Submitted to the British Machine Vision Conference 2004

Autonomous learning of perceptual categories and symbolic protocols from audio-visual input

C. J. Needham, D. R. Magee, V. Devin
P. Santos, A. G. Cohn and D. C. Hogg
School of Computing
The University of Leeds
Leeds, LS2 9JT, UK
vision@comp.leeds.ac.uk
http://www.comp.leeds.ac.uk/vision

Abstract

The development of cognitive vision systems that autonomously learn how to interact with their environment through input from sensors is a major challenge for the Computer Vision, Machine Learning and Artificial Intelligence communities. This paper presents a framework in which a symbolic inference engine is integrated with a perceptual system. The rules of the inference engine are learned from audio-visual observation of the world, to form an interactive perceptual agent that can respond suitably to its environment. From the video and audio input streams interesting objects are identified using an attention mechanism. Unsupervised clustering of the objects into perceptual categories generates a symbolic data stream. Inductive Logic Programming is then used to generalise rules from this symbolic data. Although in principle the framework could be applicable to a wide range of domains (perhaps of most interest to the automatic programming of robots), it is demonstrated here in simple game playing scenarios. First the agent observes humans playing a game, and then attempts to play using the learned perceptual categories and symbolic protocols.

1 Introduction

The perceived world may be thought of as existing on two levels: the sensory level in which meaning must be extracted from patterns in continuous observations, and the conceptual level in which the relationships between discrete concepts are represented and evaluated. Making the link between these two levels is key to the development of artificial cognitive systems that can exhibit human-level qualities of perception, learning and interaction. This is essentially the classic AI problem of "Symbol Grounding" [11]. The ultimate aim of our work is fully autonomous learning of both continuous models, representing object properties, and symbolic models of temporal events, defining the implicit temporal protocols present in many structured visual scenes.

Much work has been carried out in the separate areas of pattern recognition and model building in continuous data (see for example [5]) and symbolic learning in various domains such as robotics/navigation [3], bioinformatics [21] and language [13]. Several earlier systems have linked low-level video analysis systems with high-level (symbolic) event analysis in an end-to-end system, such as the work of Siskind [20] that uses a hand-crafted symbolic model of 'Pickup' and 'Put-down' events. This is extended in [6] to include a supervised symbolic event learning module, in which examples of particular event types are presented to the learner. Moore and Essa [17] present a system for recognising temporal events from video of the card game 'blackjack'. In that work, multiple low-level continuous temporal models (Hidden Markov Models), and object models (templates) are learned using a supervised procedure, and activity is recognised using a hand defined Stochastic Context-Free Grammar. A similar approach is used by Ivanov and Bobick [12] for gesture recognition and surveillance scenarios. However, none of these systems is capable of autonomous (unsupervised) learning of both continuous patterns and symbolic concepts. The motivation behind our research is to learn both low-level continuous object models and high-level symbolic models from data in an arbitrary scenario with no human interaction. Systems capable of unsupervised learning of both continuous models of image patches and grammar-like (spatial) relations between image patches have been presented by the static image analysis community (e.g. [1]). These involve the use of general (non-scene specific) background knowledge of the type of relations that may be important (e.g near, far, left-of, etc.). It is our aim to develop conceptually similar approaches for the analysis of dynamic video data. These would be similar to the grammars used in [17, 12], which are currently hand defined. A perception-action learning approach will be employed in order to learn an agent that can interact with the world. In the work of [7], the link between visual perception of action and generation of the same action is learned for a humanoid robot performing simple tasks. Initially the action is performed by a human. The robot subsequently learns to mimic this action by experimentation. It can then copy an action observed at a later time. Such learning would be a valuable addition to our framework.

The contexts we envisage also require audio analysis and generation. Speech recognition and production software could be used to perform this task, normally requiring supervised learning [4, 8, 19]; however an unsupervised approach to this task is favoured to fit in with the philosophy of learning a cognitive agent. Such an approach also has the advantage that participants can make non-word utterances (e.g. animal noises) or make sounds using objects or instruments.

The proposed framework consists of three elements: an attention mechanism, unsupervised learning of perceptual categories (audible and visual), and symbolic learning of temporal protocols. Figure 1 provides an overview of the learning phase of the framework. Egocentric learning is carried out, meaning the constructed models are based on the behaviour of an agent with respect to the scenario, rather than being holistic models of the complete scenario. Models of the *protocols* of the activities performed are learned, as opposed to abstract descriptive rules (or even strategies), in order to easily drive the behaviour of a synthetic agent that can interact with the real world in a near-natural way, which is our aim.

