Appears in Proceedings of the European Conference on Artificial Intelligence (ECAI'02), Lyon, 2002.

Modeling Interaction Using Learnt Qualitative Spatio-Temporal Relations and Variable Length Markov Models

Aphrodite Galata, Anthony Cohn, Derek Magee and David Hogg [1]

Abstract.

Motivated by applications such as automated visual surveillance and video monitoring and annotation, there has been a lot of interest in constructing cognitive vision systems capable of interpreting the high level semantics of dynamic scenes. In this paper we present a novel approach for automatically inferring models of object interactions that can be used to interpret observed behaviour within a scene. A real-time low-level computer vision system, together with an attentional control mechanism, are used to identify incidents or events that occur in the scene. A data driven approach has been taken in order to automatically infer discrete and abstract representations (*symbols*) of primitive object interactions; effectively the system learns a set of qualitative spatial relations relevant to the dynamic behaviour of the domain. These symbols then form the alphabet of a VLMM which automatically infers the high level structure of typical interactive behaviour. The learnt behaviour model has generative capabilities and is also capable of recognizing typical or atypical activities within a scene. Experiments have been performed within the traffic monitoring domain; however the proposed method is applicable to the general automatic surveillance task since it does not assume a priori knowledge of a specific domain.

1 INTRODUCTION

In recent years there has been a lot of interest in constructing cognitive vision systems capable of interpreting the high level semantics of dynamic scenes [3, 6, 13, 19, 2, 7, 8]. However, whereas previous authors have generally selected in advance which types of interactions to model/recognise, the aim of this paper (as in [6, 2]) is to not to predetermine these activities, but to learn them automatically.

Our proposed method has some similarities with the work of Brand and Kettnaker [2]. First, they both produce one single model, that descibes the entire behaviour space as compared to the usual practice of having separate models for each (usually predefined) behaviour. We propose the use of VLMMs [20, 10] to infer the structure of the underlying training data automatically; similarly, the entropically estimated HMMs developed by Brand are capable of extracting the underlying hidden structure of the data. Finally, both models are capable of encoding the high order temporal dependencies that are present in complex behaviours. A major difference however is that Brand uses low level, continuous variables, whereas our learnt VLMM is essentially a *symbolic* predictive model: the underlying continous variables are first abstracted to a discrete space, analogous to the set of finite relations in a qualitative spatial relation representation such as that used by [6]. In some cases, this has the advantage of being rather more immune to the problems of noise whilst still managing to preserve the essential underlying patterns or dependencies that govern behaviour in the observed domain. Arguably it also produces a more understandable model since its components are higher level abstractions.

Our approach is also similar to that of [6] which learns qualitative spatio-temporal models of events in traffic scenes (e.g. following, overtaking). However, whereas that system required a predefined set of qualitative spatial relations, our approach essentially learns these automatically and moreover explicitly computes probabilities associated with the behavioural patterns.

Thus we use a data driven approach to automatically infer discrete and abstract representations (*symbols*) of primitive object interactions. These symbols are then used as an alphabet to infer the high level structure of typical interactive behaviour using VLMMs [20, 10]. It is worth pointing out explicitly that the use of probabilistic reasoning here contrasts with conventional low level use of probabilities – this Markov model concerns high level semantic notions.

These scene feature descriptors are invariant of the absolute position and direction of the interacting objects within a scene. They constitute the input to a statistical learning framework [15] where discrete representations of interactive behaviours can be learned by modelling the probability distribution of the feature vectors within the interaction feature space.

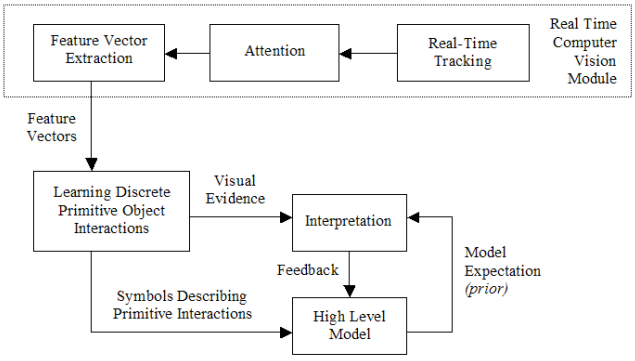

Figure 1. System Overview

[1] School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom. E-mail: {afro, agc, drm, dch}@comp.leeds.ac.uk

Figure 1 gives an overview of the system including the learning element of the architecture whereby the typical behaviour patterns are learned, stored and used to drive interpretation. The system can currently be used to recognize typical interactive behaviour within the traffic domain and identify atypical events.

