

Recognising High-Level Agent Behaviour through Observations in Data Scarce Domains

Rolf Hugh Baxter

A dissertation submitted for the degree of Doctor of Philosophy

Heriot-Watt University

School of Engineering and Physical Sciences

July 2012

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that the copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the prior written consent of the author or of the University (as may be appropriate).

Abstract

This thesis presents a novel method for performing multi-agent behaviour recognition without requiring large training corpora. The reduced need for data means that robust probabilistic recognition can be performed within domains where annotated datasets are traditionally unavailable (e.g. surveillance, defence). Human behaviours are composed from sequences of underlying activities that can be used as salient features. We do not assume that the exact temporal ordering of such features is necessary, so can represent behaviours using an unordered “bag-of-features”. A weak temporal ordering is imposed during inference to match behaviours to observations and replaces the learnt model parameters used by competing methods. Our three-tier architecture comprises low-level video tracking, event analysis and high-level inference. High-level inference is performed using a new, cascading extension of the Rao-Blackwellised Particle Filter. Behaviours are recognised at multiple levels of abstraction and can contain a mixture of solo and multi-agent behaviour. We validate our framework using the PETS 2006 video surveillance dataset and our own video sequences, in addition to a large corpus of simulated data. We achieve a mean recognition precision of 96.4% on the simulated data and 89.3% on the combined video data. Our “bag-of-features” framework is able to detect when behaviours terminate and accurately explains agent behaviour despite significant quantities of low-level classification errors in the input, and can even detect agents who change their behaviour.

Acknowledgements

I would like to express my thanks to:

- My supervisors Professors David Lane and Ruth Aylett who have guided me throughout my Ph.D. study. Their guidance in the early stages was particularly valuable in helping me to focus my ideas and I am greatly appreciative for their support. This is especially true as the research topic moved away from the Lab's core focus of autonomous underwater systems.
- Professor Yvan Petillot and Dr. Neil Robertson for their support as colleagues and mentors. Both have made themselves available as sounding boards to bounce ideas off and provided technical advice, especially regarding the field of computer vision. They have also been of great assistance in helping me to write better scientific publications and I would like to thank them again for their contributions.
- I am indebted to the UK Ministry of Defence for funding this research, and to my colleagues at DSTL who have supported the project. Peter Houghton was especially helpful with his suggestions and comments regarding operational requirements and scenarios.
- I would also like to express my thanks to Nicolas Valeyrie for our many impromptu discussions on particle filtering, and to Wassit Limprasert for allowing me to use his person tracker as part of my research.
- My special thanks to my wife Alaina who has been patient with me throughout the PhD., has lifted my spirits and assisted in all the ways she could whenever I have been busy trying to meet deadlines. And finally, I would like to thank my parents for their support and encouragement throughout my studies, without which I would have never made it this far.

ACADEMIC REGISTRY

Research Thesis Submission



Name:	Rolf Hugh Baxter		
School/PGI:	Engineering and Physical Sciences		
Version: <i>(i.e. First, Resubmission, Final)</i>	Final	Degree Sought (Award and Subject area)	Doctor of Philosophy Electrical, Electronic and Computer Engineering

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

- 1) the thesis embodies the results of my own work and has been composed by myself
- 2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
- 3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
- 4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
- 5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.

* Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.

Signature of Candidate:		Date:	
-------------------------	--	-------	--

Submission

Submitted By <i>(name in capitals)</i> :	
Signature of Individual Submitting:	
Date Submitted:	

For Completion in the Student Service Centre (SSC)

Received in the SSC by <i>(name in capitals)</i> :			
Method of Submission <i>(Handed in to SSC; posted through internal/external mail):</i>			
E-thesis Submitted (mandatory for final theses)			
Signature:		Date:	

Contents

	Page
1 Introduction	1
1.1 A Common Terminology	7
1.2 Aims	8
1.2.1 Extensions	9
1.3 Contributions	9
1.4 Thesis Roadmap	12
2 Literature Survey	15
2.1 Probabilistic Inference Techniques	16
2.1.1 Bayesian Networks	16
2.1.2 Markov Models	17
2.1.3 Hidden Markov Models	18
2.1.4 Dynamic Bayesian Networks	18
2.1.5 Bayes Filter	20
2.1.6 Sequential Importance Sampling (SIS)	21
2.1.7 SIS with Resampling	25
2.1.8 Rao-Blackwellisation	27
2.1.9 Summary	31
2.2 Video Processing Techniques	32
2.2.1 Foreground Detection	32
2.2.2 Object Classification	34
2.2.3 Object Tracking	35
2.3 Trajectory Models	38
2.4 Semantic Models	40
2.4.1 Logic Models	41
2.4.2 Constraint Satisfaction	42
2.4.3 Grammars	44
2.5 State Models	46
2.5.1 Hidden Markov Models	47
2.5.2 Dynamic Bayesian Networks	50
2.5.3 Summary	54
2.6 Feature Models	55

2.7	Multi-agent Models	58
2.8	Conclusions	60
2.8.1	Video Processing	60
2.8.2	Probabilistic Behaviour Recognition	61
2.8.3	Alternative Recognition Approaches	62
2.8.4	Feature Based vs. Sequence Based	64
3	Feature-Based Behaviour Recognition	65
3.1	Principles of The ‘Bag-of-Features’	66
3.1.1	Worked Example	69
3.2	Bayes Network Representation	69
3.3	Adding A Temporal Model	72
3.4	Summary	75
4	Efficient Inference	76
4.1	Basic Algorithm	77
4.1.1	Sample Regeneration	80
4.1.2	Summary	89
4.2	Advanced Algorithm	90
4.2.1	Partial Behaviours	92
4.2.2	Feature Repetition	93
4.2.3	Behaviour Concatenation	94
4.3	Summary	94
5	Hierarchical Recognition	97
5.1	Representation	99
5.2	The Hierarchical Filter	102
5.3	Example	103
5.4	Summary	105
6	Multi-agent Behaviour Recognition	108
6.1	Combing Filter Results	111
6.2	Identifying Multi-agent Behaviour	113
6.3	Predicting Agent Behaviour	122
6.4	Limitations	123
6.5	Summary	124
7	Implementation	125
7.1	Image Processing	127
7.1.1	Object Detection and Tracking	127
7.1.2	Primitive Feature Detection	129
7.2	Reasoning	131

7.2.1	Complex Feature Detection	131
7.2.2	Collaborator Detection	133
7.3	Operator Interaction	133
7.3.1	Prediction Criteria	135
7.3.2	Prediction Detail	136
7.4	Primitive Feature Simulator	137
7.5	Summary	138
8	Experimental Evaluation via Simulation	142
8.1	Recognition Accuracy	146
8.1.1	Post-Sequence Prediction	147
8.1.2	In-Sequence Prediction	148
8.2	Behaviour Variability	152
8.2.1	Unknown Behaviours	152
8.2.2	Ordering Mutation	155
8.2.3	Behaviour Switching and Concatenation	157
8.2.4	Superfluous Activity	162
8.3	Effect of Feature Classification Errors	164
8.4	Effects of Scaling	166
8.4.1	Number of Particles	167
8.4.2	Number of Agents	171
8.5	Summary	172
9	Experimental Validation using Video	178
9.1	The Video Based Inference Process	179
9.2	Event Recognition Performance	185
9.3	Behaviour Recognition Performance	188
9.4	Summary	191
10	Discussion	192
10.1	Feature Based Recognition	192
10.1.1	Earlier Prediction	194
10.1.2	Heuristic Particle Weights	196
10.1.3	Behaviour Confusion	196
10.2	Real-World Application	197
10.2.1	Explaining Behaviour	198
10.2.2	Event Detection Comparison	201
10.2.3	Behaviour Detection Comparison	202
10.2.4	Runtime Scaling	203
10.2.5	Complexity of Human Behaviour	205
10.3	Generic Application	209
10.4	Summary	210

11 Conclusions and Future Work	212
11.1 Conclusions	212
11.2 Future Work	215
11.2.1 Collaborator Detection	215
11.2.2 Repetitive Behaviour	216
11.2.3 Active Vision	217
11.2.4 Ontology Integration	218
11.2.5 Comparing Approaches	218
11.2.6 Behaviour Similarity	218
11.2.7 Video Event Detectors	219
11.2.8 Standardised Dataset	219
11.3 Closing Remarks	220
Appendix A	221
Bibliography	227

List of Tables

2.1	Transition matrix and prior probabilities for a simple Markov Model . . .	17
3.1	Prior, Conditional and Transition probabilities between time steps $t - 1$ and t for the DBN in Figure 3.4	73
6.1	Worked example of filter merging	114
6.2	Example probability distributions for two agents.	116
7.1	The primitive feature detection modules	129
7.2	Goal behaviours used in the evaluation	132
8.1	Summary of evaluation behaviours	143
8.2	Classification error distribution model	143
8.3	Primitive feature true-positive detection rates	143
8.4	Behaviour confusion matrix for post-sequence prediction	148
8.5	Behaviour confusion matrix for in-sequence prediction	150
9.1	Tracking errors are the normal cause of missed classifications, and are partially responsible for classification errors.	183
9.2	The parameters used with the tracker and event detection modules.	185
9.3	The number of instances of each complex behaviour in the datasets.	189
10.1	Comparison of luggage detection (<i>PlaceObject</i>) accuracy	201
10.2	Comparison of the Abandon Object (<i>AO2</i>) detection accuracy with com- peting techniques	202
10.3	Comparing precision with Chai and Yang’s MG-Recogniser	208

List of Figures

1.1	Taxonomic structure of a behaviour	4
1.2	The ‘Watched Item’ behaviour is comprised of five activities performed by two agents	5
1.3	Two agents switch from the ‘Watched Item’ behaviour to the ‘Passing Through’ 1 & 2 behaviours	10
1.4	An agent performing the ‘Passing Through 2’ behaviour, including repetitious elements that are not part of the behaviour definition.	11
2.1	A simple Bayes Network	16
2.2	The Hidden Markov Model	18
2.3	Example two-slice Bayes network defining $P(Z_t Z_{t-1})$. Shaded nodes are latent, while unshaded nodes are directly observable.	19
2.4	A simple target trajectory, and estimated trajectories by two different particles	25
2.5	Normalised particle weights for five observations	26
2.6	Example Dynamic Bayes Net for the robot localisation and mapping problem from [38]	28
2.7	Incorporating the latest observation into the RB-Posterior reverses the edge between y_t and z_t	29
2.8	A Goal Consistency Graph in which observations (rectangles) can be explained by different goals. Inconsistent goals are removed from the graph allowing predicates to be formed. Link labels (e.g. S1) indicate action order.	41
2.9	Defining the ‘Aircraft_Arrival_Preparation’ behaviour in the Video Event Description Language, taken from [46].	43
2.10	The scene is segmented into four zones to assist inference (taken from [106])	43
2.11	The multi-modal SEER architecture from [112]	48
2.12	Blaylock and Allen decompose high-level behaviour into lower-level sub-goals and actions.	48
2.13	A Cascading Hidden Markov Model	49
2.14	The environment and the trajectory of a person, taken from [24]	51
2.15	The region and behaviour hierarchy from [24]	51
2.16	DBN representing a person’s outdoor movements during everyday activity. The upper level is used for novelty detection, and the middle layer estimates the user’s goal and the trip segments he or she is taking to get there. The lowest layer represents the user’s mode of transportation, speed, and location. Two time slices, k and $k - 1$, are shown. (Taken from [83]).	53

2.17	The basic concept of image composition taken from [22]. A region in the new image is considered likely if it has a large enough contiguous region of support in the database. New ‘normal’ images can thus be inferred from the database even when they have not been previously observed.	57
2.18	A partial behaviour description of an American Football play from [65]. An agent (obj1) ‘snaps’ to the other team’s Quarterback and block their pass. Because one of the agents is explicitly defined the algorithm does not need to consider the actions of all agents when trying to match this behaviour.	59
3.1	(a) The temporal representation of the Watched Item Behaviour. (b) The Bag-of-Features representation of the Watched Item behaviour.	67
3.2	a) A Bayesian Network for representing behaviour using features. b) The generalised form	70
3.3	The addition of an ‘Interruption’ node I into the bag-of-features DBN	72
3.4	The full DBN for ‘bag-of-features’ inference	72
3.5	A visualisation of the state transition model	74
4.1	Incorporating the latest observation into the RB-Posterior reverses the edge between A_t and D_t	78
4.2	A worked example of the basic inference algorithm (Part 1). Please refer to the text for a detailed discussion. Line numbers relate to Algorithm 4.1	86
4.3	A worked example of the basic inference algorithm (Part 2). Please refer to the text for a detailed discussion. Line numbers relate to Algorithm 4.1	87
4.4	The minimal explanation changes as observations arrive. At the first observation (<i>EnterAgent</i>) <i>PT1</i> (Passing Though 1) gains the highest probability because it is the shortest explaining behaviour. By the second observation (<i>PlaceObject</i>) <i>PT1</i> can no longer explain the observations and reduces in probability, while <i>AO2</i> (Abandon Object 2) becomes more likely as the next shortest behaviour that can explain the observations. This happens again at observation three where <i>PT2</i> (Passing Through 2) becomes and remains the most probable explanation.	88
4.5	The cumulative distribution function used by Algorithm 4.1.	89
4.6	Example particle state growth for two particles and two behaviours while observing a 3 features sequence.	91
5.1	An example of how goal behaviours are hierarchically decomposed into sub-goals and activities	98
5.2	To model a set of behaviours requires two stacks of DBNS. One set for the goal behaviours, and one set for the sub-goals.	102
5.3	Algorithms 5.2 and 5.3 produce two levels of particle filters. The level 1 filter approximates the probability density of the agent’s high-level behaviour, while the level 2 filters approximate the probability of different sub-goals. Note that for k high-level behaviours there are also k level 2 filters, but only one level 1 filter. Estimates from the level 2 filters are utilised by the level 1 weighting step.	106
6.1	The <i>Theft</i> behaviour has two distinct agent roles	109

6.2	The probability of accepting a negative transition (-0.02) as the temperature changes	119
6.3	A simple example trace of the simulated annealing algorithm.	121
7.1	The 3-Layer Implementation Framework	126
7.2	Primitive Feature Simulator Example: Individual agent behaviours are merged into a single observation stream upon which the error model is then applied to insert classification errors.	139
8.1	F-Score for each behaviour prediction	147
8.2	In-sequence prediction F-Score vs. post-sequence F-Score	149
8.3	Example observation trace in which an erroneous multi-agent feature (E1,2) occurs.	150
8.4	The density of particle weights associated with two explanations (complex features) for the set of primitives: $\{PlaceObject, RemoveObject\}$. One third of the particle weights are assigned to the assumption that <i>RemoveObject</i> was a false positive and explain the <i>PlaceObject</i> primitive with a <i>LeaveObject</i> complex. The remaining two thirds of particle weight assume that both observations were true positives and explain them via the complex feature <i>TransientObject</i>	151
8.5	Comparing F-Scores when unknown behaviours are observed. Unknown behaviours have minimal impact on recognition performance. Of the 30 unknown (random) behaviours only 4 (13%) were misclassified, with no classifications made for the remaining 87%.	154
8.6	The primitive features in the <i>WI</i> and <i>AO1</i> behaviours being reordered	156
8.7	Probability density for each complex feature as observations are made (Behaviour: <i>WI</i>). (a) is consistent with the model order, (b) shows an alternative order. When <i>PlaceObject</i> and <i>ExitAgent</i> are observed contiguously the <i>LeaveObject</i> feature (which is comprised of <i>PlaceObject</i> and <i>ExitAgent</i>) has a higher probability density than when they are separated by another observation.	158
8.8	Probability density for each complex feature as observations are made (Behaviour: <i>AO1</i>). (a) is consistent with the model order, (b) shows an alternative order. When <i>PlaceObject</i> and <i>ExitAgent</i> are observed contiguously the <i>LeaveObject</i> feature (which is comprised of <i>PlaceObject</i> and <i>ExitAgent</i>) has a higher probability density than when they are separated by two observations.	159
8.9	When two behaviours are observed directly after one another the second behaviour suffers a 24% loss of recall. In combination with a 4% drop in precision this leads to an overall F-Score of 0.73.	161
8.10	Pairs of superfluous activities (e.g. $\{PlaceObject, RemoveObject\}$) cause a reduction in mean F-Score from 0.92 to 0.64.	164
8.11	Effect of primitive feature classification errors on recognition F-Score [update to use levels]	166
8.12	Effect of number of particles on F-Score	169
8.13	Effect of number of particles on runtime	170
8.14	Thread growth rates	172

9.1	Camera calibration data for PETS scene S1-T1-C (Camera 3) showing the x,y, origin.	180
9.2	Example output from the person tracker for PETS scene S1-T1-C (Camera 3) . .	180
9.3	Person tracking information is provided to the Agent Tracker which detects and <i>EnterAgent</i> event. This event is output as a Primitive Feature and provided as input to the bag-of-features inference algorithm.	182
9.4	The video processing components of the framework provide a stream of primitive feature observations to the Bag-of-features inference algorithm. The primitive feature simulator also provides such a stream and incorporates synthetic classification errors to mimic video processing failures.	184
9.5	Video event detection precision	187
9.6	Frames from PETS scene S1-T1-C. A group of agents entering the scene together and travelling as a group causes tracking errors and false <i>FormGroup</i> detections.	187
9.7	Video event detection recall	188
9.8	Behaviour recognition accuracy on the PETS data	190
9.9	Two examples of the person tracker misclassifying luggage as people on the PETS dataset.	190
9.10	Behaviour recognition F-Scores on the HW data	191
10.1	In-sequence prediction results plotted in ROC space. Optimal x,y coordinates: (0,1)	193
10.2	F-Scores for each behaviour <i>B</i> as the number of features required for prediction is altered. When predictions are made with one outstanding feature expected there is a 0.28 reduction in mean F-Score from 0.92 to 0.64.	195
10.3	Example explanation from the PETS dataset (Scenario 4). The text reads: “Agent H5 has left luggage with agent H7, but they are not known companions. H5 may have abandoned an object. The item was placed around frame 878 and abandoned around frame 1941. (High certainty)”	199
10.4	Runtime vs. number of agents with fixed 220 particles	204
A.1	Structure of the Passing Through 1 behaviour	223
A.2	Structure of the Passing Through 2 behaviour	223
A.3	Structure of the Watched Item behaviour	224
A.4	Structure of the Abandon Object 1 behaviour	224
A.5	Structure of the Abandon Object 2 behaviour	225
A.6	Structure of the Theft behaviour	225
A.7	Structure of the Hand-Off behaviour	226

Publications

Portions of the work described in this thesis has also appeared in:

Conference papers

- Real-time event recognition from video via a “Bag-Of-Activities”, R. H. Baxter, N. M. Robertson and D. M. Lane, in *Proceedings of the UAI Bayesian Modelling Applications Workshop*. July 2011
- Probabilistic Behaviour Signatures: Feature-Based Behaviour Recognition in Data-Scarce Domains, R. H. Baxter, N. M. Robertson and D. M. Lane, in *Proceedings of the 13th International Conference on Information Fusion*. July 2010
- Recognising Agent Behaviour During Variable Length Activities, R. H. Baxter, D. M. Lane and Y. Petillot, in *Proceedings of The 19th European Conference on Artificial Intelligence*. August 2010
- Behaviour Recognition for Spatially Unconstrained Unmanned Vehicles, R. H. Baxter, D. M. Lane and Y. Petillot, in *Proceedings of The IJCAI Workshop on Plan, Activity and Intent Recognition (PAIR)*. July 2009.