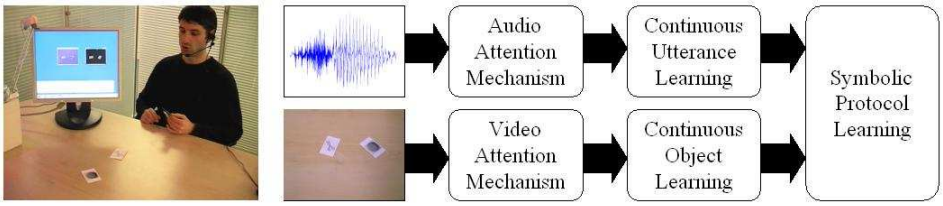


Figure 1: Overview of the learning framework

1.1 Game playing

The domain of game-playing has been chosen as our application domain, since it is rich in spatio-temporal rule-based protocols and it may be argued that many real-world social interaction scenarios may be modelled as games [10]. We have used the framework to learn the objects, utterances, events and protocols involved in various simple games including a version of "Snap", played with dice, and a version of the game "Paper, Scissors, Stone" played with cards. Typical setup is shown in Figure 1, and typical video input sequences are shown in Figure 2. Descriptions of the two games are:

Snap. A simple, single player, two dice game based on the card game snap. The two dice are rolled, one at a time. If the two dice show the same face, the player shouts "snap" and utters the instruction "collect-two". Both dice are picked up. Otherwise the player utters "collect-one", and the dice showing the lowest value face is picked up. Before rolling the player utters the instruction "roll-two" or "roll-one", depending on if there is a dice already on the table.

Paper, Scissors, Stone (PSS). Two players simultaneously select one of the object cards. Paper beats (wraps) stone, scissors beats (cuts) paper, and stone beats (blunts) scissors. Our version of this game is played with picture cards, rather than hand gestures for simplicity. Utterances ('I win', 'I lose', 'draw' and 'go') are spoken by the player to be replaced by a synthetic agent.

2 Autonomous learning

The framework divides learning into three parts: attention, learning perceptual categories and learning symbolic protocols. Figure 1 provides an overview of the learning phase. To facilitate autonomous (fully unsupervised) learning, a spatio-temporal attention mechanism is required to determine 'where' and 'when' significant object occurrences and interactions take place within the input video stream of the scenario to be learned from. We are interested in learning protocols which depend upon different *properties* of objects. In different situations, or games, different properties may be important. Such properties may be texture, shape, colour, position, etc. For each object identified by the attention mechanism a feature vector describing each property is extracted. Clusters are formed for each property separately using a clustering algorithm. Classifying models are then built using cluster membership as supervision. These models allow novel objects (identified by the attention mechanism) to be assigned a class label for each property (texture, position, etc.). This symbolic stream is combined with the vocal utterances issued by the player(s) participating in the game, which are extract and clustered from the audio signal

in a similar unsupervised manner. The symbolic stream is used as input for symbolic learning (generalisation) based on the Progol Inductive Logic Programming system [18]. The output of the continuous classification methods can be presented in such a way that instances of concepts such as equality, transitivity, symmetry, etc. may be generalised, in addition to generalisations about the protocols of temporal change. Advantages of Progol's learning approach are that learning can be performed using positive examples only, and that even with noisy data (such as imperfect clustering/classification, or occasional missing/additional objects), interesting rules are constructed.

The vocal utterances may either take the form of passive reactions (e.g. "snap"), or active statements of intent (e.g. "roll-one"). The latter generates an implicit link between the vocal utterance and the subsequent action in the data stream. Our high-level system can learn this link, and thus an agent based on the learned model can generate these utterances as a command to actively participate in its environment. It should be noted that conceptually the framework does not limit the perception and generation of action to vocal utterances; however a link is required between the perception and generation of individual agent actions for learned models to be used in an interactive agent. Vocal utterances are a good example of an action that can be perceived and generated without specialised hardware. It was for this reason they were chosen in this example implementation.

2.1 Spatio-temporal attention for object localisation

Video streams of dynamic scenes contain huge quantities of data, much of which is irrelevant to scene learning and interpretation. An attention mechanism is required to identify 'interesting' parts of the stream, in terms of spatial location ('where') and temporal location ('when'). For autonomous learning, models or heuristics are required to determine what is of interest, and what is not. Such models could be based on motion, novelty, high (or low) degree of spatial variation, or a number of other factors. It is highly likely that no single factor could provide a generic attention mechanism for learning and interpretation in all scenarios. It is our view that attention from multiple cues is required for fully generic learning.