2 LEARNING QUALITATIVE SPATIO-TEMPORAL RELATIONS

2.1 Detecting and tracking objects

A real time computer vision system [18] is used that detects and tracks moving objects within a scene. The tracking system, which is tolerant to slight lighting changes, uses a modified version of the Stauffer and Grimson [23] adaptive background model to detect moving objects. In addition, it makes explicit use of a foreground model for size, colour distribution and velocity (which is assumed to be locally invariant in time) invariants for a particular object. Foreground pixels identified by the background model are compared with various instances of the foreground model to determine to which model they belong. A Kalman filter is used to propagate these models over time (for more details, see [18]).

Figure 2. Example traffic scene

The tracker assigns a unique ID to each newly detected object in the scene and tracks changes in the object's position and velocity. It should be noted that tracking is performed on an estimated ground plane. Thus, for each video frame, the tracker returns the estimated position and velocity of each tracked object in ground plane co-ordinates. Figure 2 illustrates a scene from the example traffic domain application.

2.2 Feature space representation for object interaction

Modelling object interaction is of particular interest since it allows reasoning to be extended from individuals to groups of objects. In this paper we are interested in modelling the joint (combined) behaviour of pairs of interacting objects. Within this scheme, candidate interactions are identified to yield a set of interactive joint behaviours for training by using an attentional mechanism based on proximity cues.

For each image frame, we assume that each tracked object is centered within a rectangular area (*attentional window*), and its direction of motion is aligned with the longest axis of the rectangle. An interactive event between two objects is signalled whenever an object falls within the attentional window of another object which we call the *reference* object. The size of the rectangular area is currently set by hand, however an appropriate size for the attentional window could be learned automatically from the training data.

Many interactions are typified by the evolving spatial relationships between interacting objects and thus such relationships must be encoded within an appropriate feature vector describing the object interaction. In addition, since the location within a scene at which an interactive behaviour occurs is probably of less relevance than the interaction itself, we believe that non-scene-specific feature vector representations are probably more appropriate. Thus, our chosen feature space representation is invariant of the absolute position and direction of the interacting objects within a scene.

The feature vector representation describes the relative velocity (magnitude and direction) and the spatial relationship between a reference car and another car that falls within its attentional window. The feature vector consists of the velocity magnitude of the reference object, the vector representing the relative distance between the two objects (in the local co-ordinates of the reference object) and the velocity vector of the other object (again in local co-ordinates):

$$\mathbf{F}_{r_t} = (\|\dot{\mathbf{x}}_r\|, x_d, y_d, \dot{\mathbf{x}}_o, \dot{\mathbf{y}}_o) \tag{1}$$

It should be noted that every feature vector is normalised so that the direction of motion of the reference object is aligned with the y-axis.

Joint behaviours of pairs of interacting objects may be viewed as smooth trajectories within the feature space that are sampled at frame rate generating sequences, $\mathcal{F}_j$, of feature vectors $\mathbf{F}_{r_t}$. Each sequence describes the temporal evolution of an interactive behaviour:

$$\mathcal{F} = \{\mathbf{F}_{r_0}, \mathbf{F}_{r_1}, \ \ldots, \mathbf{F}_{r_m}\}. \tag{2}$$

2.3 Learning discrete interaction primitives

In order to learn discrete representations of object interactions, we replace the feature vectors $\mathbf{F}_{r_t}$ with their nearest (in a Euclidean sense) prototype from a finite set of prototypical object interactions. These prototypical object interactions are learned using vector quantisation. *Vector Quantization* [9, 12] is a technique originally used for data compression which provides a method of approximating the probability density function of a vector variable $\mathbf{x}(t)$ using a finite number of prototype vectors (usually referred to as *codebook* vectors) $\mathbf{c}_i(t), i = 1, 2, \ \ldots, k$.