Journals (submitted, under review)

- Recognising High-Level Agent Behaviour in Data Scarce Domains, R. H. Baxter, D. M. Lane, N. M. Robertson, submitted to *Systems, Man and Cybernetics, Part B: Cybernetics, IEEE Transactions on*. December 2011.

Chapter 1

Introduction

“behaviour: *noun*

1 manner of behaving or conducting oneself

...

4 the action, reaction, or functioning of a system, under normal or specified circumstances ”

Collins English Dictionary [3]

The ability to recognise agent behaviour through observation is a skill each of us possess. We recognise the behaviours of others frequently throughout each day; other road users on the way to work, colleagues and family members performing daily routines, yet this seemingly trivial task presents significant challenges for computers. This is because the ability to recognise behaviour requires several different skills: The ability to sense the environment, the ability to infer action purpose and the ability to apply learnt knowledge. It should be no surprise therefore that the goal of behaviour recognition has attracted researchers from many different fields.

The ability to learn behavioural models and recognise complex behaviour has been of particular interest to artificial intelligence researchers, with significant publications on plan and goal recognition as early as the 1980s (e.g. [68]). Although early work primarily focused on synthetic data with toy problems [69, 79], recent advances have been increasingly applied to real-world scenarios. In part, this has been fuelled by the technical advances that have made high-quality electronics so readily available. Behaviours have been recognised from consumer devices such as wireless network [27, 147] and GPS receivers [82, 83], as well as from a vast array of video cameras [24, 107, 108, 11, 15, 76, 35].

The wide application potential of behaviour recognition technology is frequently cited as a motivating factor for research and includes:

- Technologies for long-term healthcare: Systems that can recognise and monitor activities of daily living such as dressing, eating and leisure can help to provide assurance of patient well-being and identify abnormalities [75].
- Automated surveillance: Remote sensing devices such as radar and closed circuit television (CCTV) have been extensively deployed for public and asset security, yet it is openly acknowledged that human observers are far from optimal [36]. The ability to automatically detect behaviour of interest has the potential to reduce human sensory overload and to improve recognition performance.
- Video archiving: Similarly, the ability to automatically detect behaviours within video has potential benefits for automated video archiving [57]. Video search techniques currently rely on non-visual meta-data such as tags and transcripts, while visual information could provide valuable information for archiving and retrieval purposes.
- Autonomous vehicles: There is great interest in providing increased situation awareness to autonomous vehicles for commercial, civilian and military applications [6]. Situation awareness is defined as the ability to perceive elements of the environment and to comprehend their meaning [42]. Such abilities are important for the deployment of autonomous vehicles within multi-agent environments for both control and safety reasons.
- Entertainment/Training: Artificial intelligence techniques are increasingly being utilised by the video entertainment industry with Microsoft's Kinect and Nintendo's Wii being prime examples of behaviour recognition technologies at work in the home environment. There is also great potential for improving the complexity of virtual agents in synthetic environments (e.g. military simulation) [135, 134].

The work described herein was primarily aimed at developing deployable surveillance technology for the project's sponsor, the UK Ministry of Defence. Although this application imposed a number of operational constraints the algorithms developed maintain a generality that enables their utilisation in a wider application potential.

Despite a wide-spread interest in behaviour recognition, the situation aware computers envisaged in the 1960s (e.g. HAL¹) are still far from reality. Early work in the field often

¹HAL (Heuristically programmed ALgorithmic computer) was an artificial intelligence system in Arthur C. Clarke's science fiction novel 2001: A Space Odyssey

looked at toy problems with simplistic, manufactured observations [69, 79]. However, many early approaches were unable to deal with noisy, uncertain environments and this has generally led to a demise of early techniques in modern research. More significant progress has been made in the last decade since researchers started building real-world systems with sensor observations. Activity recognition is one area where techniques have flourished and generally considers short-term goal behaviours such as standing, walking and running [123]. These fundamental building blocks have led to slightly more complex activities, such as the inference of transportation routines by Liao *et al.* [83]. Their approach fuses an agent's GPS location with additional information to infer their mode of transport (e.g. car, bus), and is able to predict a person's goal location (e.g. home).

Behaviour recognition has also progressed within the video and image processing community [75]. Much of their work has focused on the detection and tracking of humans in video, and the recognition of simple activities which are commonly termed 'events' in vision literature. An example event might be an 'abandoned object', identified through the fact that a tracked person has moved some distance away from a bag they placed. There are also examples of more advanced events being detected involving multiple agents, although these approaches tend to rely on rules that look for the correlation of several different activities within certain spatio-temporal constraints.

Unlike much of this previous research, this thesis focuses on recognising high-level behaviours from sensor data. A high-level behaviour can be most easily described as a taxonomy structure, often termed a goal tree, in which an agent's goal is decomposed into smaller and smaller components until a sequence of actions is reached. This is illustrated in Figure 1.1 and can be related to the inverse problem of hierarchical computer planning. In planning one starts with a goal and attempts to derive a set of actions by which that goal is achieved. High-level behaviour recognition is thus the reverse of this process: given a set of (observed) actions, determine the most likely agent goal-tree.

To give a concrete example, Figure 1.2 demonstrates the 'Watched Item' behaviour, commonly seen at public transport hubs. It is composed of the following observable actions:

- The two agents enter the scene together as a group
- One agent (blue) places a luggage item on the floor
- The group of agents 'splits' as one agent (blue) moves away from the other
- The blue agent exits the scene, leaving their luggage 'watched' by the other agent (green)

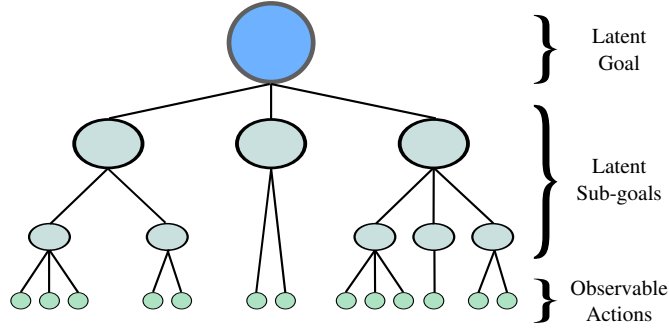


Figure 1.1: Taxonomic structure of a behaviour

Recognising this high-level behaviour involves several challenging steps. First, the environment must be sensed and signal processing performed so that the agents can be detected and tracked. The video frames in Figure 1.2 show that two agents have been detected and are highlighted by blue and green ellipses. The low-level actions of the agents then need to be detected, such as entering the scene or placing an object on the floor. From these primitive actions the complex agent behaviours must be inferred, including their sub-goals and goals. Figure 1.2 includes the sub-goal *LeaveObject*, which is composed of two primitive actions. Over-all, the Watched Item behaviour is comprised of four sub-goals. Putting this example into the wider context, the objective of high-level behaviour recognition is to identify the sub-goals/goals being performed by an agent from a larger set of candidates.

Significant progress has been made in high-level recognition. Successful examples include the inference of office behaviours by Bui and Venkatesh [24], whose work we build upon in this thesis. They used Rao-Blackwellised Particle Filtering in combination with trained trajectory models to recognise behaviours such as ‘printing a document’, comprised of a person entering the scene, loitering near a computer and then moving towards a printer. Other successful approaches have been presented by Nguyen *et al.* and applied to similar behaviours [107, 108], but again, their work has been constrained by a reliance on trained trajectory models which are simply unavailable in some domains. Furthermore, in both of these examples only single agent behaviours are addressed, with no consideration of multi-agent behaviour. This constraint is common in much of the prior research (e.g. [68, 82, 83, 85, 18, 19, 20, 102, 103, 104, 78]).

It is unfortunate that the most robust approaches for behaviour recognition adopt trained probabilistic models, making them poorly suited to data-scarce domains. For instance, for video surveillance applications, annotated libraries of video do not exist for many interesting behaviours, while operational factors become problematic when dealing with military or counter-terrorism applications [66]. Furthermore, even when data can be col-

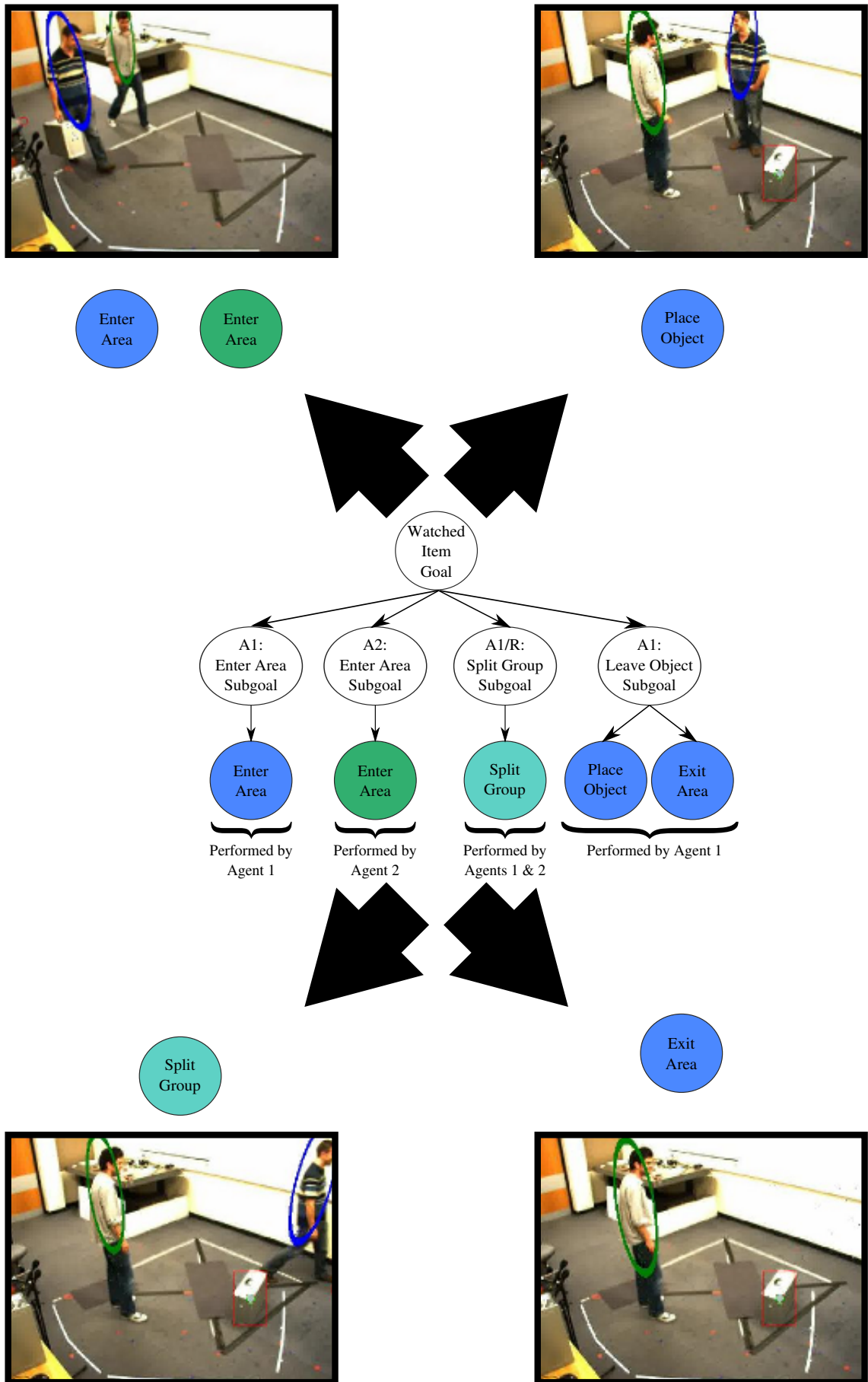


Figure 1.2: The 'Watched Item' behaviour is comprised of five activities performed by two agents

lected there is often a high cost associated. This issue is particularly relevant when collecting from real-world environments, where privacy concerns and configuration time can significantly impact collection.

In data-scarce domains there are primarily two alternative approaches that have been employed, although each approach sacrifices robustness in order to provide recognition. For example, in video surveillance applications it is not uncommon to manually specify semantic constraints, rules and relations (e.g. [46]), but these approaches are largely deterministic and lack convenient methods for handling observational uncertainty [75]. The second alternative; anomaly detection, detects abnormal behaviours and thus subtly changes the goal from behaviour recognition. Consequently, this approach has limited use where an application requires behaviour recognition. In other data-scarce domains a lack of training data has generally been overlooked. For instance, in counter-terrorism applications it is often assumed that probabilities can be gleaned from statistics or experts (e.g. [141, 66]), although there is very little consideration of how reasonable this assumption is or what implications it may have.

This thesis proposes that there is a better way to use probabilistic models in data-scarce domains and is fundamentally grounded upon the idea of an alternative behaviour representation that removes the need to learn temporal structure. Indeed, it poses the idea that the components of a behaviour can be used as salient features and is inspired by image based object detection. There is a clear difference between these two domains; images are static while behaviours are not, yet this evolution of behaviour is the very method by which a feature based approach works.

To model the ‘Watched Item’ behaviour (Figure 1.2) using traditional techniques each activity/action would be considered a state. The probability of moving from one state to the next would be represented by a transition probability, and these would be derived via a process of model training. In the absence of suitable training data such probabilities could be estimated by an expert as in [141], although this reliance is clearly limiting.

As an alternative representation, this thesis proposes that each activity be considered a feature and represents the behaviour as an unordered “bag of features” (cardinality: 1). If it is assumed that an agent is performing the ‘Watched Item’ behaviour (hence forth referred to as *WI*), then we have a set of features that we expect to see. It holds that as the agent is observed performing activities (features) the set of *expected* features reduces if observations are consistent with expectations. Thus at time step $t = 0$ each of the 5 features is expected. Let us assume that two agents are observed to enter the scene at $t = 1$. If the agents are indeed performing *WI* then the set of expected features

now reduces to $\{SplitGroup(A1, A2), PlaceItem(A1), ExitArea(A2)\}$. If one agent is observed to place a luggage item at $t = 2$ the set of expected features reduces further to $\{SplitGroup(A1, A2), ExitArea(A2)\}$. If an observation is inconsistent with the expected features then there is evidence that the agents may not be performing *WI*, and indeed, such an observation may support an alternative behaviour hypothesis.

Using this approach it is my thesis that we should be able to recognise behaviours without learning their temporal structure, and in doing so will be able to probabilistically recognise behaviours in data-scarce domains. This approach maintains an ability to reason about uncertainty while existing alternatives do not. Furthermore, a lack of fixed temporal structure means that observations can appear in an ‘unconventional’ order which is useful for wide-area, multi-sensor surveillance. In such an application different components of the plan might be performed at different times and by different people, yet this would have no effect on inference. It should also be noted that although the approach is applied to video surveillance in this research, there is no requirement to use video input. The approach could be supplied with event data from any number of sources (e.g. radar, lidar, intrusion detection systems), or even utilise multi-model observations. In contrast, very few other approaches could be applied in the same way due to their tight coupling with the data (e.g. [55, 10, 94, 73])

1.1 A Common Terminology

The multi-disciplinary nature of behaviour recognition research has led to a set of terms that are inconsistent, and sometimes conflicting. To eliminate confusion in the succeeding chapters several terms will now be defined as used within this thesis. Behaviours will be discussed in two contexts; high-level (complex) and low-level (primitive). A low-level behaviour will be considered isolated, and do not involve long-term temporal dependencies. This definition is broadly consistent with the use of ‘activity’ in prior research, where examples include *Agent Enters* and *Object Placed on Ground*. These primitive features can be seen in Figure 1.2 as being directly observable.

High-level behaviours will refer to sequences of activities that are temporally dependant and achieve some higher-level goal. For example, the high-level behaviour in Figure 1.2 (*Watching Item*) involves a number of hierarchical complex features. Similarly, the *Leave Object* feature is also considered complex because it is composed of two dependent features.

1.2 Aims

The goal of this thesis is to develop probabilistic inference techniques for high-level behaviour recognition in data scarce domains. There are three key aims of this work:

1. To design and implement an algorithm that allows high-level behaviour recognition without model training and is
 - (a) robust in noisy environments
 - (b) delivers a low false-positive rate
 - (c) can explain detected behaviour and communicate:
 - i. detection certainty
 - ii. key activity times
 - (d) can perform real-time inference
2. Establish the effect of low-level (activity) classification errors on high-level inference
3. To validate the approach using sensor data in a domain for which annotated training data is unavailable

1.2.1 Extensions

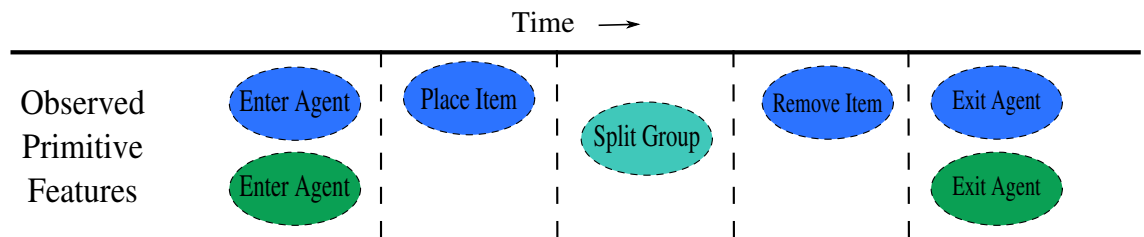
In addition to these aims there are several areas where extended goals have been identified. These goals are not key to the success of this work but extend the range of behaviours that can be detected and can be considered valuable extensions. They can be summarised as follows:

1. *Recognising switched and concatenated behaviour.* Switched behaviour is observed when an agent abandons one goal to pursue another leading to the partial observance of two different behaviours (Figure 1.3). Concatenation is the complete observance of two behaviours contiguously.
2. *Recognising repetitious behaviour.* This occurs when elements of the behaviour are repeated. This is demonstrated in Figure 1.4 in which the ‘Passing Through 2’ behaviour is being performed (consisting of the primitive features: *EnterAgent*, *PlaceObject*, *RemoveObject*, *ExitAgent*). The agent repeats two of the components (*PlaceObject*, *RemoveObject*).