For our implementation in the game-playing domain, an attention mechanism which can identify salient areas of space and time is necessary. For this reason motion has been chosen as it is straight-forward to work with. The spatial aspect of our attention mechanism is based around a generic blob tracker [15] that works on the principle of multi-modal (Gaussian mixture) background modelling, and foreground pixel grouping. This identifies the centroid location, bounding box and pixel segmentation of any separable moving objects in the scene in each frame of the video sequence. The temporal aspect of our attention mechanism identifies key-frames where there is qualitatively zero motion for a number of frames (typically 3), which are preceded by a number of frames (typically 3) containing significant motion.

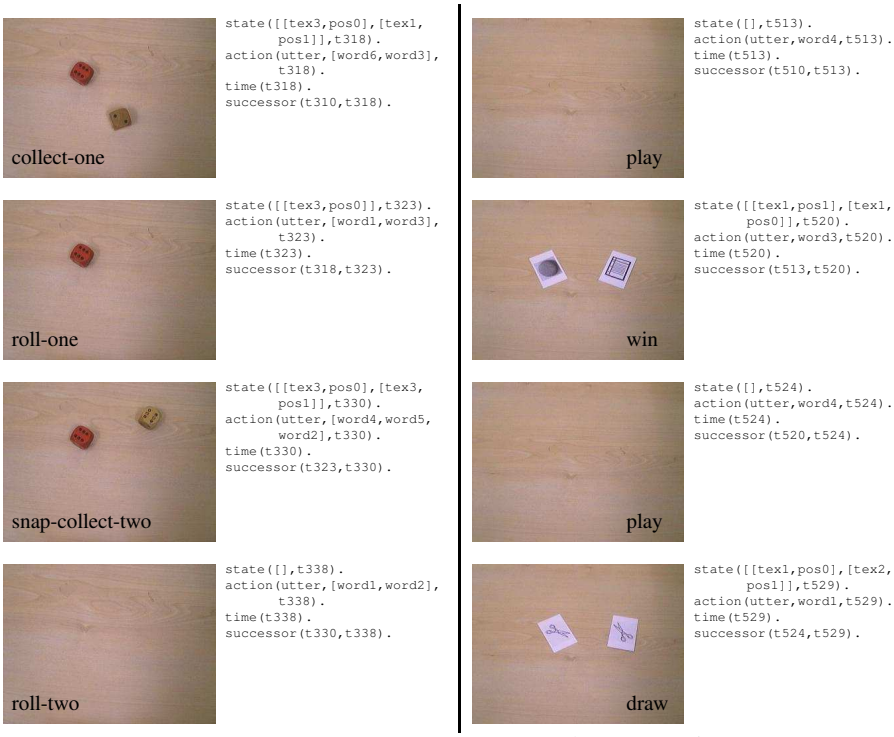
2.2 Continuous object learning and classification

In autonomous learning it is not in general possible to know *a-priori* what types of visual (and other) object properties are important in determining object context within a dynamic scene. For this reason the use of multiple (in fact large numbers of) features such as colour, texture, shape, position, etc. is proposed. We group sets of features together into

hand defined semantic groups representing texture, position, etc.[1] In this way (initial) feature selection within these semantic groups is performed during continuous learning, and feature selection and context identification between the groups is performed during the symbolic learning stage.

For each semantic group, a set of example feature vectors is partitioned into classes using a graph partitioning method (an extension of [22]), which also acts as a feature selection method within the semantic group (full details appear in [16]). The number of clusters is chosen automatically based on a cluster compactness heuristic.

Once a set of examples is partitioned, the partitions may be used as supervision for a conventional supervised statistical learning algorithm such as a Multi-Layer Perceptron, Radial Basis Function or Vector Quantisation based nearest neighbour classifier (the latter is used in our implementation). This allows for the construction of models that encapsulate the information from the clustering in such a way that they can be easily and efficiently applied to novel data. These models are used to generate training data suitable for symbolic learning. For each object identified by the attention mechanism, a (symbolic) property is associated with it for each semantic group. Figure 2 shows example 'static' frames and corresponding symbolic data streams.



(a) Snap (b) Paper-Scissors-Stone

Figure 2: Example audio-visual input and symbolic representation

[1] This work uses a 96D rotationally invariant texture description vector (based on the statistics of banks of Gabor wavelets and other related convolution based operations), and a 2D position vector only.