A limitation of classical VQ algorithms such as the k-means algorithm [22] is that the final distribution of prototypes is very sensitive to their initial placement within the feature space and this can result in sub-optimal distributions. The usual solution is to repeat the k-means algorithm for several sets of prototype initialisation values and then choose the prototype distribution that gives the best approximation for the probability density function of the training data.

Instead of taking this approach, we use a variant of the VQ algorithm proposed by Johnson and Hogg [15] which is less sensitive to the initial positioning of prototypes. It is based on the competitive learning paradigm [21, 17]; however it also incorporates a prototype sensitivity mechanism. This results in a robust VQ algorithm which is less sensitive to the initial placement of prototypes and gives approximately correct density matching.

After VQ is performed, each interactive behaviour is represented by a sequence of prototypes:

$$\{\mathbf{p}_{i_0},\ \mathbf{p}_{i_1},\ \ldots,\ \mathbf{p}_{i_m}\}. \quad (3)$$

Figure 3 illustrates the learnt primitive interactions for the traffic domain example application. These can be viewed as a qualitative discretisation of the continuous relational space. Whereas in a conventional qualitative spatial reasoning representation, the discretisation would be manually preassigned, we are able to learn a representation which maximises the discernability given a granularity (i.e. the number of relations desired). Note that the representation is approximately symmetric with respect to an axis aligned to the direction of motion of the reference object.

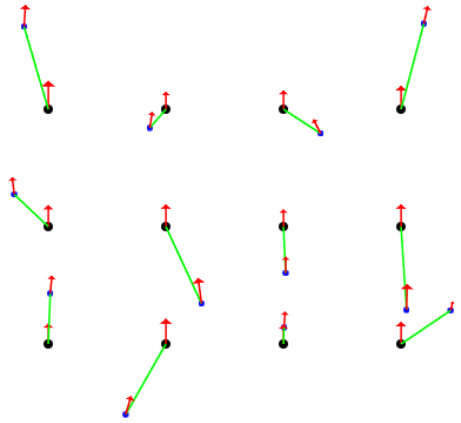

Figure 3. Learnt primitive interactions – traffic domain example. The two dots represent pairs of close vehicles (distinguished by the size of the dot – the larger dot being the reference vehicle). The arrows show their direction of movement and the connecting vector their relative orientation and distance. These patterns represent typical "midpoints" as result of clustering the input data into n different conceptual "regions".

2.4 Experimental data

Training data was generated from a 10-min video sequence of a traffic scene (see Fig. 1), sampled at 25 frames per second. In order to reduce noise, the output of the tracker was low-pass filtered with a 5Hz cut-off filter (a vehicle needs to be present in the scene for at least one fifth of a second, otherwise it is ignored and treated as noise output from the tracker, rather than an actual vehicle).

Using the attentional control mechanism, 80,000 (approximately) feature vectors representing instances of vehicle interactions were generated. A set of 12 prototypical feature vectors corresponding to primitive interactions between vehicles was learned from this data set after performing VQ. Experiments were performed to decide the appropriate number of prototypes with respect to model performance in analytical and generative tasks. A VLMM learnt over 12 prototypes gave the best performance for this data set.

3 MODELLING BEHAVIOUR USING VARIABLE LENGTH MARKOV MODELS

We are interested in building models of behaviour which are able to support both recognition and generative capabilities such as the prediction of future behaviours or the synthesis of realistic sample behaviours. We achieve this by using variable memory length Markov models (VLMMs) [10] to efficiently encode the sequences of the prototypes corresponding to observed interactive behaviour in the feature space.

3.1 Variable length Markov models

Variable length Markov models deal with a class of random processes in which the memory length varies, in contrast to an n-th order Markov models. They have been previously used on the data compression [4, 1] and language modelling domains [20, 10, 11] and recently, they have been successfully introduced in the computer vision domain for automatically inferring stochastic models of the high level structure of complex and semantically rich human activities [7, 8]. Their advantage over a fixed memory Markov model is the ability to locally optimise the length of memory required for prediction. This results in a more flexible and efficient representation which is particularly attractive in cases where we need to capture higher-order temporal dependencies in some parts of the behaviour and lower-order dependencies elsewhere.