1.3 Contributions

With respect to behaviour recognition the contributions of this thesis are:

1. A new algorithm for behaviour recognition that builds on existing research with Rao-Blackwellised Particle Filters [24]. Key to the approach is a new behaviour representation that considers activities as behavioural features and encapsulates this idea using Dynamic Bayesian Networks (DBN). A significant novelty of this DBN is that it is application independent, unlike many prior approaches. The algorithm is demonstrated within an automated visual surveillance framework, and we present preliminary results from both simulated data and real surveillance-like video. Using video data from the publicly available PETS 2006 dataset [139], in addition to further video collected by the author, we demonstrate the algorithm’s ability to generate accurate and detailed behaviour reports in real-time within a noisy, sensor-based environment. We achieve a mean recognition precision of 96.4% on the simulated data and 89.3% on the combined video data, while no parameter estimation is required.



Behaviours

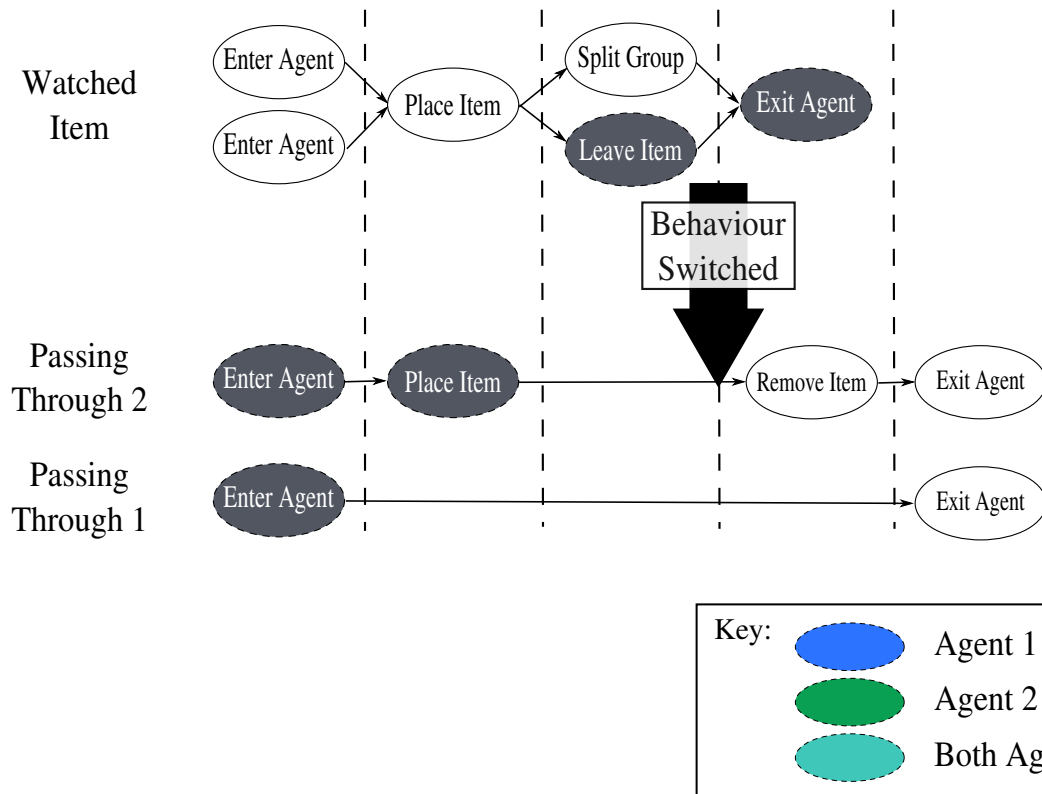


Figure 1.3: Two agents switch from the 'Watched Item' behaviour to the 'Passing Through' 1 & 2 behaviours

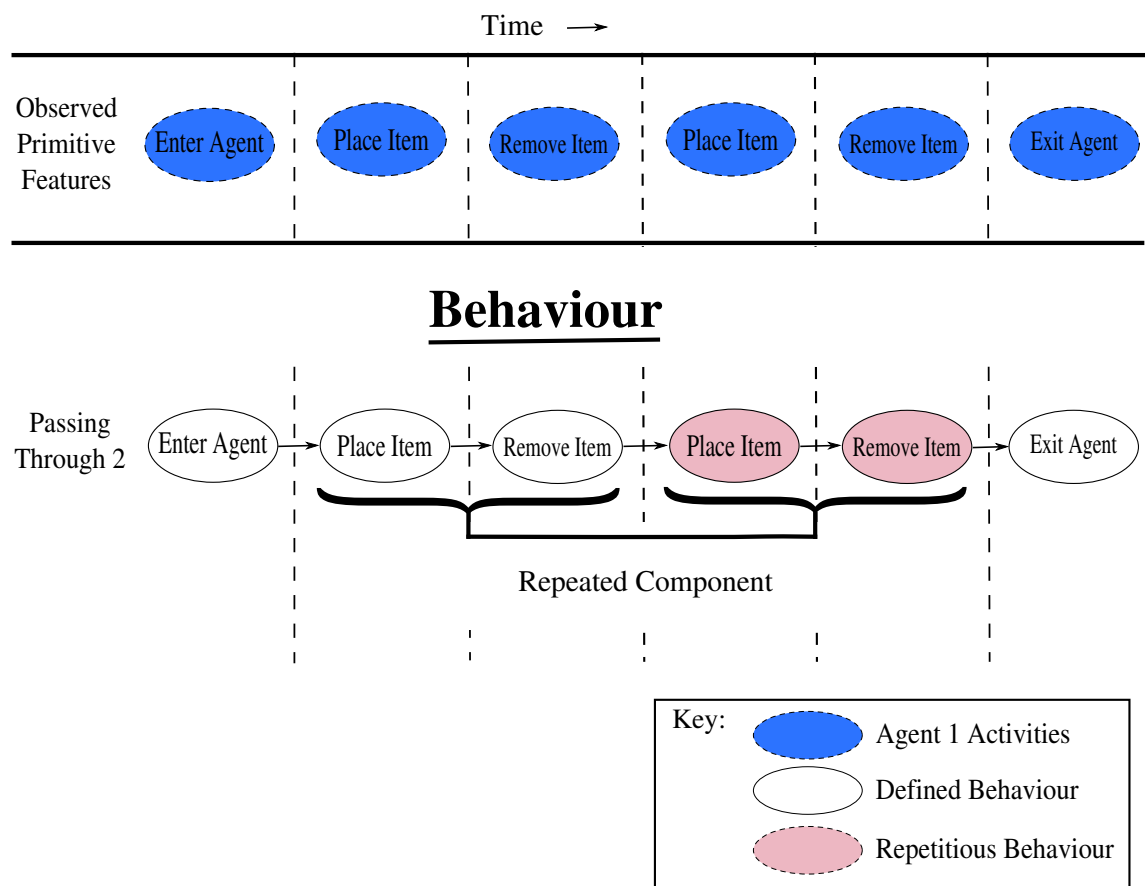


Figure 1.4: An agent performing the ‘Passing Through 2’ behaviour, including repetitious elements that are not part of the behaviour definition.

2. The algorithm is extended to model behaviour at multiple levels of abstraction (i.e. goals, sub-goals) and define multi-agent behaviour by specifying agent:sub-goal relationships. It is shown that the resulting model can detect agents acting in isolation as well as those involved in multi-agent behaviour without any prior knowledge of agent groups. This ability is demonstrated in our experiments which involve 3 solo behaviours and 4 multi-agent behaviours.
3. It is shown that the approach can recognise agents concatenating and switching between behaviours. The approach even remains robust when behaviours contain significant similarities, and performs comparably with other, trained, ‘multi-goal’ approaches. We achieve a mean precision of 88% for concatenated behaviour and 87% for switched behaviour at the cost of reduced recall (65% and 60% respectively).

This thesis also makes further contributions with respect to probabilistic reasoning:

1. The framework employs a hierarchicalisation procedure that involves the cascading of particle filters. This cascade functions in a similar way to cascaded Hidden Markov Models, where classifications from one model provide observations to another. Set into the context of particle filters, the probability estimates for abstract behavioural components (e.g. sub-goals) are partially based on probability density estimates for non-abstract components (activities). This cascade allows behaviours to be defined and recognised in terms of re-usable, abstract features and facilitates a succinct representation and behaviour explanations.
2. Key to our approach is the ability to use multiple particle filters to hypothesise different multi-agent scenarios. This required the development of a technique for combining posterior filtering densities from multiple filters to obtain a combined distribution. The process developed functions on normalised filter densities and could thus be applied in numerous other scenarios. The re-normalisation process also accounts for conflicting evidence and remains robust in challenging situations.

1.4 Thesis Roadmap

The next chapter will introduce pertinent related work. This will include an overview of the three main approaches to recognition: semantic models, state models, and feature models. In order to remain succinct the discussion will primarily focus on the work that is

most similar to our own, although a brief discussion of underlying probabilistic and video processing techniques will also be presented. The chapter will conclude with a discussion on multi-agent behaviour recognition, which is often considered a sub-field of behaviours recognition.

Having introduced the background material Chapter 3 will begin to introduce the framework by discussing the underlying concept: the bag-of-features. This will then be formalised by presenting the Bayesian Network representation.

Chapter 4 will combine the underlying probabilistic inference techniques from Chapter 2 and concepts from Chapter 3 to derive an efficient inference mechanism for the bag-of-features approach. This mechanism will be introduced in two stages; a basic algorithm that introduces the core algorithm components, and an advanced algorithm that optimises performance.

Up until this point in the thesis the underlying principles will have been presented in a ‘flat’ representation. That is, behaviours will have been considered as a single set of primitive features (activities). However, a taxonomy structure is often seen as a more natural representation for behaviour, and allows the decomposition of goals into smaller components (sub-goals, actions). Chapter 5 will go beyond the basic representation presented in the previous chapters to describe how such a decomposition can also be applied to bags-of-features.

Within this thesis the major benefit of applying a taxonomy structure is that it allows us to model multi-agent behaviour. How this is achieved is the focus of Chapter 6, which updates the original inference algorithms from Chapter 4. The approach utilises combinatorial optimisation to identify collaborating agents, and a Simulated Annealing algorithm will be introduced to show how this may be efficiently achieved.

Chapter 7 will draw on all of the previous chapters to describe the implementation details of our validation framework within the visual surveillance domain. The validation will make use of simulated data in addition to real video data. This chapter will discuss the details of how these two forms of data were obtained and processed, and will additionally discuss the mechanism by which behaviour predictions are made.

The evaluation will commence in Chapter 8 using the simulated data. This data source allows rigorous testing of different scenarios and conditions, and is followed in Chapter 9 with experiments using video data. In addition to the basic recognition performance of

the approach the experiments will also consider more complex conditions such as agents changing their behaviour, and will evaluate the impact that low-level processing errors have on higher-level inference.

Chapter 10 will interpret the results in a number of ways. First, the results will be discussed in relation to the bag-of-features framework, and will then be discussed in relation to real-world applications. We will consider how well the approach is able to meet requirements such as the ability to explain predictions and handle noisy environments, and will also compare results with other published research. Finally, this chapter also considers the scalability of the approach and identifies areas of limitation.

Chapter 11 will close the thesis by drawing together the contributions of this research and summarising what has been learnt. There are also a number of directions in which future work might proceed and these too will be discussed.

Chapter 2

Literature Survey

As highlighted in the introduction, automated behaviour recognition is a multifaceted problem that encompasses techniques from several different areas of research. This has led to a vast quantity of literature considering different aspects of the problem. In this chapter we identify how pertinent prior work has informed our research, and discuss the novelties of bag-of-features inference in relation to existing techniques.

To assist the discussions later in the chapter, Section 2.1 will introduce several generic Bayesian inference techniques. Numerous approaches have utilised and extended these techniques, so it is important to understand their merits at a generic level before introducing implementation specific extensions. Similarly, some of the approaches to be discussed utilise video processing techniques to track agent behaviour, so these too will be introduced in their generic form in Section 2.2.

Once these generic techniques have been presented the discussion will turn to a more specialised introduction to behaviour recognition approaches. This will commence with techniques using agent trajectory in Section 2.3. Because complex agent behaviour is hard to encapsulate using trajectories alone, many authors have considered more descriptive modelling approaches. These include semantic models (Section 2.4), state models (Section 2.5) and feature-based models (Section 2.6).

Because this research is focused on recognising complex behaviour a discussion of related work would not be complete without considering multi-agent behaviour. There are several elements to multi-agent behaviour recognition and these will be discussed in Section 2.7. Finally, Section 2.8 will draw this chapter together by making several conclusions and

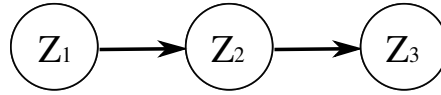


Figure 2.1: A simple Bayes Network

identifying areas requiring further research.

2.1 Probabilistic Inference Techniques

The need to reason about uncertainty has led to the development of probabilistic inference techniques for a wide range of applications, including behaviour recognition. Yet many of these techniques build upon fundamental principles applied in different ways. Several of the techniques introduced later in this chapter will rely on an understanding of these principles. The next two subsections will introduce Bayesian Networks and their temporal counterpart: Dynamic Bayesian Networks. This will be followed by the Bayesian Filter, which performs exact inference on generic temporal models. For some problems exact inference becomes intractable. Algorithms that perform approximate inference can prove useful in these situations, with one such algorithm being Sequential Importance Sampling (SIS). This sample based approach will be introduced in subsection 2.1.6, followed by a modified version of the algorithm: Sequential Importance Sampling with Resampling. Finally, Rao-Blackwellisation is a technique that improves approximate inference by utilising the structures found in some temporal models, and will be introduced at the end of the section.

2.1.1 Bayesian Networks

Bayesian modelling is frequently used to encapsulate processes in a probabilistic formation. One of the benefits of Bayesian models is that they can be represented graphically using a Bayesian Network (or Bayes Network for short), as shown in Figure 2.1. A Bayes Network is an acyclic graphical model where nodes denote variables and edges denote dependencies. In this example Z_2 is dependant on Z_1 , while Z_3 is *conditionally independent* of Z_1 given Z_2 (i.e. $P(Z_3|Z_1, Z_2) = P(Z_3|Z_2)$). Although variables are often considered binary, this is not a requirement of the generic Bayes Network. To define a network one must specify its **V**ertices, **E**dgcs, and a **C**onditional **P**robability **T**able, and is often represented by the tuple $\{V, E, CPT\}$.

Transition	I	J	Priors
I	0.9	0.1	0.2
J	0.5	0.5	0.8

Table 2.1: Transition matrix and prior probabilities for a simple Markov Model

2.1.2 Markov Models

A limitation of the generic Bayes Network is that it models the state of a system as a snapshot, that is, during a single instance in time. This makes them poorly suited for modelling processes that evolve (non stationary processes). However, Bayes Nets can be combined with a temporal model to form the Markov Model. In a Markov Model each node represents the state of a process at a particular instance in time. With respect to Figure 2.1, if the subscripts are considered as time indices then it can be said that Z_2 is the state of a process at time $t = 2$, and the probability that state Z_3 takes on a certain value z may be written as $P(Z_3 = z | Z_2 = z_2)$. Markov models are efficient because they make a Markov Assumption, that is, future states are independent of past states given the current state.

Markov Models represent the states of a system as a linear chain. The probability of transitioning from one state to another is specified via a transition matrix, and due to the Markov assumption, only needs to consider transitions from the state at time t to $t + 1$. Each model also requires a set of prior probabilities which specify the probability of each state at the first time step.

To model behaviour using a Markov Model each detectable action is first associated with a state, while the model parameters (transition matrix and prior probabilities) are normally learnt or defined by an expert. To model N activities requires N models. Recognition is performed by calculating the probability of each model having caused the observed action sequence. For example, consider the transition matrix and prior probabilities shown in Table 2.1. The probability of the observed action sequence $I \rightarrow I \rightarrow J$ can be calculated as:

$$P(Z_t = J | Z_{t-1} = I)P(Z_{t-1} = I | Z_{t-2} = I)P(Z_{t-2} = I) = 0.1 * 0.9 * 0.2 = 0.018 \quad (2.1)$$

By comparing this probability against that derived with another behaviour model one can determine which behaviour most likely explains the action sequence.

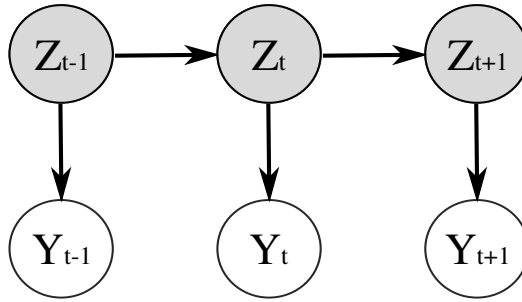


Figure 2.2: The Hidden Markov Model

2.1.3 Hidden Markov Models

The Hidden Markov Model (HMM) is a variation of the Markov Model in which the states of the process are not directly observable (i.e. they are hidden) [121]. Instead, the process is monitored through a sequence of observations from which the true process state must be estimated. This can be represented using a Hidden Markov Model, which uses hidden variables to represent the (unobservable) process state, and observable variables representing the observations.