2.3 Attention, learning and classification for audio

The attention mechanism for the audio input is based on the energy of the signal. Non-overlapping windows are formed each containing 512 samples, which is the power of 2 (needed for the Fourier transform) closest to a frame of video. (The audio is sampled at $8172Hz$.) The energy for each window is calculated as the sum of absolute values of the samples. The start of an utterance is detected when the energy of a window is greater than a fixed threshold. Then, each utterance can be represented as a sequence of consecutive windows for which the energy is over the threshold.

Spectrum analysis is performed on each detected window W_n, resulting in S_n the absolute value of the Fourier transform of W_n. The dimensionality of each spectrum S_n is reduced[2] from 512 to 17 by histogramming. Reducing the spectrum to this dimensionality makes the clustering robust to variations in the pitch of the voice [14]. Each utterance detected is then represented by a temporal sequence of L reduced-dimensionality windows. L is chosen such that it is equal to the length of the shortest utterance (in windows) in the training set. This is achieved by resampling the temporal sequence of reduced-dimensionality windows which represent an utterance. The utterances are now of identical length, and kmeans clustering is performed on the set of utterances several times with different numbers of clusters. The optimal number of clusters (C) to use is automatically chosen such that the ratio between the mean distance of each utterance to the centre of the closest cluster and the mean distance between all the cluster centres is minimised. Using C clusters, each utterance of the training set is classified (nearest cluster centre) to create a symbolic data stream as shown in Figure 2; thus an utterance may be represented symbolically as one of `word1,word2,...`. This method for automatically choosing the number of clusters does tend to over-cluster the data (too many clusters are created), yet this is dealt with by creating equivalence classes between the utterances, as discussed in Section 2.5.

2.4 Symbolic learning using Inductive Logic Programming

The previous sections described how models are learned that can convert continuous sensory input into a symbolic data stream in an unsupervised way. Learning models of the spatio-temporal structure of the resultant (possibly noisy) symbolic streams obtained is our goal, i.e. to learn a model of any implicit temporal protocols presented by the scene. Structure in such streams differs greatly from the structure learned by our lower level processes, in that the data consists of variable numbers of objects (and thus a variable length list of state descriptions is available). In addition, concepts such as reflexivity (equality of certain properties), symmetry and transitivity exist in the scenarios. These concepts cannot be captured by purely statistical learning methods, such as those used for low-level learning. An inductive logic programming approach is employed, implemented using Progol [18]. Progol allows a set of positive examples to be generalised by inductively subsuming the data representations by more general data representations/rules (with the aim of reducing representational complexity, without over-generalising). Crucial in any inductive learning approach is the way in which data is represented. Progol aims to reduce representational complexity using a search procedure. In realistic scenarios, a search of all

[2] The dimensionality is reduced to 17; this depends upon the rate that the audio is sampled ($8172Hz$), the fundamental pitch frequency of the human voice (average around $275Hz$) and the size of the window used (512 samples equivalent to windows at $16Hz$). $275/16$ is approximately 17.

possible data representations is not possible, and Progol must be guided by rules (mode declarations) that define the general form of the solution. Figure 2 shows examples of symbolic streams, from which rules are learned. For Progol, both the input data and output generalisations are in Prolog format, which allows straightforward incorporation of these rules into a Prolog program (see Section 2.6).

The general form of the desired solution is to generalise the "action" which occurs. In this case these are utterances. Thus in Progol's mode declarations it is stipulated that the generalisations must contain `action(utter,Word,Time)` in the head of all the rules. Little restriction is placed on the form of the bodies of the rules. The bodies capture the important features that must be present in the symbolic perceptual data stream in order for an action to be performed. Examples of the rules learned for the snap game are shown in Figure 3.

2.5 Building equivalence classes of utterances

The generalisation rules found by Progol are used to construct equivalence classes among utterances, since the method for utterance clustering is prone to cluster into more than the true number of clusters as mentioned above. The procedure for generating equivalence classes is based on the hypothesis that rules with similar bodies (encoding the *perceptual* inputs) are related to equivalent utterances in the rule heads (the *action* outputs). There are many possible ways of defining similarity in logic programs [9]. In this work, however, similarity is understood as classical unification of terms [3].

