Proc. AAAI*, pages 1631–1638. AAAI Press. Menlo Park, 2010.

[73] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Computer Vision, 1995. Proceedings., International Symposium on*, pages 265–270. IEEE, 1995.

[74] C. Stauffer and WEL Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[75] C. Sutton and A. McCallum. *An introduction to conditional random fields for relational learning*. Introduction to statistical relational learning. MIT Press, 2006.

[76] D. Thirde, M. Borg, J. Aguilera, H. Wildenauer, J. Ferryman, M. Kampel, et al. Robust real-time tracking for visual surveillance. *EURASIP Journal on Advances in Signal Processing*, 2007:3, 2007.

[77] B. Töreyin, Y. Dedeoğlu, and A. Cetin. HMM based falling person detection using both audio and video. *Computer Vision in Human-Computer Interaction*, pages 211–220, 2005.

[78] D.L. Vail, M.M. Veloso, and J.D. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8. ACM, 2007.

[79] V.N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York Inc, 2000.

[80] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.*, volume 1, pages I–511. IEEE, 2001.

[81] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2002.

[82] H. Wang, M.M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conference, London, UK*, pages 1–11, 2009.

[83] Xiaogang Wang, Xiaoxu Ma, and Eric Grimson. Unsupervised activity perception by hierarchical bayesian models. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[84] Y. Wang and G. Mori. Learning a discriminative hidden part model for human action recognition. *Advances in Neural Information Processing Systems (NIPS)*, 21:1721–1728, 2008.

[85] Yang Wang and G. Mori. Max-margin hidden conditional random fields for human action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 872 –879, june 2009.

[86] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2-3):249–257, 2006.

[87] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. *Computer Vision–ECCV 2008*, pages 650–663, 2008.

[88] T. Xiang and S. Gong. Beyond tracking: Modelling activity and understanding behaviour. *International Journal of Computer Vision*, 67(1):21–51, 2006.

[89] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using Hidden Markov Model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92*, pages 379–385, 1992.