This is shown graphically in Figure 2.2, where the shaded nodes represent the hidden (or commonly referred to as ‘latent’) process states and the unshaded nodes denote observations. As before, the model parameters consist of the transition matrix and prior probabilities, but now also include an emission model. The emission model specifies the probability of observing $Y_t = y_t$ given the process is in state $Z_t = z_t$. Furthermore, the Markov assumption states that each observation is conditionally independent given the state, and thus $P(y_t|y_{1:t}, Z_{1:t}) = P(y_t|Z_t)$.

2.1.4 Dynamic Bayesian Networks

Murphy showed that Hidden Markov Models are actually a special case of Dynamic Bayesian Network (DBN) [101]. DBNs model temporal processes, but unlike HMMs, they can have any number of hidden states. A network not only encodes the temporal structure of a process, but also encodes node dependencies within each time-step. As before, DBNs make the assumption that the process being modelled is Markov. It should be noted that the structure of a DBN remains static between time slices, and thus may be considered homogeneous. DBNs would be more appropriately named *Temporal Bayesian Networks* [101].

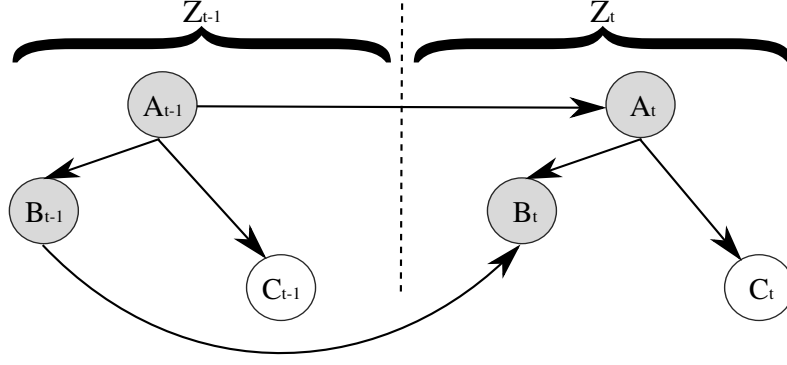


Figure 2.3: Example two-slice Bayes network defining $P(Z_t|Z_{t-1})$. Shaded nodes are latent, while unshaded nodes are directly observable.

To specify a DBN using the notation from [101], assume that Z_t represents both the hidden and output variables of a state-space model. A DBN is then defined to be a pair (B_0, B_{\rightarrow}) , where B_0 is a Bayes Net defining the prior $P(Z_1)$ and B_{\rightarrow} is a two-slice Bayes Net defining $P(Z_t|Z_{t-1})$ such that:

$$P(Z_t|Z_{t-1}) = \prod_{i=1}^N P(Z_t^i | Pa(Z_t^i)) \quad (2.2)$$

Where Z_t^i is the i 'th node of the Bayes Net at time t , $Pa(Z_t^i)$ are the parents of node Z_t^i and $P(Z_t^i | Pa(Z_t^i)) = P(Z_1^i)$ when $t = 1$. The resulting joint distribution of T time-slices is given by:

$$P(Z_{1:T}) = \prod_{t=1}^T \prod_{i=1}^N P(Z_t^i | Pa(Z_t^i)) \quad (2.3)$$

For example, Figure 2.3 shows a basic two-slice Bayes Net defining $P(Z_t|Pa(Z_t))$. One can see that nodes A_t and B_t are dependent upon the previous time step. The unrolled network (the joint distribution) can be written for two time slices as:

$$P(Z_{1:2}) = \prod_{t=1}^2 \prod_{i=1}^N P(Z_t^i | Pa(Z_t^i)) \quad (2.4)$$

$$= \prod_{i=1}^N P(Z_1^i) \times P(Z_2^i | Pa(Z_2^i)) \quad (2.5)$$

$$= P(A_1)P(B_1|A_1)P(C_1|A_1) \times P(A_2|A_1)P(B_2|A_2, B_1)P(C_2|A_2) \quad (2.6)$$

It can be observed from equations 2.3 and 2.6 that a DBN can be ‘unrolled’ over time to give a standard Bayes Network. This allows a large class of algorithms designed for standard Bayes Networks to be applied to DBNs.

To map the dynamic Bayesian network notation back to HMM notation, recall that in a HMM the hidden state (Z_t) is comprised of only one node. Because there is only one hidden node, in HMM terminology the conditional probability $P(Z_t|Z_{t-1})$ is often simply referred to as the transition probability $A(i, j)$, where i represents the state at $t - 1$ and j the state at time t .

2.1.5 Bayes Filter

The Bayes Filter can be used to perform recursive Bayesian estimation. Consider the general state-space model with hidden variables Z_t and observed variable y_t . Recall from the previous section that $P(Z_1)$ represents the prior (initial) distribution and $P(Z_t|Z_{t-1})$ represents the transition model. Note that for simplicity, the superscript indices representing variables have been removed. Recall also that a Markovian assumption is made and thus the sequence of T observations $y_{1:T} = \{y_1, y_2, \dots, y_T\}$ is assumed to be conditionally independent given the marginal distribution $P(y_t|Z_t)$.

The joint distribution of all states and all observations up until time T can therefore be written as:

$$P(Z_{0:T}, y_{1:T}) = \prod_{t=1}^T P(y_t|Z_t) P(Z_t|Z_{t-1}) \quad (2.7)$$

The marginal $P(Z_t|y_{1:T})$ is known as the filtering distribution. To see how this may be calculated, suppose that $P(Z_{t-1}|y_{1:t-1})$ is known. There are then two steps to recursive Bayesian estimation. The first step, termed *Prediction*, uses the transition model to calculate the prior probability of the state at time t . This is shown in Equation 2.8.

$$P(Z_t|y_{1:t-1}) = \int P(Z_t|Z_{t-1}) P(Z_{t-1}|y_{1:t-1}) dz_{t-1} \quad (2.8)$$

At time-step t an observation y_t becomes available which can be used to update the prediction via the *Update* step. The marginal $P(Z_t|y_{1:T})$ is calculated using the observation model $P(y_t|Z_t)$:

$$P(Z_t|y_{1:t}) = \frac{P(y_t|Z_t) P(Z_t|y_{1:t-1})}{P(y_t|y_{1:t-1})} \quad (2.9)$$

Where the denominator can be calculated as:

$$P(y_t|y_{1:t-1}) = \int P(Z_{t-1}|y_{1:t-1}) P(Z_t|Z_{t-1}) P(y_t|Z_t) dz_t \quad (2.10)$$

The posterior $P(Z_t|y_{1:T})$ can thus be calculated recursively using the two steps: Prediction (Equation 2.8) and Update (Equation 2.9). Return now to the original assumption that $P(Z_{t-1}|y_{1:t-1})$ is known, for which it may be unclear what happens when $t = 1$. Under this condition no observations have been seen and thus $P(Z_{t-1}|y_{1:t-1})$ is simply the prior $P(Z_1)$. At all other time-steps $P(Z_{t-1}|y_{1:t-1})$ is the estimate from the previous time-step.

Recursive Bayesian estimation is particularly useful for processes that evolve with time. The arrival of a new observation allows the posterior probability to be updated without maintaining a history of all previous observations, making it efficient for processes with long duration.

This two-step Bayes Filter performs exact Bayesian inference, however, a limitation of this solution is that Equation 2.10 cannot typically be calculated analytically. While it can be calculated for models with discrete state spaces, and for restricted models such as the Kalman Filter, for other models approximation techniques are required [39].

2.1.6 Sequential Importance Sampling (SIS)

The Sequential Importance Sampling (SIS) algorithm implements recursive Bayesian estimation using Monte Carlo sampling. It is variously known as particle filtering [40], the bootstrap filter [54], the condensation algorithm [17] and Sequential Monte Carlo [39]. The SIS algorithm allows approximate inference to be performed when exact inference is inappropriate.

Fundamentally, the approach uses a set of weighted random samples (particles) to approximate the posterior density $P(Z_{1:T}|y_{1:T})$, from which the filtering density $P(Z_T|y_{1:T})$ can be calculated. To see how this is achieved, first re-write $P(Z_{1:T}|y_{1:T})$:

$$P(Z_{1:T}|y_{1:T}) = \frac{P(Z_{1:T}, y_{1:T})}{P(y_{1:T})} = \frac{P(y_{1:T}|Z_{1:T})P(Z_{1:T})}{\int P(y_{1:T}, Z_{1:T})dz_{1:T}} \quad (2.11)$$

Because $\int P(y_{1:T}, Z_{1:T})dz_{1:T}$ is a constant normalisation factor it can be isolated and $P(Z_{1:T}|y_{1:T})$ rewritten, denoting the constant as Ψ . Furthermore, because the process is assumed to be Markov $P(y_T|Z_{1:T}) = P(y_T|Z_T)$. This gives:

$$P(Z_{1:T}|y_{1:T}) = \frac{1}{\Psi} \left(\prod_{t=1}^T P(y_t|Z_t) \right) P(Z_{1:T}) \quad (2.12)$$

Denote a random sample of size N as $\{Z_{1:T}^i, \omega_T^i\}_{i=1}^N$. Let $Z_{1:T}^i$ be the i 'th particle sampled from $P(Z_{1:T})$, and ω_T^i be its associated weight such that $\sum_{i=1}^N \omega_T^i = 1$ and $\omega_T^i = \prod_{t=1}^T P(y_t|Z_t)$. Substituting $Z_{1:T}^i$ and ω_T^i into Equation 2.12 the normalising constant can be dropped giving:

$$P(Z_{1:T}|y_{1:T}) \approx \sum_{i=1}^N \omega_T^i \delta(Z_{1:T}, Z_{1:T}^i) \quad (2.13)$$

In calculating ω_T^i there is actually a recurrence relation that can be taken into consideration. Consider the weight ω_{T-1}^i :

$$\omega_{T-1}^i = \prod_{t=1}^{T-1} P(y_t|Z_t = Z_t^i) \quad (2.14)$$

When Z_t^i is sampled from $P(Z_t|Z_{t-1} = Z_{t-1}^i)$ the weight can actually be factorised to give the weight update equation:

$$\omega_t^i = \omega_{t-1}^i P(y_t|Z_t = Z_t^i) \quad (2.15)$$

Having defined the prediction and weight update steps, the generic SIS algorithm can be summarised by Algorithm 2.1. A particle set $\{Z_1^i, \omega_1^i\}_{i=1}^N$ is initialised according to the prior $P(Z_1)$ and ω_1 . The particles then predict the new state at time $t + 1$ by drawing from the transition kernel $P(Z_t^i|Z_{t-1}^i)$ and are weighted according to Equation 2.15.

Algorithm 2.1 The standard generic SIS algorithm

- 1: Init: Generate $[\{Z_1^i, \omega_1^i\}_{i=1}^N] \sim P(Z_1)$ and ω_1
 - 2: **for** $t = 1$ to T **do**
 - 3: **for** $i = 1$ to N **do**
 - 4: Prediction: Draw $Z_t^i \sim P(Z_t^i|Z_{t-1}^i)$
 - 5: Update: Assign weight to Z_t^i according to Equation 2.15
 - 6: **end for**
 - 7: **end for**
-

Arulampalam *et al.* showed that the SIS algorithm can be modified to utilise two kernels in [9]. This can be useful when a second kernel is required to incorporate different state knowledge. To show how this is achieved let us re-write Equation 2.12 as follows:

$$\begin{aligned} P(Z_{1:T}|y_{1:T}) &= \frac{1}{\Psi} \times \left(\prod_{t=1}^T P(y_t|Z_t) \right) \times P(Z_{1:T}) \\ &= \frac{1}{\Psi} \times \left(\prod_{t=1}^T P(y_t|Z_t) \right) \times P(Z_t|Z_{t-1})P(Z_{t-1}|Z_{t-2}) \times \\ &\quad P(Z_{t-2}|Z_{t-3}) \dots P(Z_2|Z_1)P(Z_1) \end{aligned}$$

Because $P(Z_{1:T})$ is sampled using the transition kernel, if it is assumed that $P(Z_1|Z_{-1}) = P(Z_1)$ (as before) then Equation 2.12 can be further rewritten as:

$$P(Z_{1:T}|y_{1:T}) = \frac{1}{\Psi} \times \left(\prod_{t=1}^T P(y_t|Z_t) \right) \times P(Z_{1:T}) \quad (2.16)$$

$$= \frac{1}{\Psi} \times \left(\prod_{t=1}^T P(y_t|Z_t) P(Z_t|Z_{t-1}) \right) \quad (2.17)$$

Let us denote $G(Z_t, y_t) = P(y_t|Z_t)$ and $Q(Z_t, Z_{t-1}) = P(Z_t|Z_{t-1})$, where Q is the transition kernel. Suppose that there is a further kernel $K(Z_t, Z_{t-1}, y_t) = P(Z_t|Z_{t-1}, y_t)$ and function $H(Z_t, Z_{t-1}, y_t)$ that satisfy:

$$G(Z_t, y_t) Q(Z_t, Z_{t-1}) = H(Z_t, Z_{t-1}, y_t) K(Z_t, Z_{t-1}, y_t) \quad (2.18)$$

Then it follows that Equation 2.17 can be re-written:

$$P(Z_{1:T}|y_{1:T}) = \frac{1}{\Psi} \times \left(\prod_{t=1}^T P(y_t|Z_t) P(Z_t|Z_{t-1}) \right) \quad (2.19)$$

$$= \frac{1}{\Psi} \times \left(\prod_{t=1}^T G(Z_t, y_t) Q(Z_t, Z_{t-1}) \right) \quad (2.20)$$

$$= \frac{1}{\Psi} \times \left(\prod_{t=1}^T H(Z_t, Z_{t-1}, y_t) K(Z_t, Z_{t-1}, y_t) \right) \quad (2.21)$$

$$= \frac{1}{\Psi} \times \prod_{t=1}^T H(Z_t, Z_{t-1}, y_t) \prod_{t=1}^T K(Z_t, Z_{t-1}, y_t) \quad (2.22)$$

Let $\xi_{1:T}^i$ be the i 'th particle sampled from $\prod_{t=1}^T K(Z_t, Z_{t-1}, y_t)$.

$$P(Z_{1:T}|y_{1:T}) = \sum_{i=1}^N \omega_t^i \delta(Z_{1:T}, \xi_{1:T}^i) \quad (2.23)$$

When we are only interested in the filtering distribution $P(Z_T|y_{1:T})$ the history of all previous states can be dropped via the Markov assumption to give:

$$P(Z_T|y_{1:T}) \approx \sum_{i=1}^N \omega_T^i \delta(Z_T, \xi_T^i) \quad (2.24)$$

where

$$\omega_T^i = \prod_{t=1}^T H(\xi_t^i, \xi_{t-1}^i, y_t) \quad (2.25)$$

$$= \omega_{t-1}^i H(\xi_t^i, \xi_{t-1}^i, y_t) \quad (2.26)$$

Equation 2.18 can be rearranged in terms of $H(Z_t, Z_{t-1}, y_t)$ allowing its substitution to:

$$\omega_t^i = \omega_{t-1}^i H(\xi_t^i, \xi_{t-1}^i, y_t) \quad (2.27)$$

$$= \omega_{t-1}^i \frac{G(\xi_t^i, y_t) Q(\xi_t^i, \xi_{t-1}^i)}{K(\xi_t^i, \xi_{t-1}^i, y_t)} \quad (2.28)$$

$$= \omega_{t-1}^i \frac{P(y_t|Z_t)P(Z_t|Z_{t-1})}{P(Z_t|Z_{t-1}, y_t)} \quad (2.29)$$

This updated SIS procedure can be summarised via Algorithm 2.2. As before, the particle set is initialised according to the prior distribution $P(Z_1)$, with weights drawn from the prior weight distribution ω_1 . The algorithm then iterates for T time steps. In the prediction step the system kernel ($P(Z_t|Z_{t-1})$) and the second kernel ($P(Z_t|Z_{t-1}, y_t)$) are both used to generate the new state particles $\{Z_t^1, Z_t^2 \dots Z_t^N\}$. Each particle in the new sample is weighted in accordance with its likelihood given the new observation y_t , and thus the weighting forms the update step, which now includes the second kernel $P(Z_t|Z_{t-1}, y_t)$. As before, particles that are more likely will attract a higher weight while less likely particles attract lower weights. Note that as the number of particles approaches ∞ the approximation of $P(Z_{1:T}|y_{1:T})$ will approach the true probability.

Algorithm 2.2 The Two-Kernel SIS algorithm

- 1: Init: Generate $[\{Z_1^i, \omega_1^i\}_{i=1}^N] \sim P(Z_1)$ and ω_1
 - 2: **for** $t = 1$ to T **do**
 - 3: **for** $i = 1$ to N **do**
 - 4: Prediction: Draw $Z_t^i \sim Q(Z_t^i|Z_{t-1}^i, y_t)$
 - 5: Update: Assign weight to Z_t^i according to Equation 2.29
 - 6: **end for**
 - 7: **end for**
-

To give a simple example of the SIS filter in action consider a hypothetical target-tracking filter. Suppose the target is a moving object in Cartesian space with a constant velocity and trajectory. Now suppose that the target can only move along one of two trajectories and there are two particles in a particle filter (one for each potential trajectory). Each particle is initialised with the approximate location of the target at the first time-step. The associated trajectories can be observed in Figure 2.4, in which one can see that Particle 1 provides a better estimate of the trajectory than particle 2.

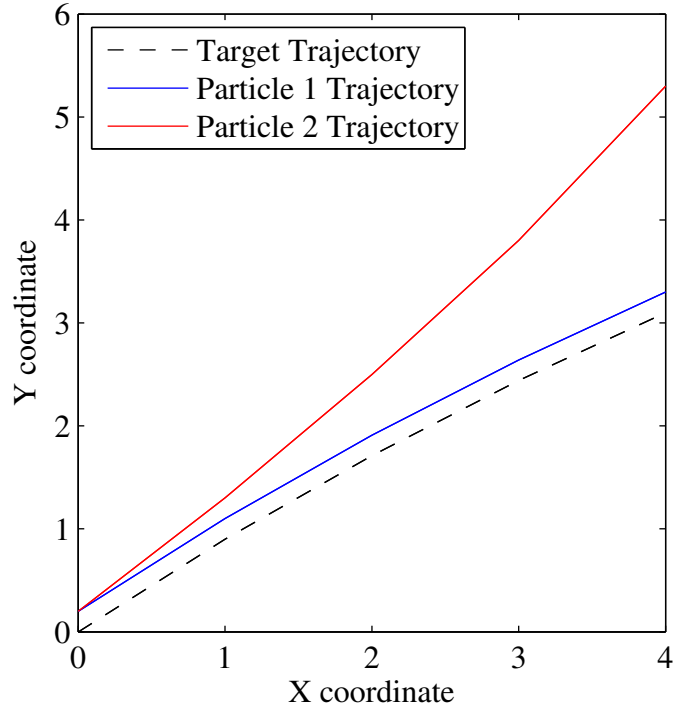


Figure 2.4: A simple target trajectory, and estimated trajectories by two different particles

Assume that the target's position is observed at one second intervals with a Normal error distribution (mean error of $0.1m$, standard deviation of 1). If each particle estimates the target's new position at the same frequency, the Euclidean distance could be used to measure the error of each prediction and convert this to an observation probability using the error distribution. The normalised particle weights for each observation can be seen in Figure 2.5, where, as one would expect, Particle 1 becomes most probable.