Assume w is a string of tokens used as a memory to predict the next token a' according to an estimate $\hat{\mathrm{P}}(a'|w)$ of $\mathrm{P}(a'|w)$. The main idea behind the variable length modelling method is that if the output probability $\hat{\mathrm{P}}(a'|aw)$ that predicts the next token a' is significantly different from $\hat{\mathrm{P}}(a'|w)$, then the longer memory aw may be a better predictor than w. A weighted Kullback-Leibler divergence [20] is used to measure the additional information that is gained by using the longer memory aw for prediction instead of the shorter memory w:

$$\Delta H(aw, w) = \hat{\mathrm{P}}(aw) \sum_{a'} \hat{\mathrm{P}}(a'|aw) \log \frac{\hat{\mathrm{P}}(a'|aw)}{\hat{\mathrm{P}}(a'|w)}. \quad (4)$$

If $\Delta H(aw, w)$ exceeds a given threshold ε, then the longer memory aw is used, otherwise the shorter memory w is considered sufficient for prediction.

The transition probabilities and priors are derived from estimates of $\mathrm{P}(a_n|a_1a_2\ldots a_{n-1})$ and $\mathrm{P}(a_1a_2\ldots a_n)$ calculated for various values of n ($n = 1, 2, \ldots, N$). The estimates are given by:

$$\hat{\mathrm{P}}(a_n|a_1a_2\ldots a_{n-1}) = \frac{v(a_1a_2\ldots a_{n-1}a_n)}{v(a_1a_2\ldots a_{n-1})}, \quad (5)$$

and

$$\hat{\mathrm{P}}(a_1a_2\ldots a_n) = \frac{v(a_1a_2\ldots a_n)}{v_0}, \quad (6)$$

where $v(a_1a_2\ldots a_n)$ is the number of times the string of tokens $a_1a_2\ldots a_n$ appears in the training data and v_0 is the total length of the training sequences.

The training algorithm involves building a *prefix tree* [10] where each node corresponds to a string up to a predetermined length N. The transition frequencies are counted by traversing the tree structure repeatedly with strings of length N where the strings are generated by sliding a window of fixed length N along a training sequence of tokens. Transition probabilities are computed using equation 6 and a pruning procedure is then applied, while the prefix tree is converted to a *prediction suffix tree* [20]. For each node n_p in the prefix tree, a corresponding node n_s in the suffix tree is created if and only if the Kullback-Leibler divergence between the probability distribution at n_p and the probability distribution at its ancestor node in the prefix tree is larger than threshold ε. Finally, the suffix tree is converted to an automaton representing the trained VLMM. A more detailed

description on building and training variable length Markov models is given by Ron *et al.* [20].

Thus, a VLMM is equivalent to a *Probabilistic Finite State Automaton* (PFSA) represented by $\mathcal{M} = (Q, \Sigma, \tau, \gamma, \pi)$ where Σ is a finite *alphabet* –the set of tokens– and Q is a finite set of model states. Each state corresponds to a token string of length at most N, ($N \geq 0$), representing the *memory* for a conditional transition of the VLMM. The *transition function* is $\tau : Q \times \Sigma \rightarrow Q$ and $\gamma : Q \times \Sigma \rightarrow [0, 1]$ is the *output probability function* representing the memory conditioned probabilities of the next *token* $a \in \Sigma$. Finally, $\pi : Q \rightarrow [0, 1]$ is the probability distribution over the *start states*. The functions γ and π are such that for every $q \in Q$, $\sum_{a \in \Sigma} \gamma(q, a) = 1$ and $\sum_{q \in Q} \pi(q) = 1$.

3.2 Modelling interactive behaviour using a VLMM over prototypes

Temporal dependencies in behaviour are learned by using a VLMM to capture the memory conditioned probabilities of transitions between the learnt prototypes. The choice of VLMMs as a mathematical framework for modelling interactive behaviour is based on their ability to capture behavioural dependencies at variable temporal scales in a simple and efficient manner.