To construct equivalence classes, firstly, every pair of input rules whose heads are of the form `action(utter,word,time)` are checked for whether their bodies unify. Clauses with empty bodies are discarded as they do not provide any evidence for equivalence. For every pair of clause heads whose bodies unify, a predicate `equiv/2` is created, stating the hypothesis of equivalence between the utterances in their arguments. For instance, let `action(utter,w_i,t_x)` and `action(utter,w_j,t_y)` be two clause heads with unifying bodies, then the predicate `equiv(w_i,w_j)` represents the hypothesis of equivalence between utterances `w_i` and `w_j`. Equivalence classes are created by taking the transitive closure of the relation `equiv/2`.

2.6 Inference engine for agent behaviour generation

The symbolic protocols learned by the Progol program are used by a Prolog program to form an inference engine that can be used to drive an interactive cognitive agent that can participate in its environment. With a small amount of additional housekeeping Prolog code this program has been made to take its input from the lower level systems using network sockets, and output its results (via a socket) to an utterance synthesis module, which simply replays an automatically extracted audio clip of the appropriate response (the one closest to the cluster centre). An audio-visual response from a virtual participant could be used here (and sometimes is); however this adds nothing to the science presented. In our system, a human participant is required to follow the instructions uttered by the synthetic agent (as there is currently no robotic element to our system).

Currently the rules produced by Progol (ordered from most specific to most general if

[3] Informally, two terms are unifiable if they have at least one common instance [2].

necessary[4]) directly form part of a Prolog program. We impose a limit of a single action generation per time step in the (automatic) formulation of this program. We are working on a rule interpreter which can handle a wider range of scenarios (multiple simultaneous actions, non-deterministic/stochastic outcomes, etc.), however this is not necessary for the scenarios presented in this paper.

3 Evaluation and results

All experiments have been performed from live audio-visual input. Firstly a period of 'training' is undertaken during which perceptual categories are learned: features are extracted and object/utterance models are constructed. All object/utterance descriptions are classified to produce a symbolic data stream. The amount of training data depends upon the activity being observed. A set of generalisations are learned using Progol, and these are read into the Prolog inference engine module of our perceptual system. Once *object, utterance, event* and *protocol* models have been learned, the autonomous agent can respond to visual input in a game-playing phase. Figure 3 shows a rule-set which perfectly and concisely represents the protocol of the game snap. It can be seen from the `snap` rule (the third rule) that the concept of property equality (reflexivity) can be used in the generalisation of the training data. In this example, the two `Bs` indicate that the texture categories of the two objects must be the same for a snap to occur.

```
action(utter,[w4,w3],A) :- state([],A).
action(utter,[w4,w1],A) :- state([[B,C]],A).
action(utter,[w5,w2,w3],A) :- state([[B,C],[B,D]],A).
action(utter,[w2,w1],A) :- state([[B,C],[D,E]],A).
```

Figure 3: Example of a set of symbolic rules for the Snap game. Actions are sequences of utterances, where `wi` is an utterance corresponding to an utterances in an equivalence class. (`w1` = one, `w2` = collect, `w3` = two, `w4` = roll, `w5` = snap)

Two sets of objects have been used in the games. Snap is played with two dice, from which six texture categories are formed, one for each dice face. The attention mechanism for key-frame identification and object detection works with over 98% accuracy (minimal noise), very occasionally tracking two objects as one when rolled very close to each other. Empirical evaluation of the dice classification scheme has shown it to perform over 90% correct classification on a single dice and over 80% correct classification of two dice in a scene. The cards used in paper-scissors-stone are rather different to each other, and over 99% classification has been achieved throughout trials. Classification of the audio signal into utterances works 99% correctly, with typically only one or two utterances misclassified out of a set of two hundred, although this is into a greater number of clusters than the minimum possible (for which equivalence classes are formed after rule generalisation, see Section 2.5). Forcing clustering into the 'true' number of clusters using this methods was found to give poor classification. The construction of equivalence classes among utterances forms classes in which the members of each class correspond to the same vocal

[4]In the case that the body of one rule is a specialisation of another, the most general rule is moved below the most specific one in the ordering (if not the case already). This may be determined automatically using a subsumption check on each pair of rule bodies. Otherwise rule ordering is as output by Progol.

	#events in training	% correct	comment
snap	36	50	Little data - learned roll-two and collect-one
	74	97	Learned the protocol, snap-collect instead of snap-collect-two
	106	97	Learned the protocol, collect-two instead of snap-collect-two
paper scissors	108	92	Learned all but two cases (out of six) of 'lose', and one draw
stone	158	94	Learned an additional incorrect 'lose', and missing a draw
	248	97	Learned all rules, though also an additional incorrect 'lose' rule.