Now assume that the filter actually contains 100 particles equally distributed between the two trajectories, and each is initialised with a slightly different origin. The particles with origin's closest to the true origin would obtain more weight than those with poorer initialisations, and thus the filter could be used to derive a more accurate estimation of the target trajectory.

2.1.7 SIS with Resampling

Sample degeneracy is cited as a common problem with the SIS algorithm [39]. Degeneracy occurs after several iterations of the algorithm, where all but a small number of particles will have negligible weight. This can be observed in Figure 2.5, where one can see that after only five observations particle 2 has negligible weight. A large proportion of

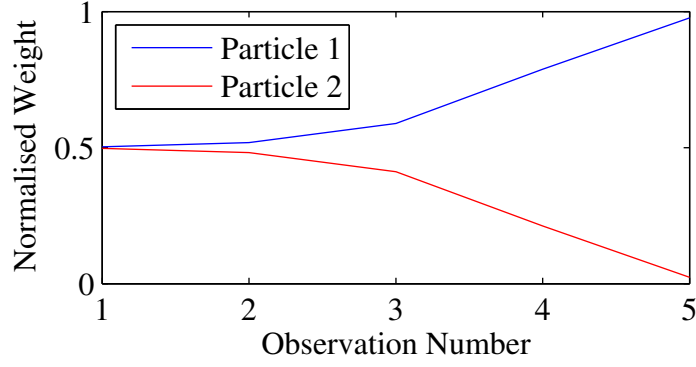


Figure 2.5: Normalised particle weights for five observations

Algorithm 2.3 Systematic Sampling

- 1: Prototype: $[\{Z_t^{i*}, \omega_t^{i*}\}_{i=1}^N] = \text{Resample}([\{Z_t^i, \omega_t^i\}_{i=1}^N])$
 - 2: $[\{Z_t^{i*}, \omega_t^{i*}\}_{i=1}^N] = \emptyset$
 - 3: $C_1 = 0$
 - 4: **for** $i = 2$ to N **do**
 - 5: Construct the cumulative distribution function (CDF): $C_i = C_{i-1} + \omega_t^i$
 - 6: **end for**
 - 7: Reset $i = 1$
 - 8: Draw random starting point: $u_1 \sim \mathbb{R}[0, N^{-1}]$
 - 9: **for** $j = 1$ to N **do**
 - 10: Move along the CDF: $u_j = u_1 + N^{-1}(j - 1)$
 - 11: **while** $u_j > c_i$ **do**
 - 12: $i = i + 1$
 - 13: **end while**
 - 14: Add new sample point: $Z_t^{j*} = Z_t^i$
 - 15: Add new weight: $\omega_t^{j*} = N^{-1}$
 - 16: **end for**
-

inference time is therefore wasted updating particles that have little effect on the posterior density. It has been shown that the variance of importance weights can only increase with time, and thus the problem of degeneracy cannot be directly avoided.

This phenomena is overcome by adding a re-sampling step into the algorithm, forming the SIR algorithm. The re-sampling step involves generating a new set of particles $\{Z_t^{i*}\}_{i=1}^N$ by sampling N times with replacement from the approximate representation of $P(Z_T|y_{1:T})$ (given by Equation 2.24) so that $P(Z_t^{i*} = Z_t^i) = \omega_t^i$. The resulting particle set is an independent and identically distributed (i.i.d.) sample from Equation 2.24, and thus the weights are reset to $\{\omega_t^i = 1/N\}_{i=1}^N$. The effect of re-sampling is that particles with very low weights in $\{Z_t^i\}_{i=1}^N$ are less frequent in $\{Z_t^{i*}\}_{i=1}^N$, while particles with higher weights are more frequent.

Referring back to the hypothetical filter distributions in Figures 2.4 and 2.5, assume that once again the filter actually consists of 100 particles equally distributed between the

Algorithm 2.4 The Two-Kernel SIR algorithm

```
1: Generate  $[\{Z_1^i, \omega_1^i\}_{i=1}^N] \sim P(Z_1)$  and  $\omega_1$ 
2: for  $t = 1 : T$  do
3:   for  $i = 1 : N$  do
4:     Re-Sample:  $\tau_t^i \sim \{Z_{t-1}^i, \omega_{t-1}^i\}$  using Algorithm 2.3
5:     Propagation: Draw  $Z_t^i \sim K(Z_t^i, \tau_{t-1}^i, y_t)$ 
6:     Weighting: Assign weight to  $Z_t^i$  according to Equation 2.24
7:   end for
8: end for
```

two trajectories. The SIR filter now re-samples the particles according to the weight distribution, and thus Figure 2.5 is representative of the number of ‘Trajectory 1’ particles in the filter. Correspondingly, there are fewer particles representing ‘Trajectory 2’ and inference time is not wasted updating these particles, but focused instead on finding the best ‘Trajectory 1’ estimation.

There are a number of $O(N)$ approaches for performing the re-sampling step, including algorithms based on first order statistics [26], residual sampling [87] and systematic re-sampling [9, 71]. The approach taken in this work is systematic re-sampling [71] due to the ease of its implementation, and is summarised in Algorithm 2.3. Its integration into the SIR algorithm is described in Algorithm 2.4.

2.1.8 Rao-Blackwellisation

Sequential importance sampling/re-sampling gains its efficiency through sampling Z_t , but bears the limitation that sampling a high-dimensional state space is inefficient. In some cases a model encapsulates a tractable structure that can be analytically marginalised out conditioned upon other nodes. This marginalisation considerably reduces the state space that must be sampled and therefore increases efficiency [38].

This process of marginalising out some variables is known as Rao-Blackwellisation, and its integration with the SIR algorithm gives rise to the Rao-Blackwellised Particle Filter (RBPF). The underlying concept is to partition the collection of latent variables Z_t into those that will be sampled, and those that will be marginalised. Denote the sampled component as r , and the marginalised component $z : Z_t = \{r_t, z_t\}$. The posterior in RBPF filtering can thus be expressed by the following factorisation:

$$p(Z_t | y_{1:t-1}) = p(z_t | r_t, y_{1:t-1}) p(r_t | y_{1:t-1}) \quad (2.30)$$

Algorithm 2.5 The Rao-Blackwellised Particle Filter algorithm

- 1: Generate $[\{z_0^i, r_0^i, \omega_0^i\}_{i=1}^N] \sim P(Z_0)$ and ω_0
 - 2: **for** $t = 1 : T$ **do**
 - 3: **for** $i = 1 : N$ **do**
 - 4: Resampling: Generate $r_t^i \sim \{r_{t-1}^i, \omega_{t-1}^i\}$ using Algorithm 2.3
 - 5: Prediction: Draw $Z_t^i = \{r, t\}_t^i \sim q(z_t^i | r_{t-1}^i, y_t) q(r_t^i | r_{t-1}^i, y_t)$
 - 6: Update: Assign weight to Z_t^i according to Equation 2.37
 - 7: **end for**
 - 8: **end for**
-

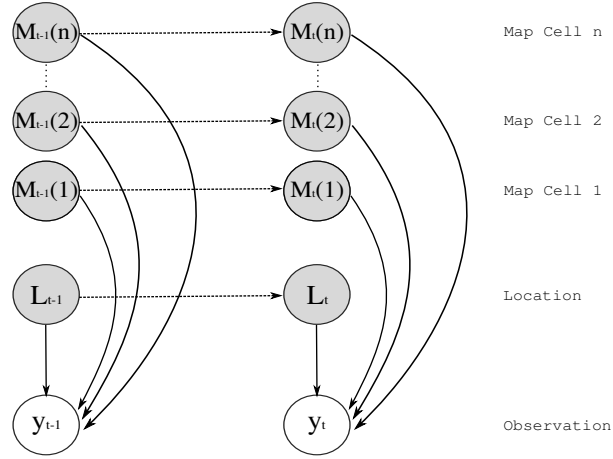


Figure 2.6: Example Dynamic Bayes Net for the robot localisation and mapping problem from [38]

A Simple Example

Doucet *et al.* illustrate Rao-Blackwellisation with a Robot Localisation and Mapping example in [38]. Consider a robot moving in a discrete grid of l cells, where each cell can be shaded or unshaded. The robot can detect the colour of the current cell, but this detection is subject to a noise model $f(\cdot)$ which randomly flips the detection. As the robot moves around the grid it tracks its location (L_t) and updates its map ($M_t(L_t)$). However, the problem is that the robot does not always move as expected (e.g. wheel slip), and easily becomes lost meaning it does not know which location of the map to update.

The Dynamic Bayes Net for this problem is shown in Figure 2.6, and can be modelled in a particle filter by encapsulating the agent state (shaded nodes) in each particle. A standard particle filter samples the entire state, however, sampling can be made more efficient by partitioning the state into $r_t = L_t, z_t = \{M_t(i)\}_{i=1}^l$. A Rao-Blackwellised particle filter only samples r_t (in this case, the agent's location) and marginalises the remaining variables (the map variables) by conditioning on r_t .

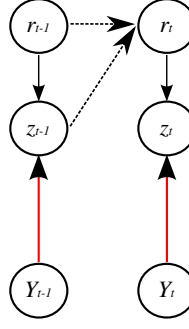


Figure 2.7: Incorporating the latest observation into the RB-Posterior reverses the edge between y_t and z_t

Incorporating Evidence

A limitation with the standard Rao-Blackwellised factorisation is that the latest evidence (y_t) is not considered when calculating the distribution of z_t . This can result in a large number of particles predicting values of z_t that will obtain low weights during the weighting step. These particles are effectively wasted samples and lead to sub-optimal performance. It is therefore logical to incorporate information about the latest observation when calculating the distribution of z_t . This is done by applying the Two-Kernel SIR approach to the RBPF algorithm. To proceed, the Rao-Blackwellised Posterior $P(z_t | r_t, y_{1:t-1})$ is converted into $P(z_t | r_t, y_{1:t})$. This is done by sampling from kernel $K(\{z, r\}_t, \{z, r\}_{t-1}, y_t)$, which is the Rao-Blackwellised form of $K(Z_t, Z_{t-1}, y_t)$ from section 2.1.6. This has the effect of reversing the edge between nodes y_t and z_t in the DBN and is visualised in Figure 2.7.

Using this new RB-Posterior, the distribution of z_t will now incorporate the latest evidence. These concepts can be merged into the SIR algorithm to form the Rao-Blackwellised Particle Filter (RBPF), which consists of N random samples of the form $\{\{r, z\}_{1:t}^i, \omega_t^i\}_{i=1}^N$ that characterise the posterior density $P(Z_t | y_{1:t})$. Each sample point $\{r, z\}_t^i$ has an associated weight ω_t^i such that $\sum_{i=1}^N \omega_t^i = 1$ as with the original SIS/SIR algorithms. The posterior density at time t is approximated as:

$$P(Z_t | y_{1:t}) \approx \sum_{i=1}^N \omega_t^i \delta(\{r, z\}_t, \{r, z\}_t^i) \quad (2.31)$$

As shown in Section 2.1.6, the Two-Kernel weight update equation is given by:

$$\omega_t^i \propto \omega_{t-1}^i \frac{Q(\{r, z\}_t, \{r, z\}_{t-1}, y_t)}{K(\{r, z\}_t, \{r, z\}_{t-1}, y_t)} \quad (2.32)$$

$$\propto \omega_{t-1}^i \frac{P(y_t | \{z, r\}_t) P(\{z, r\}_t | \{z, r\}_{t-1})}{P(\{z, r\}_t | \{z, r\}_{t-1}, y_t)} \quad (2.33)$$

Using this new weighting function and the three steps of the SIR algorithm, the RBPF algorithm can be described as below.

Resampling

A Resampling step such as Algorithm 2.3 multiplies samples with high weights and eliminates those with low weights. The resampled set $\{Z_t^{i*}, \omega_t^{i*}\}_{i=1}^N$ can be defined in terms of the RBPF by:

$$\{Z_t^{i*}, \omega_t^{i*}\}_{i=1}^N = \{\{r, z\}_t^{i*}, \omega_t^{i*}\}_{i=1}^N \quad (2.34)$$

Prediction

Where as the standard SIR algorithm estimated the probability density function $P(Z_T | y_{1:T})$, this step is now partitioned into the two components r_t and z_t . The first component is sampled from the distribution $P(r_t | y_{1:t-1})$ as per equation 2.35.

$$P(r_t | y_{1:t-1}) = P(r_t | Z_{t-1}) P(Z_{t-1} | y_{1:t-1}) \quad (2.35)$$

Once r_t has been predicted the exact conditional $P(z_t | r_t, y_{1:t})$ can be calculated and z_t predicted.

Update

The final step in the algorithm weights the particles so that $\sum_{i=1}^N \omega_t^i = 1$, where the weights should be proportional to the distribution in Equation 2.37.

$$\omega_t^i \propto \omega_{t-1}^i \frac{P(y_t|Z_t^i) P(Z_t^i|Z_{t-1}^i)}{P(Z_t^i|Z_{t-1}^i, y_t)} \quad (2.36)$$

$$\propto \omega_{t-1}^i \frac{P(y_t|\{r, z\}_t^i) P(\{r, z\}_t^i|\{r, z\}_{t-1}^i)}{P(\{r, z\}_t^i|\{r, z\}_{t-1}^i, y_t)} \quad (2.37)$$

As before, $P(Z_t|Z_{t-1}, y_t) = K(Z_t, Z_{t-1}, y_t)$, and correspondingly, $K(Z_t, Z_{t-1}, y_t) = P(\{r, z\}_t^i|\{r, z\}_{t-1}^i, y_t)$. Recall from Section 2.1.6 that $K(Z_t, Z_{t-1}, y_t)$ is some importance distribution that is assumed to be known, and thus no further changes are required to apply Rao-Blackwellisation to the algorithm.

2.1.9 Summary

This section has introduced some generic techniques for performing probabilistic inference. It began with Bayes Networks, which are graphical models that use edges to denote dependence between different variables in the model. Many Bayes Networks consist of latent (hidden) variables and observed variables. Given a set of known variable values Bayesian inference allows us to calculate the conditional probabilities of another set of variables, and is often used to calculate the probability of the latent variables given observations.

Where Bayes Networks become limited is with respect to temporal processes. Because a Bayes Network represents the state of a system at a single point in time it must be integrated with a temporal model to form a Dynamic Bayesian Network (DBN). Such a network describes the dependencies between nodes at different time-steps and can be ‘unrolled’ to give a standard Bayes Network. DBN inference can be performed using the Bayes Filter, which implements recursive Bayesian estimation and ‘updates’ the probability whenever a new observation is made. However, the Bayes Filter performs exact Bayesian inference and can become intractable for some problems. For this reason the Sequential Importance Sampling (SIS) and Sequential Importance Sampling with Resampling (SIR) algorithms have been suggested as two alternatives to the Bayes Filter. These algorithms perform approximate inference via sampling and can be applied in situations where the Bayes Filter becomes intractable.

A limitation of the SIS/SIR algorithms is that sampling is inefficient in high-dimensions. However, performance can be improved by integrating the concept of Rao-Blackwellisation,

which marginalises out variables by conditioning on a set of known variables. The Rao-Blackwellised particle filter (RBPF) provides such an integration by sampling some variables using approximate inference, and analytically marginalising others conditioned on the sampled variables. This has the effect of reducing the number of variables that must be sampled and improves inference.

2.2 Video Processing Techniques

In the context of behaviour recognition video processing is normally concerned with the identification and tracking of moving foreground objects. Broadly speaking, the processes involved can be categorised into three distinct stages:

- Foreground detection, in which the static background is segmented from the moving foreground components.
- Object classification, in which foreground components are classified into different types of object (e.g. person, luggage, clutter).
- Object tracking, in which objects are tracked between video frames to produce an object motion trajectory.

2.2.1 Foreground Detection

To detect foreground objects there are three popular techniques: background subtraction, temporal differencing, and optical flow analysis [72].

Adaptive Background Subtraction

Background subtraction performs pixel-by-pixel subtraction of the current image frame from a reference frame to identify foreground pixels (having a non-zero value). The reference frame represents the static background and is normally learnt by averaging pixel values over a number of frames. Because the background is rarely fixed, the reference frame must be adapted at run-time to accommodate illumination and motion changes (e.g.

shadows, trees). This can be achieved in a number of ways. Simple approaches use the median of the last n frames to update the background model [32], but ignore foreground pixels during the update so as not to pollute the background image. However, one of the key limitations of this approach is that a pixel cannot model multiple background objects, such as moving leaves in-front of a building [116]. Stauffer and Grimson suggested using a mixture of Gaussians to model multiple background objects [132], while Elgammal *et al.* propose Kernel Density Estimation [41] to solve the same problem. Performance comparisons of these and other approaches can be found in [116, 7].

Fusier *et al.* have applied background subtraction for video scene understanding in [46]. They adopt a colour mean and variance approach using a Gaussian distribution to model the colour of each background pixel. To improve performance they also add components to detect shadows and highlights by separating the brightness and chromaticity components of the background image [60].

Temporal Differencing

In some respects Temporal Differencing is similar to Background Subtraction in that it performs a pixel-by-pixel comparison between multiple frames. A threshold can then be defined to identify substantial image changes. However, unlike Background Subtraction this comparison is performed between two or three consecutive frames, rather than using a background (reference) frame.

Because consecutive frames are used, Temporal Differencing is very robust in dynamic scenes (e.g. lighting changes), however, it is frequently observed to do a poor job at extracting all relevant pixels in a moving object [64]. This leaves ‘holes’ within foreground images, although a number of morphological operations can be performed to assist in such matters. Furthermore, Lipton *et al.* comment that the approach fails when objects become occluded or cease their movement [86]. They use Temporal Differencing for tracking humans and vehicles from video.