Initially, the training sequences of feature vectors are converted into sequences of prototypical object interactions by observing the closest prototype, in a *nearest neighbour* sense, to the current training feature vector at each time instant. The output sequences are then used to train a VLMM represented by the PFSA $\mathcal{M}_p = (Q_p, \mathcal{P}, \tau_p, \gamma_p, \pi_p)$ where $\mathcal{P}$ is the finite set of prototypical object interactions and $\mathcal{M}_p$ represents the learnt interactive behaviour model.

The algorithm for training variable length Markov models of behaviour has two parameters, the maximal order N of the model and the threshold value ε controlling the accuracy of the learnt behaviour model. The choice of these parameters plays an important role in the performance of the learnt behaviour model (especially that of ε).

The choice of parameters for the training algorithm can be based on a measure of how well the learned model describes the training data. One such measure, traditionally used in text compression [1] and language modelling [10, 11], can be the *model cross-entropy rate* (or *model entropy*) [5, 11] $\hat{\mathrm{H}}_M$:

$$\hat{\mathrm{H}}_M = -\frac{1}{n} \log \mathrm{P}_M(R) \tag{7}$$

where R is a sample string of symbols of length n.

In our case, $M = \mathcal{M}_p$, $R = \mathbf{p}_{r_1} \ldots \mathbf{p}_{r_n}$ is a sequence of prototype labels of length n, and $\mathrm{P}_{\mathcal{M}_p}(R)$ is the probability of R according to the VLMM model $\mathcal{M}_p$:

$$\mathrm{P}_{\mathcal{M}_p}(R) = \prod_{i=1}^{n} \mathrm{P}(\mathbf{p}_{r_i} | q_{r_i}), \tag{8}$$

and $\mathrm{P}(\mathbf{p}_{r_i} | q_{r_i}) = \gamma_p(q_{r_i}, \mathbf{p}_{r_i})$.

Using equation (8) together with (7), an estimate of model entropy for the learnt VLMM model M_p is given by:

$$\hat{\mathrm{H}}_{\mathcal{M}_p} = -\frac{1}{n} \sum_{i=1}^{n} \log \mathrm{P}(\mathbf{p}_{r_i} | q_{r_i}). \tag{9}$$

A measure of how well the learned VLMM describes the training data is given by calculating the model entropy (Eq. 9) over the training data. A good aproximation of observed behaviour is indicated by a low model entropy value.

3.2.1 *Performing generative tasks*

The trained model $\mathcal{M}_p$ represents the learnt interactive behaviour model and has generative capabilities. Behaviour generation is achieved by traversing the PFSA, selecting either the most likely transition (maximum likelihood behaviour generation) or sampling from the transition distribution (stochastic behaviour generation) at each state, and emitting the corresponding prototype vectors. This results in an ordered set $\mathcal{G}$ of prototype vectors $\mathbf{p}_{q_r}$ which are the output of the transitions $q_{r+1} = \tau_b(q_r, \mathbf{p}_{q_r})$ between states q_r, q_{r+1}.

In order to use the model $\mathcal{M}_p$ for behaviour prediction, it is first necessary to locate the current model state. Since model states may encode a history of previous behaviour, the model is initially used in a recognition mode, accepting successive prototypes representing observed behaviour and making the corresponding state transitions. Having located the current model state, prediction of future behaviour can be achieved using the model either as a stochastic or a maximum likelihood behaviour generator.

A VLMM needs to be able to handle the problem of *unseen events* – cases where token sequences which might have not appeared previously in the training data, appear during the recognition process. Problems related to the sparseness of the training data are well studied in the speech recognition and language modelling domains and various methods for handling unseen events have been proposed to compensate for the scarcity of training data (see for example [14] for an in-depth discussion on the subject). One of the best known methods of handling unseen events is the *backing-off* method [16], which is quite prevalent in state-of-the-art speech recognisers.