Figure 4: Evaluation results

utterance. Each `word_i` is not always assigned to an equivalence class, for example, when all the action rules with `word_i` in the head have no bodies.

Figure 4 describes the results of learning the protocol of both games from real world data, using ILP. Each game has been learned using three sizes of training data sets (events are equivalent to time steps). The '% correct' column refers to the theoretical percentage of time that the perceptual agent provides the correct response from the learned symbolic rules. Since the agent is embodied in a live system, only a subset of possibilities would be presented in an empirical evaluation, which could produce unreliable results. Live testing of the system produces similar results to those presented, with the occasional misclassification of visual objects leading to an incorrect response. Video of the learning and execution phases of the cognitive agent can be viewed at: `http://www.xxx.xx.xx/xxxx`. The ILP generalisations degrade gracefully with noise, or when there is little data. Less-general rules are lost, rather than the entire process failing, as more noise is introduced. This is essential for future work involving incremental and iterative learning.

4 Discussion and conclusions

A framework for the autonomous learning of perceptual categories and symbolic protocols has been presented. It has been demonstrated that a set of object, utterance and temporal protocol models can be learned autonomously, that may be used to drive a cognitive agent that can interact in a natural (human-like) way with the real world. The symbolic representation used is explicitly grounded to the sensor data, since both the perceptual categories and symbolic protocols have been learned from audio and video input. Although our synthetic agent has no robotic capability, it can issue vocal instructions and participate in simple games. The combination of low-level statistical object models with higher level symbolic models has been shown to be a powerful paradigm.

5 Acknowledgements

This work was funded by the European Union, as part of the CogVis project.

References

[1] S. Aksoy, C. Tusk, K. Koperski, and G. Marchisio. Scene modeling and image mining with a visual grammar. In *Frontiers of Remote Sensing Information Processing*, pages 35–62. World Scientific, 2003.

[2] K. R. Apt. *From logic programming to Prolog*. Prentice-Hall, Inc., 1996.

[3] C. Bryant, S. Muggleton, C. Page, and M. Sternberg. Combining active learning with inductive logic programming to close the loop in machine learning. In *Proc. AISB Symposium on AI and Scientific Creativity*, 1999.

[4] Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue. *Survey of the State of the Art in Human Language Technology*. Cambridge University Press, 1996.

[5] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, 2000.

[6] A. Fern, R. Givan, and J. Siskind. Specific-to-general learning for temporal events with application to learning event definitions from video. *Journal of Artificial Intelligence Research*, 17:379–449, 2002.

[7] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning about objects through action - initial steps towards arfificial cognition. In *Proc. IEEE International Conference on Robotics and Automation*, volume 3, pages 3140–3145, 2003.

[8] J. L. Flanagan and L.R. Rabiner. *SPEECH SYNTHESIS*. Dowden Hutchinsonand Ross, Inc., 1973.

[9] F. Formato, G. Gerla, and M. I. Sessa. Similarity-based unification. *Fundamenta Informaticae*, 40:393 – 414, 2000.

[10] S. Hargreaves-Heap and Y. Varoufakis. *Game Theory, A Critical Introduction*. Routledge, 1995.

[11] S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[12] Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. on Patterna Analysis and Machine Intellegence*, 22(8):852–872, 2000.

[13] D. Kazakov and S. Dobnik. Inductive learning of lexical semantics with typed unification grammars. In *Oxford Working Papers in Linguistics, Philology, and Phonetics*, 2003.

[14] Eric Keller. *Fundamentals of speech synthesis and speech recognition: basic concepts, state-of-the-art and future challenges*. John Wiley and Sons Ltd., 1994.

[15] D. R. Magee. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 20(8):581–594, 2004.

[16] D. R. Magee, D. C. Hogg, and A. G. Cohn. Autonomous object learning using multiple feature clusterings in dynamic scenarios. Technical Report School of Computing Research Report 2003.15, University of Leeds, UK, 2003.

[17] D. Moore and I. Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *Proc. AAAI National Conf. on AI*, 2002.

[18] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[19] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[20] J.M. Siskind. Visual event classification via force dynamics. In *Proc. AAAI National Conference on AI*, pages 149–155, 2000.

[21] M. Sternberg, R. King, R. Lewis, and S. Muggleton. Application of machine learning to structural molecular biology. *Philosophical Transactions of the Royal Society B*, 344:365–371, 1994.

[22] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.