Optical Flow Analysis

Optical flow analysis uses image intensity to identify moving image regions. Suppose that $I(x, y, t)$ is a spatio-temporal intensity function. During the interval d_t it can be assumed that the neighbourhood of (x, y) is translated a small distance (dx, dy) and thus $I(x, y, t) \approx$

$I(x + dx, y + dy, t + dt)$. However, the approach relies on a number of conditions [14]:

1. Objects have constant illumination
2. Objects have lambertian surface reflectance (isotropic luminance)
3. Pure translation parallel to the image plane

Because these conditions are rarely true in real-world scenarios they are relaxed such that it is assumed that they hold locally. The validity of this assumption thus effects the accuracy with which optical flow analysis estimates image motion. Furthermore, a number of issues further effect accurate motion estimation including transparency and occlusion, for which the scene must be segmented into regions corresponding to the independently moving objects.

Although there are a large number of optical flow algorithms (see [14] for a summary of several variations), their computational complexity and sensitivity to noise mean that they are rarely employed for real-time applications [72, 64].

2.2.2 Object Classification

Having identified the moving foreground pixels object classification is often performed on the connected pixels (commonly referred to as ‘blobs’). Object classification allows irrelevant blobs to be discarded, for instance those caused by dynamic scene elements such as moving trees or lighting changes. Furthermore, object identification also allows relevant motion models to be engaged during object tracking.

The most popular techniques for object classification from surveillance video can be broadly divided into two categories:

Shape based

Geometric properties such as size and shape often provide important cues about an object’s type and are frequently used to distinguish between humans, vehicles, luggage and clutter. For instance, lighting changes often create groups of small, sporadic blobs, which,

when combined with other cues (e.g. motion) allow for their elimination from the tracking process. Combining geometric information with a priori statistics is often one of the best ways to identify object class [25]. For instance, by combining ellipsoid detection the average height/width of humans has been shown to be very effective for detecting people [16]. Similar approaches have also been applied to detect common luggage items [91].

Motion based:

Object motion can also be used to identify object type, with gait detection having been proposed as a method for identifying particular people [126, 81]. Motion is also an important attribute in identifying static objects, but it is more frequently used to identify types of object motion rather than the object itself. For instance, distinguishing between people walking and running, or performing different gestures [47].

2.2.3 Object Tracking

Having identified the objects of interest tracking is often employed to estimate the trajectory of each object over a set of frames. Although target tracking is a problem encountered in many domains it still remains challenging, with noise, occlusions, multiple targets and abrupt changes in motion all causing particular difficulties.

The wide application potential of tracking research means that related work can be associated with different types of data (e.g. radar, sonar, video). To focus the discussion this section will primarily review video based tracking, but it is important to acknowledge that these techniques can often be applied to other types of data. Although Hu *et al.* identify four general categories for visual tracking approaches (region-based, contour-based, feature-based and model-based [64]), in many cases categorises can be combined to harness more robust approaches.

Region Based

Region-based algorithms are often integrated with background subtraction techniques, although this is not necessarily a pre-requisite if the approximate location of the object is already known. Using a bounding box to encapsulate each object, variations of the image

region can be tracked. For instance, the Adaptive Mean-Shift algorithm [137] uses the colour histogram of an object to match regions between frames. An object histogram is constructed by counting the number of occurrences of each pixel colour, and the intersection between one object histogram and an image histogram can be combined with a threshold to determine when a match is found. To account for changes in scale the histogram can be normalised by the image size, and lighting changes can be accommodated by slowly adjusting the object histogram according to a learning parameter [138].

Although earlier region-based approaches were unable to accommodate object occlusion, more recent work has progressed in this area. McKenna *et al.* use region histograms for tracking groups of people [96]. At each frame background subtraction is performed and a new region tracker is initialised for each novel region. People can be tracked using individual regions or collections of regions in close proximity which helps to overcome segmentation errors. To accommodate occlusion bounding boxes that overlap are said to have formed a group. When this happens updating the individual colour models (histograms) is suspended. The most likely group member associated with each pixel can be determined from the collection of group histograms, and group bifurcation can also be addressed when groups split by determining the most likely histogram(s) for each component.

One of the key limitations of region based approaches is that they cannot be used to recover the position or orientation (pose) of the object. However, in many cases the position of the object can be approximated in sparse scene by using the centre of the bounding box's lower element.

Active Contour Based

In active contour-based tracking the object boundary is used instead of object region. Contours are a more efficient representation than regions and also encapsulate the object's shape, making them a more detailed descriptor. In active contour based tracking curves are evolved under the influence of external potentials using active contour models such as Snakes [67], Balloons [29] and Geodesic active contours [114]. However, because these models rely upon accurate initialisation of the contour, automatically initiated tracking is challenging. Indeed, [30] suggests that this is the most difficult problem to solve.

A further limitation with contour based approaches is that they can only track objects through partial occlusion [64]. For instance, Yilmaz *et al.* combine active contours with

feature tracking (e.g. colour, texture) to predict the contours hidden by occlusions [146]. However, only partial occlusions are considered, and there is little discussion of full occlusion in any of the literature.

Feature Based

In feature-based tracking elements of the object are extracted and clustered to identify higher-level features. These features are matched between frames and can consist of global, local and dependence-graph based features. The advantage of this approach is that even during partial occlusion, it is likely that some of the features will remain visible. Furthermore, many features are robust to changes in lighting.

Global features relate to the entire object and include centroid, perimeter and area. Polana and Nelson demonstrate using the centroid of objects in [122] and can show that tracking can be performed during full occlusion by using the objects motion information (velocity) from the previous K frames.

Local features often take the form of lines, curves or corners. Coifman *et al.* use corner features to track vehicles in [30], and more recent work by Yen-Lin *et al.* also tracks vehicles, this time using individual headlights as features [28].

Dependency-graph features consist of geometric relations and distances, but are relatively rare in the literature. Fan *et al.* employ surface descriptors in [43] using features based on surface discontinuity (jump boundaries) and surface orientation discontinuities (creases). However, one of the drawbacks of matching dependency-graph features is that they require time consuming search and matching algorithms which inhibits real-time performance [64].

Model Based

Model-based tracking is the final approach to be discussed. Although model-based tracking can be applied to any number of ‘objects’, this discussion will be constrained to human-model tracking with the understanding that the same ideas can also be applied to other types of object. There are several elements to human model-based tracking: the body model, the motion model, and search and prediction strategies.

The body model can be of varying complexity, where more complex modelling generally delivers better tracking at the expense of longer computation [64]. In general models segment the body into individual limbs and a torso, to which individual motion models are then attached. Because limb movement is strongly constrained motion models tend to work well. Sukthankar *et al.* present an example motion model in their work on learning human motion models for different activities [133]. Once a motion model has been derived it can be used to predict object pose given the object’s current state using sampling. As the next image frame arrives the proposed models can be projected into the image plane and the similarity between the predicted and observed motion can be determined using an evaluation function.

The key limitation with model based tracking is that the models must be derived before tracking can begin. Furthermore, their computational complexity has implications for real-time processing, especially when multiple objects must be tracked concurrently in a complicated scene.

2.3 Trajectory Models

Trajectory based behaviour models have been particularly popular with vision researchers. Wang *et al.* use Optical Flow Analysis to detect abnormal traffic behaviours in [143]. After detecting moving pixels they quantize them using a codebook based on pixel position (10×10 cell) and direction of motion. The video sequence is then segmented into ten second clips which gives sets of spatio-temporal features, and can then be clustered using techniques based on Latent Dirichlet Allocation [21]. In this context the video clips represent documents, and the moving pixels represent words. Each video clip is modelled as a mixture of K topics, where each topic is a distribution over the dictionary of words (moving pixels).

Using this approach Wang *et al.* show that the ‘normal’ motion trajectories within a traffic scene can be detected, including correlated trajectories. For example, they show that topics associated with traffic, and those associated with pedestrians crossing the road are correlated such that they rarely occur together. This gives rise to the activity ‘pedestrians cross the road when there is not much traffic’. Wang *et al.* can also detect abnormal behaviours in videos, indicated by uncommon topics occurring (e.g. traffic driving the wrong-way up a one-way street).

Aside from the fact that Wang *et al.* have focused on detecting abnormal behaviour, a key

difference from this research is that they have focused solely on motion trajectories. Their approach has no concept of scene semantics and this limits recognition to behaviours exhibiting distinct motion trajectories. Because of this fact it is the author's opinion that their approach is better suited to well constrained environments where motion is limited by strict constraints.

Antonakaki *et al.* also considered motion based analysis, focusing on short term motion anomalies (e.g. abrupt changes in motion) and long-term trajectory anomalies [8]. Using background subtraction they extract the location of moving people and construct a set of local features. These include object centroid, width, height, mean speed and histogram. Using these feature vectors a Hidden Markov Model can be trained to model 'normal' trajectories', from which abnormal behaviours can then be identified by their low probability.

To model the short-term behaviours they train a one-class Support Vector Machine (SVM). SVMs construct a hypersphere around a set of training points in multi-dimensional feature space [92]. Training the approach with a set of 'normal' motions allows the classifier to identify abnormal motion for each frame of video. Taken into context with the previous 24 frames of video, the SVM-based classifier uses the percentage of 'abnormal' frames to detect anomalous motion.

For visual surveillance there are a number of limitations with this approach. With regards to motion based abnormality there are many instances when abrupt changes in motion can occur. This is particularly true in public transport hubs, where somebody may suddenly run or stop running (for example to catch a train). As with Wang's approach, trajectory based recognition and anomalies are best suited to very constrained environments, and are not suitable for detecting complex human behaviour in most domains.

To introduce one final trajectory based technique, Dee and Hogg used a cost based approach in [34]. They combine a set of goal locations with knowledge about the scene. For instance, in a car-park scenario an agent may wish to get from their car to an exit. The exit is therefore considered a goal. They assume that agents perform goal-directed behaviour and that trajectories are formed from piecewise linear components. Crucially, a state transition model is associated with the goals that are consistent with an agent's trajectory, with a cost with each goal transition. As a result, paths with fewer transitions will have lower cost than those with more transitions, from which a pseudo likelihood can be computed.

Cost based approaches are rare in the literature and this approach proposes some novel

behaviour recognition techniques. One can see that in constrained environments where the number of ‘normal’ behaviours is limited it may be unnecessary to reason about behaviour at a semantic level, although the limitations of trajectory based recognition still stand. That said, similar ideas could be applied to non-trajectory based applications by interjecting semantically meaningful goals. As before, a cost could be associated with goal transitions and this could prove an interesting technique. In this respect the approach is somewhat similar to bag-of-features inference, which penalises unexpected observations via a low probability.

Summary

To summarise, it is often argued that interesting behaviours rarely occur in surveillance video, while there is a wealth of un-anotated video containing ‘normal’ behaviour. Modelling this behaviour allows the ‘normality’ of a novel video to be assessed and identifies interesting behaviour by its lack of representation in the training data. Several authors have proposed approaches based on agent trajectory and are particularly well suited to constrained environments in which normal trajectories are limited (e.g. traffic domains). However, there are a limited number of behaviours that can be modelled by trajectories, alone and this is a major limitation of these approaches.

2.4 Semantic Models

One approach for modelling complex behaviours is to define the semantic relationships between the components. These relationships may be in the form of temporal and spatial constraints, or compositional rules. Because of the complexity of these rules they are normally specified by an expert rather than extracted through model learning. However, semantic models are largely deterministic and treat states as facts (e.g. [49, 52, 78, 69, 68]), or at the very least assume that they are detected with a high degree of accuracy (e.g. [46, 124]). Furthermore, the mechanisms for dealing with observation uncertainty are not traditionally available [75]. There are generally three types of semantic model: Logic Models, Grammars, and Constraint Satisfaction.

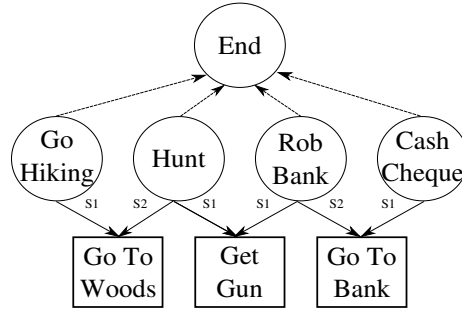


Figure 2.8: A Goal Consistency Graph in which observations (rectangles) can be explained by different goals. Inconsistent goals are removed from the graph allowing predicates to be formed. Link labels (e.g. S1) indicate action order.

2.4.1 Logic Models

Logic models represent some of the earliest methods of recognising human behaviour (e.g. [68, 69, 78]). Kautz presented one such approach, proposing that behaviour could be explained by constructing a plan library from first-order axioms [69]. A plan library is a set of goal taxonomies: hierarchical decompositions of abstract goals into sub-goals and actions. For example, Figure 2.8 illustrates a hierarchy with four possible plans and three possible actions, where link labels (e.g. S1) indicate the action order. A *Get Gun* observation is consistent (can be explained) by the predicate $Hunt \vee Rob\ Bank$. The *Rob Bank* goal becomes inconsistent upon further observing *Go To Woods*, which makes *Hunt* the only consistent goal.

One of the major limitations of this approach is that it is unable to deal with observation and goal uncertainty. If the observations can be explained by the predicate $Hunt \vee Rob\ Bank$, one would expect the *Hunt* goal to be more likely given prior knowledge, yet this approach has no mechanism of using or communicating probability. Furthermore, there is an assumption that all actions are purposeful, which has implications for noisy sensors generating false-detections and uncollaborative agents (e.g. deliberate deception).

More advanced logic-based approaches have also been proposed, including a representation based upon Probabilistic Horn Abduction (PHA) [117] by Goldman *et al.* [120]. PHA uses prolog-like rules specified via propositions that allow hypotheses to be made. Each hypothesis can have an associated prior probability, and thus more likely explanations can be chosen over less likely ones and some reasoning about uncertainty can take place. However, the approach is still restricted by an inability to handle observational uncertainty.

2.4.2 Constraint Satisfaction

Constraint satisfaction is closely related to logic models in that they too use predicates to specify behaviours. These predicates model behaviour using rules (e.g. geometric relations, ordering), which overlap with the way in which humans describe complex events [75]. Constraint satisfaction has been particularly popular for recognising high-level behaviour from video, with [53, 46, 106, 58] all adopting constraint based approaches.

For example, Fusier *et al.* use surveillance of aircraft arrival procedures in [46]. They segment the foreground and background components using background subtraction and track objects using feature based tracking before classifying objects as people, ground vehicles, aircraft or equipment. To model behaviours they introduce VERL (Video Event Description Language), which specifies spatio-temporal rules that take advantage of a priori knowledge about the scene. For example, a priori knowledge might include geometric zones of interest such as loading areas, exit points or entrances.

They also define several types of primitive and composite states. A primitive state is a directly observable phenomena (e.g. vehicle in zone), while a composite state involves two or more concurrent primitives (e.g. vehicle located and stopped inside a zone). They also define primitive events (e.g. vehicle leaves zone) and composite events (e.g. stopped vehicle in zone then leaves). In many respects these events are similar in definition to the primitive and complex features used in this thesis.

Using VERL to reference these events and states Fusier *et al.* specify spatio-temporal rules using constraints such as during and before, as shown in Figure 2.9. This example involves a vehicle and a person entering and stopping in specific zones. Although they report that several complex behaviours could be recognised, including multi-agent behaviour, one of the major limitations of this work is the amount of prior knowledge used. For instance, the extensive use of ‘zones’ helps simplify the problem by eliminating potential explanations. In many domains, and especially a more generalised surveillance problem, one cannot segment the environment so distinctly. Furthermore, in a busier scene visual occlusion is more likely, increasingly the likelihood of ‘missed’ observations. Under such conditions constraints may remain unsatisfied, and no detection can be made.

Scene segmentation has also been used by Robertson and Reid in [106]. They segment each environment into logical zones, as illustrated in Figure 2.10 in which four zones have been defined. However, unlike Fusier *et al.*, they use motion information (e.g. location,

```

CompositeEvent(Aircraft_Arrival_Preparation,
PhysicalObject((p1: Person), (v1: Vehicle), (z1: Zone),
                (z2: Zone), (z3: Zone), (z4: Zone))
Components((c1: CompositeState Gpu_Arrived_In_ERA(v1,z1))
             (c2: CompositeEvent Gpu_Enters_Gpu_Access_Area(v1,z2)))
             (c3: CompositeState Gpu_Stopped_In_Gpu_Access_Area(v1,z2)))
             (c4: CompositeState Handler_Gets_Out_Gpu(p1,v1,z2,z3)))
             (c5: CompositeEvent Handler_From_Gpu_Deposits_Chocks_Or_Stud(p1,v1,z2,z3,z4)))
Constraints((v1- >Type = "GPU")
             (z1- >Name = "ERA")
             (z2- >Name = "GPU_Access")
             (z3- >Name = "GPU_Door")
             (z4- >Name = "Arrival_Preparation")
             (c1 before c2)
             (c2 before c3)
             (c3 before c4)
             (c4 before c5)
             (c4 during c3)
             (c5 during c3)))

```

Figure 2.9: Defining the ‘Aircraft_Arrival_Preparation’ behaviour in the Video Event Description Language, taken from [46].

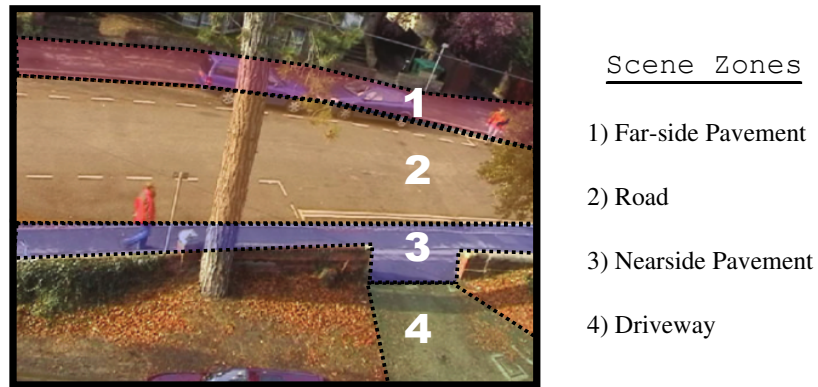


Figure 2.10: The scene is segmented into four zones to assist inference (taken from [106])

velocity) to train a set of Hidden Markov Models to recognise low-level behaviours such as running and walking, and simple events such as ‘person crossed the road’.