We use a simple backing-off method which is an extension of the method proposed by Guyon *et al* in [10]. Assume the model $\mathcal{M}_p$ is presented with a prototype $\mathbf{p}_i \in \mathcal{P}$ while in a state which emits this prototype with probability zero, given a history $s\mathbf{p}_j$ of previous prototypes. If there is a state within the model that emits $\mathbf{p}_i$ with a non-zero probability given only the history of the previous prototype $\mathbf{p}_j$, then we back-off at this state and emit $\mathbf{p}_i$. Otherwise, we return to the initial model state, lose all the previous memory and emit $\mathbf{p}_i$ with the prior probability $\hat{\mathrm{P}}(\mathbf{p}_i)$. This *backing-off* method is simple but effective and was found to give similar or slightly better results compared to the methods proposed by Katz [16] and Guyon *et al* [10].

3.2.2 *Assessing model performance*

The performance of the model can be measured by the model entropy over the test data. A model that has achieved a good generalisation of the observed behaviour, will result in very similar entropy values over both the training and the test data. Otherwise, significantly different values indicate a model that is overfit to the training data and therefore is not fully representative of the target behaviour and thus more training data is required.

For our traffic domain example, we used the 12 learnt prototypes as an alphabet to model VLMMs for different values of N and ε. The training and test data sets had the same number of prototype sequences (940 each). The sequences where randomly chosen to belong to one or the other data set.

Figures 4 and 5 demonstrate the results when $\varepsilon = 0$ and $\varepsilon = 0.0001$ are used respectively. When using a maximum memory length $N = 6$, the sizes of the corresponding VLMMs where 13,808 and 266 states respectively. In the general case, an increase in the value of ε will result in an increase of the model's entropy whereas an increase on the model's maximum memory N will decrease the

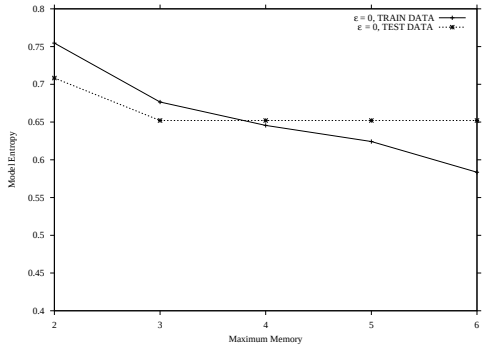

Figure 4. Traffic domain example: Maximum memory length vs model for the training and test data when $\varepsilon = 0$

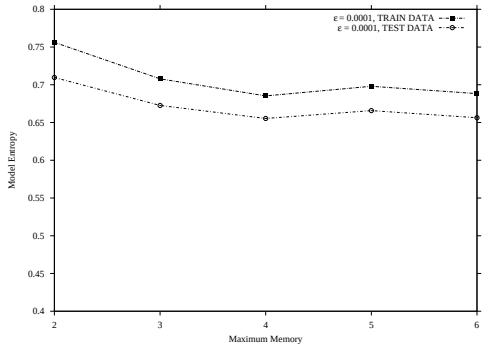

Figure 5. Traffic domain example: Maximum memory length vs model for the training and test data when $\varepsilon = 0.0001$

model entropy (however, increasing N does not always guarantee lower model entropy). As can be seen from the graphs, $\varepsilon = 0$ gives the best model, although at the expense of a large model size (number of states and transitions within VLMM). However, a slight increase in ε can dramatically decrease the number of states within the model with only a very small increase in the model entropy.

4 CONCLUSION

The main purpose of this paper has been to show how a discrete, and essentially symbolic set of relationships representing the spatio-temporal relationship between pairs of close moving objects, can be learned automatically and then used as the alphabet for a VLMM which learns the typical patterns of behaviour in the domain. The principal novelty of our approach is exactly this: to propose a method for learning the alphabet of spatio-temporal relationships and to apply a VLMM to this rather high level set of entities rather than much lower level perceptual artifacts.