Robertson and Reid detect complex behaviour using sets of rules, events and states. *States* represent facts about the scene, such as people being ‘together’ or ‘not together’, and are themselves detected via simple rules such as proximity. On the other hand, *events* represent activities such as walking. To illustrate their use in a high-level rule consider the behaviour definition below:

Behaviour: Agent crosses road to meet other agent
Event: (move-to-road, t_1)
Event: (move-to-pavement, t_2)
Constraint: t_1 before t_2
Fact: Location(t_1)
Fact: Location(t_2)
Constraint: Location(t_1) \neq Location(t_2)
Event: (meeting, t_4)
Constraint: t_3 before t_4

This behaviour can be summarised as:

“IF the event ‘move-to-road’ is followed by event ‘move-to-pavement’ AND the current location is not the same as the location triggering the first event (i.e., the road is crossed) AND, subsequently, a meeting takes place THEN the explanation is that, ‘the agent crossed the road to meet the other agent’.”[106]

Although Robertson and Reid use Hidden Markov Models to probabilistically recognise low-level activities their high-level behaviour rules do not reason about uncertainty. As with other semantic models this means that they cannot communicate behaviour likelihood or reason about the uncertainty of lower-level detections.

2.4.3 Grammars

Several authors have identified links between natural language processing (NLP) and behaviour recognition, with Geib and Steedman presenting a strong case for the closer integration of NLP and behaviour recognition research [52]. They explicitly map the behaviour recognition problem into the NLP domain by stating that both domains are concerned with inferring the underlying meaning for a sequence of tokens, and suggest that results from NLP could inform behaviour recognition work. In the NLP domain these tokens often arrive in the form of letters and words and are often received via audio observations, but are essentially the same as ‘activity’ tokens observed via some other means. By considering grammar rules as goal rules a range of NLP inference algorithms can be adapted to infer behaviours.

Within this area of behaviour recognition a number of different models have been proposed including the probabilistic context free grammar (PCFG). A PCFG consists of a number of expansion rules with associated probabilities. For example, in the simplified traffic model presented by Pynadath [118] there are two expansions of the ‘pass’ behaviour:

Pass \rightarrow Left Right (0.9)

Pass \rightarrow Right Left (0.1)

The rules tell us that when a driver passes another vehicle this is most likely accomplished ($P = 0.9$) by moving into the left lane and then back into the right (US traffic regulations), while there is a smaller probability of them moving into the right lane and then back to the left ($P = 0.1$). However, some concepts do not translate into the PCFG representation, such as the idea of agent/world state. For example, an agent may be less likely to over take when they are approaching their exit. World/agent states often influence agent actions but cannot be incorporated into PCFG based approaches.

More sophisticated models such as Probabilistic State Dependent Grammars (PSDG) do allow for state depended rules [119]. Fundamentally the approach is very similar to PCFGs, with the key difference being that expansion probabilities are now depended upon state information. For instance, the expansion **Drive** \rightarrow **Exit** represents a transition from normal driving to exiting a highway, and would have a higher probability when it is sensed that the agent is nearing their exit.

However, Nguyen *et al.* highlighted that most grammar research has focused on high-level inference, and has not addressed the issue of noisy observations from low-level sensors [107].

Conclusion

Logic models, constraint satisfaction and probabilistic grammars have all been proposed for semantically recognising complex behaviour. Logic models were considered in some of the earliest research and often used consistency graphs to identify the minimum number of goals required to explain observations. Some models incorporated behaviour priors to

facilitate uncertainty reasoning, but logic models as a whole have been replaced by more powerful techniques such as probabilistic grammars. The Probabilistic State Dependent Grammar has been a popular choice in this area and can use agent state information to alter token expansion probabilities at run-time. Although grammar based approaches are more powerful than their logic-based predecessors they share a common limitation: an inability to reason about observation uncertainty. In other words they treat observations as facts, which is not ideal for recognising complex behaviour from sensor based environments. Indeed, both of these techniques have been more commonly employed by researches engaged in theoretical work without noisy observations.

In contrast, those researching video understanding often use constraint satisfaction to identify behaviour. This too treats observations as facts, but probabilistic video processing techniques are often employed to consider some level of observational uncertainty. In this way behaviour inference is only performed upon observations with high probability. Grammar and logic approaches could also be applied in a similar way and with further research this may prove to be effective, although no direct comparisons of grammar/logic-based models and constraint satisfaction exist. Furthermore, constraint based approaches rely upon all activity being observed, so cannot directly reason about partial behaviours. Lastly, constraint and logic based-approaches cannot compare the likelihoods of competing hypotheses, nor communicate the certainty of detections.

2.5 State Models

State models provide an alternative set of approaches for behaviour recognition and use semantic knowledge to capture the state of a system in space and time [75]. Within a behaviour recognition context these models generally include structural information such as the relationships between different behavioural components, and are often combined with abstraction schemes. In most circumstances human intuition is relied upon to provide this structural information, while model parameters are often learnt from training data. This allows state models to recognise more complex behaviour than trajectory models. Furthermore, state models have made extensive use of Bayesian techniques to reason about uncertainty, and in doing so have overcome many of the limitations of semantic models. In this section several state-based approaches will be introduced.

2.5.1 Hidden Markov Models

Oliver *et al.* introduced the Layered Hidden Markov Model (LHMM) for inferring office behaviours from video, audio and computer observations (e.g. mouse movement) [112]. Using the LHMM as a fusion mechanism they successfully recognise six scenarios included a person giving a presentation and face-to-face conversation. The LHMM is a hierarchical (2-layer) extension of the standard Hidden Markov Model and is illustrated in Figure 2.11 (taken from [112]).

Focusing initially on the video observations, a feature vector is constructed for each frame and includes the density of skin tones, motion, faces detected and foreground components. From these feature vectors a bank of HMMs categorises the scene as containing zero, one, or many persons present. This categorisation then forms one component of the feature vector at the second layer of HMMs.

Audio signals are processed in a similar way, constructing a feature vector and using a second bank of HMMs to identify types of audio. The audio categorisation becomes the next component of the second layer feature vector, which is completed using information about computer activity (mouse movement). Having fully constructed the second layer feature vector a third bank of HMMs is queried to identify the most likely meeting room activity. This can be seen in Figure 2.11.

One of the benefits of this hierarchical model is that data from different sensors can be fused to facilitate high-level inference. For instance, video-based observations are processed independently from audio-based observations before their ‘joint’ meaning is considered. This means that, in theory, any number of different sensing techniques could be employed to detect low-level behavioural attributes. Furthermore, segmenting the state-space in this way significantly reduces the HMM state spaces, simplifying the training process.

However, hierarchical models bring another potential benefit: behaviour decomposition. Although not entirely explored by Oliver *et al.*, Blaylock and Allen go further using their Cascaded Hidden Markov Model [19]. This is structurally very similar to the Oliver’s LHMM, but rather than fusing observations from different sensors they focus on the decomposition of complex behaviour into simpler components (sub-goals/actions). This is shown in Figure 2.12.

To model a decomposition of depth D a set of D stacked banks of HMMs are used where

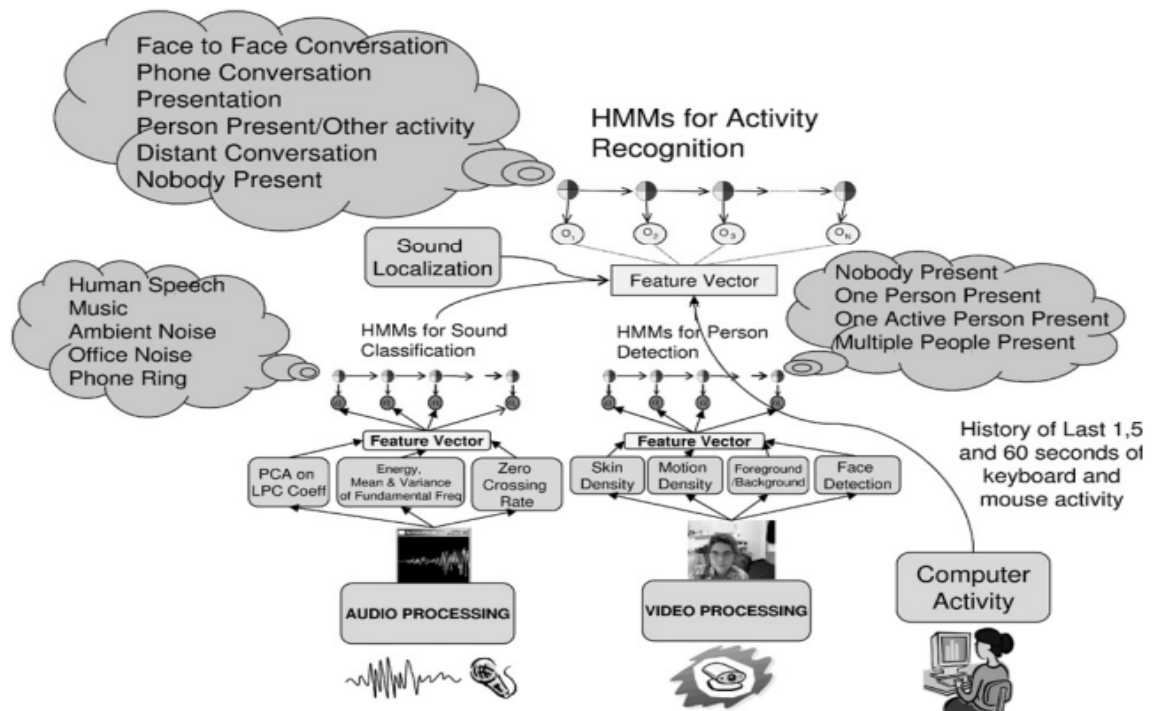


Figure 2.11: The multi-modal SEER architecture from [112]

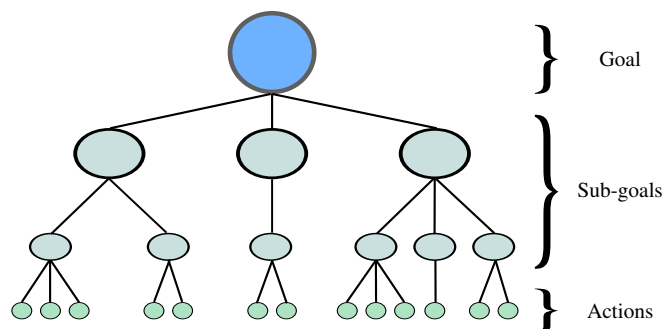


Figure 2.12: Blaylock and Allen decompose high-level behaviour into lower-level sub-goals and actions.

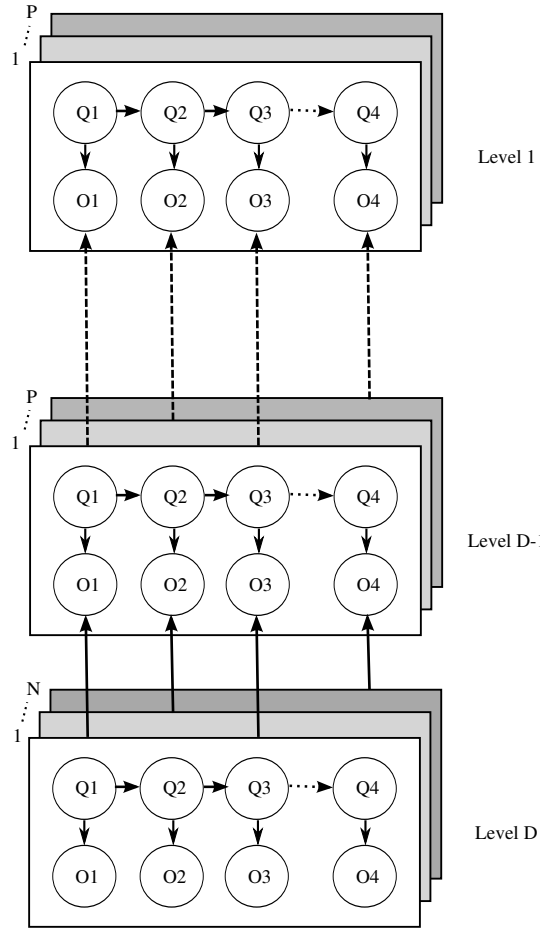


Figure 2.13: A Cascading Hidden Markov Model

the most likely HMM from each bank emits a state observation to the bank above. These state emissions correspond to hidden states in the HMM above it. Recognition commences at the bottom of the stack with the observed agent actions forming input. Thus, at each time step there is a chain of hidden state variables connecting the observed input (at level D) to the highest-level of abstraction (level 1). This is illustrated in Figure 2.13.

Blaylock and Allen train each level of their model individually, using synthesised data [105] from the Monroe corpus [20]. This corpus is an emergency response planning domain with goals such as ‘clear road hazard’. They demonstrate that in a model of depth 8 performance varies at different levels of the model, although they state that this is primarily related to the complexity of the state-space at each level. However, one of the limitations of their work is that the nature of their observations is unclear, especially with regards to noise. Because they use synthesised data to both train and test their model one cannot be certain how the model would perform in a noisy, real-world application.

Although powerful, one of the limitations of HMMs is that their latent variables consist of only one node. However, there are occasions when the agent’s state can be more

appropriately modelled using several latent variables. For these applications Dynamic Bayesian Networks (DBN) have been used. The reader may recall from Section 2.1 that HMMs are in fact the simplest form of DBN, and thus it is not surprising that a more complex DBN is often required to model complex behaviour.

2.5.2 Dynamic Bayesian Networks

There are several examples of behaviour recognition models based on DBNs, including some that are ambiguously named such as the Abstract Hidden Markov Model (to be discussed shortly) [24]. However, in surveying the literature one can also note that most of these approaches are implemented using Sequential Monte Carlo sampling (particle filtering). One of the reasons for this is that by increasing the complexity of the models the run-time performance is reduced. This hinders real-time inference. By using particle filter approximation only a subset of the state space is explored and real-time performance can be recovered.

Furthermore, particle filtering is a recursive formalism, *updating* probability estimates as new observations arrive. This approach naturally lends itself to on-line inference, allowing predictions to be made and refined at each observation. In contrast, Hidden Markov Models consider a window of observations so are less efficient for on-line inference.

Although there are various examples of particle filtering in the literature the work of Bui and Venkatesh [24] is very relevant to this research. Bui and Venkatesh presented the first successful application of particle filtering to behaviour recognition and used a Rao-Blackwellised Particle Filter [38]. Their work considered agent behaviours within an indoor environment of corridors and rooms. Each location in the environment was segmented into seven regions to provide a discrete state-space of agent locations as shown in Figure 2.14. In the figure one can see that the agent enters the corridor (region 1) from the right and moves to the linux server (region 3) in the Vision Lab (region 5). The agent then moves to the printer (region 4) before returning to the server. After exiting the room they briefly enter Office 2 (region 7) before leaving the scene from the left end of the corridor. At the modelling level they consider behaviour at the three levels of abstraction shown in Figure 2.15.

In addition to decomposing behaviours into sub-behaviours they also make the assumption that the current behaviour is only dependent on the agent's current state. That is, when an agent enters the Vision Lab only Vision Lab behaviours need to be considered, and

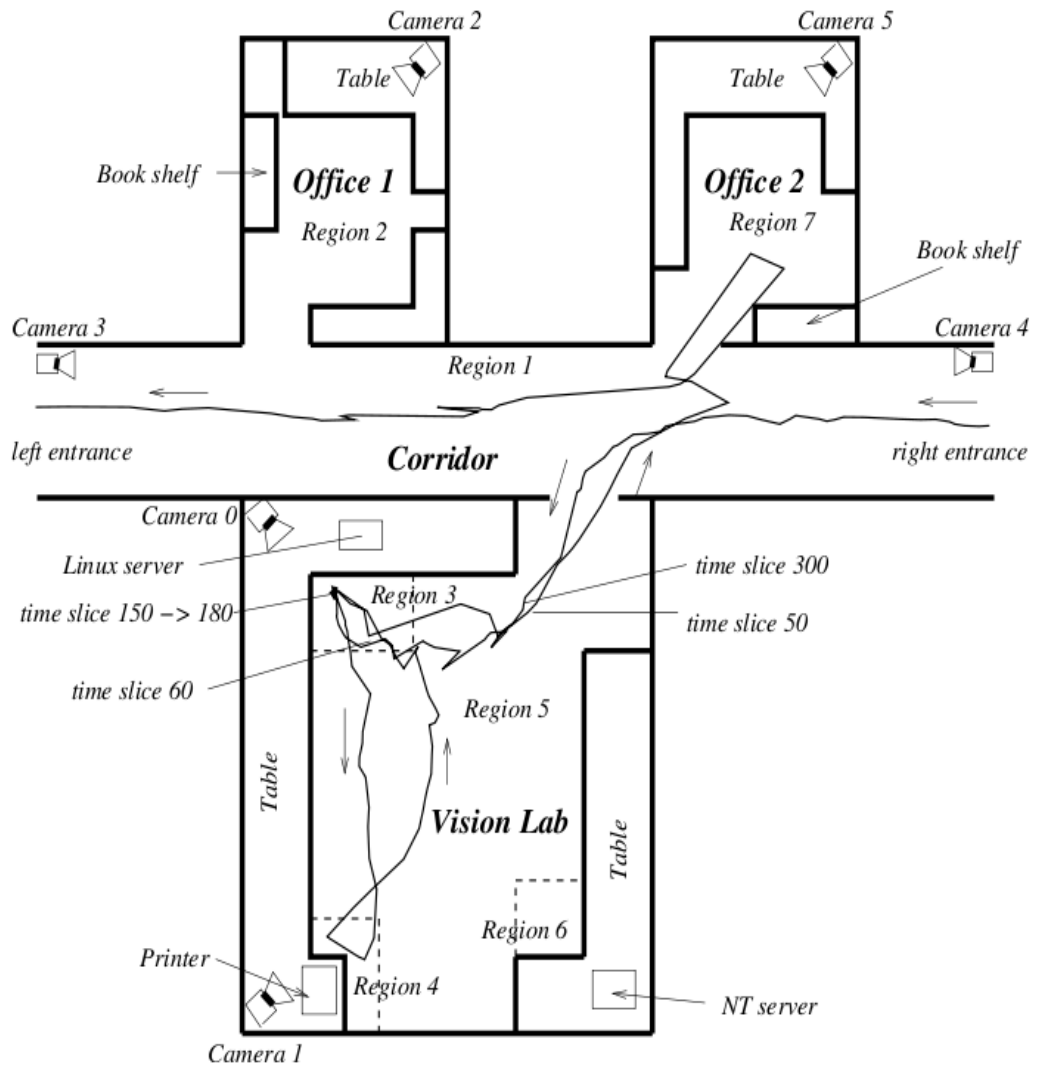


Figure 2.14: The environment and the trajectory of a person, taken from [24]

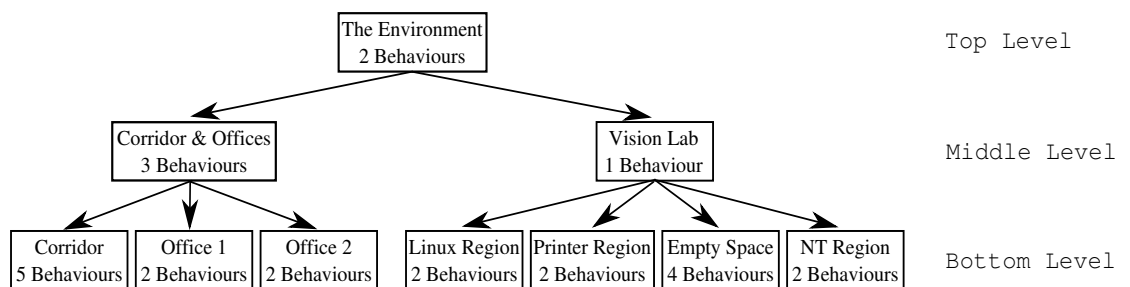


Figure 2.15: The region and behaviour hierarchy from [24]

the next Middle Level state does not change until the bottom level (Vision Lab) behaviour terminates. Furthermore, behaviour termination is easy to detect because they are region dependent, and thus a Vision Lab behaviour terminates when the agent transitions from the lab to the corridor. To model behaviour they use a hierarchy of states to represent the active behaviour at each behaviour level, where each particle in their filter is instantiated with a behaviour hierarchy.