Of course there remain many possibilities for further work. We are evaluating the generative and predictive powers of the models and we will apply it to other domains. Note that in order to be predictive, given the normalised feature representation, one may wish also to learn the typical paths (c.f. [8, 15]) of motion so that the behaviours can be applied "in context". We may also consider other kinds of spatio-temporal relationships apart from orientation, distance and velocity. Certain behaviours may be position or orientation dependent so we may also need to experiment with non normalised feature vectures (or rather, with different normalisations, since normalisation is a useful abstraction mechanism in machine learning). It would also be useful to experiment with ways of learning "closeness" (i.e. the size of the attentional window). However we believe the present system represents a useful first step in the direction of learning high level "cognitive" models of dynamic behaviours from video data.

ACKNOWLEDGEMENTS

The support of the EU under the CogVis project IST-2000-29375 is gratefully acknowledged.

REFERENCES

[1] T. Bell, J. Cleary, and I. Witten, *Text Compression*, Prentice Hall, 1990.
[2] M. Brand and V. Kettnaker, 'Discovery and Segmentation of Activities in Video', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22**(8), 844–851, (2000).
[3] Hilary Buxton and Richard Howarth, 'Spatial and temporal reasoning in the generation of dynamic scene descriptions', in *Proceedings on Spatial and Temporal Reasoning*, ed., R. V. Rodríguez, pp. 107–115, Montréal, Canada, (aug 1995). IJCAI-95 Workshop.
[4] G. Cormack and R. Horspool, 'Data Compression using Dynamic Markov Modelling', *Computer Journal*, **30**(6), 541–550, (1987).
[5] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley, 1991.
[6] J. Fernyhough, A.G. Cohn, and D. Hogg, 'Constructing qualitative event models automatically from video input', *Image and Vision Computing*, **18**, 81–103, (2000).
[7] A. Galata, N. Johnson, and D Hogg, 'Learning behaviour models of human activities', in *British Machine Vision Conference, BMVC'99*, (1999).
[8] A. Galata, N. Johnson, and D Hogg, 'Learning Variable Length Markov Models of Behaviour', *Computer Vision and Image Understanding (CVIU) Journal*, **81**(3), 398–413, (2001).
[9] R. M. Gray, 'Vector Quantization', *IEEE ASSP Magazine*, **1**(2), 4–29, (April 1984).
[10] I. Guyon and F. Pereira, 'Design of a Linguistic Postprocessor using Variable Memory Length Markov Models', in *International Conference on Document Analysis and Recognition*, pp. 454–457, (1995).
[11] J. Hu, W. Turin, and M Brown, 'Language Modelling using Stochastic Automata with Variable Length Contexts', *Computer Speech and Language*, **11**(1), 1–16, (1997).
[12] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition*, volume 7 of *Edinburgh Information Technology series*, Edinburgh University Press, 1990.
[13] Y. Ivanov and A. Bobick, 'Parsing Multi-Agent Interactions', in *IEEE Conf. Computer Vision and Pattern Recognition*, (1999).
[14] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, 1998.
[15] N. Johnson and D. Hogg, 'Learning the Distribution of Object Trajectories for Event Recognition', *Image and Vision Computing*, **14**(8), 609–615, (1996).
[16] S. Katz, 'Estimation of probabilities from sparse data for the language model component of a speech recognizer', *IEEE Transactions on Acoustics, Speech and Signal Processing*, **35**(3), 400–401, (1987).
[17] T. Kohonen, 'The Self-Organizing Map', *Proceedings of the IEEE*, **78**(9), 1464–1480, (1990).
[18] D. Magee, 'Tracking multiple vehicles using foreground, background and motion models', in *Proc. ECCV Workshop on Statistical Methods in Video Processing*, (2002).
[19] N. Oliver, B. Rosario, and A. Pentland, 'A Bayesian Computer Vision System for Modeling Human Interactions', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22**(8), 831–843, (2000).
[20] D. Ron, S. Singer, and N. Tishby, 'The Power of Amnesia', in *Advances in Neural Information Processing Systems*, volume 6, 176–183, Morgan Kauffmann, (1994).
[21] D. E. Rumelhart and D. Zipser, 'Feature Discovery by Competitive Learning', *Cognitive Science*, **9**, 75–112, (1985).
[22] R. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, 1992.
[23] C. Stauffer and W. Grimson, 'Adaptive background mixture models for real-time tracking', in *Proc. CVPR*, pp. 246–252, (1999).