Assume that each particle is initialised with a random active behaviour status. At each time-step each particle predicts the agent’s next region using a trained trajectory model corresponding to the current bottom level behaviour. Each particle is then weighted according to the probability of the observation given the current state ($P(y_t|x_t)$). If a behaviour transitions to a termination state the behaviour at the level above (middle or top) is allowed to transition, selecting a new sub-behaviour for the level below.

In their results Bui and Venkatesh show that the behaviour being observed has a higher probability than all other behaviours, although they do not consider the problem of actually classifying behaviour. Nevertheless, their approach provides encouraging support for the use of particle filtering to perform approximate inference on DBNs. Indeed, this work is frequently cited as inspiration for other particle filter wrapped DBNs, including work by Nguyen *et al.* in which two model variations are presented and evaluated on similar, region based problems [108, 107].

One of the contributions of Bui and Venkatesh is that of identifying an effective mechanism for approximate DBN behaviour recognition. Indeed, their underlying DBN has not been reused, but Rao-Blackwellised particle filtering has become increasingly popular as a mechanism for behaviour recognition. A further example can be found in [83] in which transportation routines are inferred from GPS coordinates by Liao *et al.* Their underlying DBN consists of eight latent variables to indicate (amongst other things) the person’s speed and velocity, mode of transport (e.g. bus, car) and trip segment (e.g. road), and is shown in Figure 2.16.

Starting at the top of the model, the novelty variable indicates whether ‘normal’ behaviour is being performed. When true, all other variables follow transitions learnt from historical data, but when set to false, random transitions occur. The g_k variable is used to indicate the goal location of the agent (e.g. home, work), while t_k indicates the trip segment (comprised of a start location, end location and mode of transport). The trip segment influences the route an agent takes, for example, by restricting movement to the bus route. Variable m_k indicates the current mode of transport and influences the agent’s velocity and location. Finally, the ‘switching’ nodes are used to indicate when a change of mode

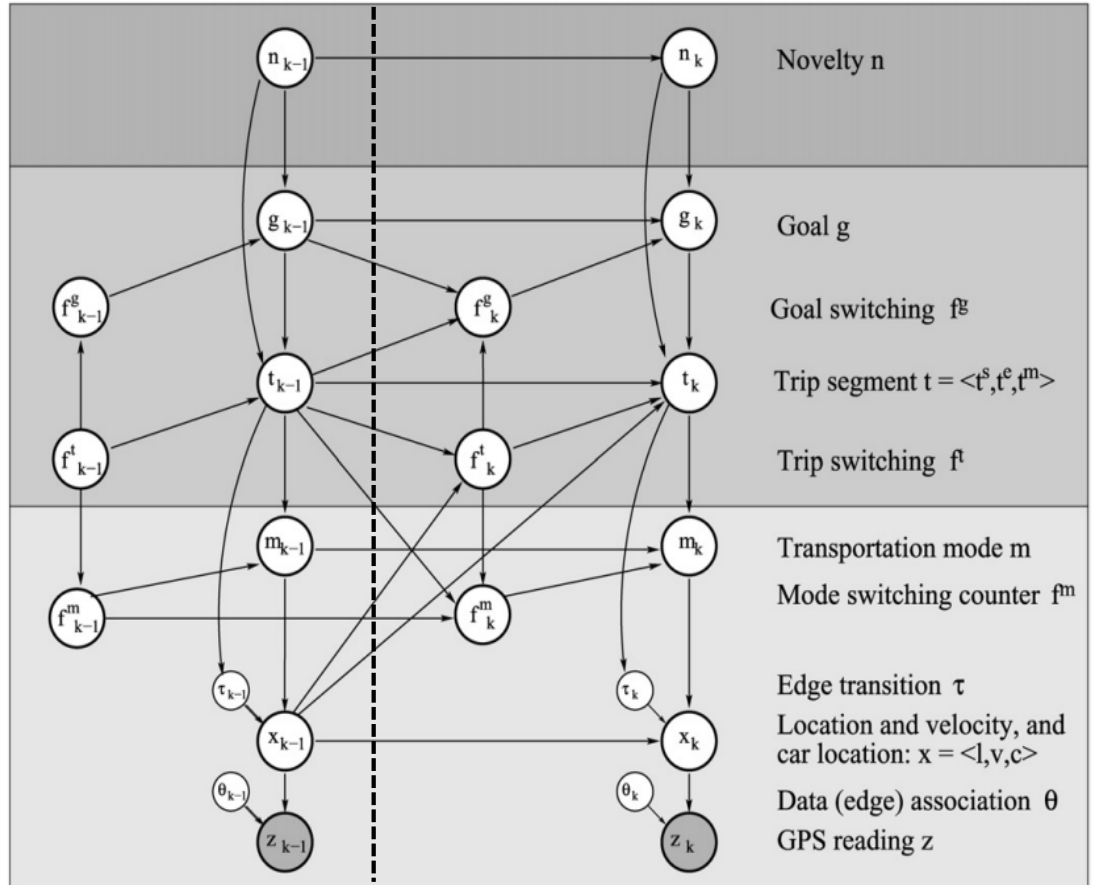


Figure 2.16: DBN representing a person's outdoor movements during everyday activity. The upper level is used for novelty detection, and the middle layer estimates the user's goal and the trip segments he or she is taking to get there. The lowest layer represents the user's mode of transportation, speed, and location. Two time slices, k and $k - 1$, are shown. (Taken from [83]).

will occur shortly, for example when waiting for a bus to arrive.

It is not important for the reader to fully understand Liao’s model (see [83] for a comprehensive introduction), but it is important to highlight that each particle in their particle filter encapsulates this DBN. This is quite unlike traditional particle filter applications, which are often used for person tracking applications and employ some combination of continuous distributions. In contrast, Liao, Bui and Nguyen have all made use of discrete variables in their filters allowing particles to represent very complex states.

2.5.3 Summary

This section has introduced several state models that are particularly relevant to this research. The Layered Hidden Markov Model (LHMM) and Cascaded Hidden Markov Model (CHMM) both use banks of HMMs in a hierarchical structure. In the LHMM the two levels allowed different types of data to be fused, processing audio, video and computer information separately before combining simple activity classifications to form a higher-level feature vector. This vector was then provided as input to the upper bank of HMMs from which complex behaviour was detected. The CHMM is structurally very similar but has been applied to recognise behaviours at different levels of abstraction. At each time-step the lowest bank of HMMs receives an observation, and the model with the highest probability of having generated the observation sequence emits a state. This state is then cascaded into the next bank of HMMs one level up, a process that repeats until a top-level goal classification is made.

These two models both extend the standard Hidden Markov Model; a type of Dynamic Bayesian Network. The distinguishing feature between HMMs and DBNs is that a HMM only has one hidden variable, while DBNs can have many. The increased modelling complexity of DBNs is one of the driving factors that has led to some researchers adopting DBN-based approaches instead of HMMs.

Although particle filtering is more traditionally applied to tracking problems, Bui and Venkatesh showed that it can also be used to perform efficient inference on complex DBNs, especially when combined with Rao-Blackwellisation. Researches inspired by their approach have derived models for several different applications, and have often been demonstrated on noisy, real-world data. Unlike semantic models particle filter wrapped DBNs can reason about observational uncertainty, and their improved modelling power means they can recognise more complex behaviours than, for example, simple trajec-

ries.

However, in all cases these models have been trained using corpora, preventing their direct application to data-scarce domains. Furthermore, the underlying DBNs have not been application dependent and this has limited the generality of prior research. Removing these limitations has become a key goal of this research and distinguishes our work. Furthermore, inspiration is drawn from the LHMM and CHMM by replacing the banks of HMMs with particle filters. In this way abstraction can be achieved, simplifying the behaviour modelling process.

2.6 Feature Models

Given that our approach is based on the idea of feature-based recognition a review of the literature would not be complete without some discussion of similarly inspired work. Object detection is one area in particular where feature-based approaches are abundant, and is concerned with learning and detecting the presence of objects in static images and video. Identifying a robust modelling approach is a key challenge in this domain, as objects may undergo operations such as rotation, translation and scaling. This challenge has led to the development of a number of different techniques for identifying invariant object features. For instance, it is not uncommon to transform an object image into the frequency or scale domains [88, 90], where invariant salient features can be more readily identified.

Feature based methods are also popular in text-document classification, and several behaviour recognition approaches have built upon probabilistic Latent Semantic Analysis (pLSA) [59] and Latent Dirichlet Allocation [21]. A key similarity with our own work is that these approaches adopt a bag-of-words assumption: that is, the order of words in a document can be neglected. Our approach is similarly defined; the exact order of behavioural components can be neglected in the representation, and loosely imposed during recognition.

Niebles *et al.* build upon Latent Dirichlet Allocation to achieve behaviour recognition, but focus on recognising low-level behaviours [109]. Their features are based on the spatio-temporal properties of small moving regions, and they show recognition of repetitive behaviours such as hand clapping and waving. However, they also acknowledge that their features have no semantic meaning, and the lack of temporal information prevents their approach from recognising more complex, non-repetitive behaviour. In fact, this

is one of the key distinctions between their approach and ours: we only ignore temporal information at the modelling stage. Core to our approach is the idea of combining features with *behaviour evolution* to re-impose a weak ordering at the recognition stage. This ensures that as a behaviour evolves, the features observed match those that are expected.

Xiang and Gong used clustering techniques and unsupervised learning to identify both normal and abnormal behaviours in [145]. During the training phase they segment video sequences into sets of frames representing low-level behaviours. This segmentation can be performed in several ways, for example, using fixed temporal windows or using ‘non-activity’ regions (frames with little movement) as boundaries. Once segmented, object detection and tracking is performed and a discrete 7-dimensional feature vector (including centroid, dimensions and motion) is associated with each foreground object. Clustering is then used with automatic model order selection [127] to identify distinct low-level behaviours.

During the recognition stage frames are categorised as normal or abnormal by calculating an anomaly measure. This is based on the likelihood of each cluster having generated the observed frame. Using a threshold approach (e.g. 80% of frames in 10 second window are normal), normal sequences can be classified using the known behaviour clusters while abnormal sequences are flagged.

This work is interesting because it has gone some way towards removing the need for annotated training data, but suffers two key limitations: 1) inference cannot be explained to operators, and 2) the validation is constrained to low-level behaviours involving few agents. In fact, their validation is performed on aircraft arrival video (also used by [46]), but the behaviours recognised include ‘aircraft arrives’, and ‘air-bridge connects’. The complexity of these behaviours is very limited and involves few agents, and it is unclear how well the approach would perform in more complex environments. This is especially true for more typical surveillance scenes involving multiple interacting and non-interacting agents. In this respect it is unlikely that meaningful clusters could be learnt using this non-semantic approach.

Boiman and Irani propose abnormal behaviour detection using image re-composition in [22]. They pose the problem as a puzzle in which one tries to compose newly observed video using data extracted from previous video examples (a database). Regions of data that match larger contiguous chunks from the database are considered more normal, while regions that are more difficult to re-construct have a higher likelihood of abnormality. This process of re-composition is illustrated in Figure 2.17, in which a newly observed image (a) can be recomposed (b) using three images from the database (c).

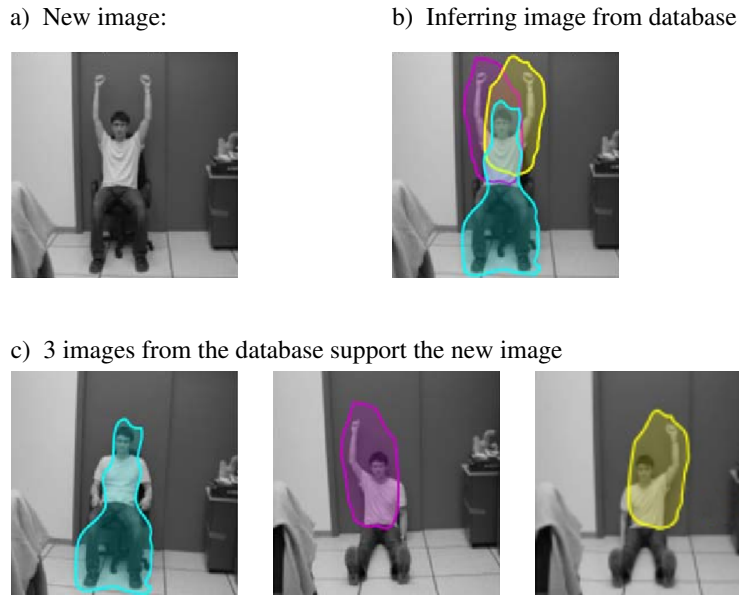


Figure 2.17: The basic concept of image composition taken from [22]. A region in the new image is considered likely if it has a large enough contiguous region of support in the database. New ‘normal’ images can thus be inferred from the database even when they have not been previously observed.

To extend this process to the temporal domain, small local descriptors are extracted for short sequences of video (4 frames), and given a new set of video frames the likelihood of each pixel can be calculated. Boiman’s results are very impressive, but suffer a limitation in that subtle abnormality cannot always be detected. For instance, assume that the ‘Watched Item’ behaviour from Chapter 1 was represented in the video database. Now suppose that an agent enters the scene, places a luggage item next to an arbitrary person and then exits the scene. It is likely that this agent’s behaviour could be matched to existing database sequences, yet the subtlety between this behaviour and ‘Watched Item’ is that the agents are not associated (enter independently). In other words, these behaviours are semantically different, but not very different visually. This is one of the primary limitations of this approach, aside from an inability to recognise specific known behaviours.

Summary

Feature based recognition is common for object/document classification but has had limited application in behaviour recognition. One of the reasons for this is that it is difficult to encapsulate the long-term temporal components of a behaviour into a feature. Without this only very simple behaviours can be recognised, as demonstrated by Niebles *et al.* in [109]. Xiang and Gong used high-dimensional feature vectors in their work and were able to recognise slightly more complex behaviour, clustering feature vectors to automatically identify simple behaviours [145]. However, these clusters had no semantic meaning, so

recognised behaviour could not be explained. A similar problem is suffered by Boiman and Irani in [22], which again cannot semantically explain detections. Furthermore, their approach focuses on anomaly detection only, so cannot identify individual instances of known behaviour.

2.7 Multi-agent Models

In some cases recognising multi-agent behaviour is a relatively minor extension from the single-agent scenario. This is particularly true when the number of agents is fixed and behaviours involve all agents. Sukthankar and Sycara provide a good example of this in their work recognising paired agent combat behaviours [134]. They experiment with this problem in the context of a virtual reality training environment for military combat officers, tracking agent movements to provide (x,y) coordinates. Agent trajectories are then extracted by segmenting coordinates into short, overlapping temporal windows, and Hidden Markov Models are trained on pairs of agent trajectories. Like the HMM-based approaches presented earlier, the most likely behaviour can be determined by comparing the likelihoods of different model parameters having generated the observation sequences. However, the key limitation of this work is that only simple behaviours can be recognised from trajectories alone.

Zhang *et al.* have presented a non-trajectory based approach in [37]. They use a two-level (cascaded/layered) HMM in which the upper bank detects group behaviour. In many respects their approach is very similar to Oliver *et al.*'s Layered Hidden Markov Model which was used to fuse audio and visual information. The key distinction is that Zhang *et al.* explicitly model group behaviour at the upper level of their HMM using a state-space comprised of solo and multi-agent features. For example, the features encapsulates distance of agents from the projector. They recognise meeting-room behaviours including 'white-board presentation with note-taking', and as such has only been demonstrated with behaviours of limited complexity. Furthermore, it is unclear whether their approach is restricted to a fixed number of agents, which would further limit application.

In many environments the number of agents is unconstrained and the agents involved in a behaviour can change. This presents a problem for the approaches above, which assume that the agents involved in a behaviour are known. There is however an alternative approach: to assume that all agents can be involved in a behaviour. If one assumes that behavioural components can be performed by any agent then the problem becomes one of identifying behaviour participants. In such cases other researchers have used domain

```

(agentGoal obj1
  (agent (obj1 (C)) ; Obj1 is always at the center (C)
    (goal obj1_act1 snapToQB (obj1))
    (goal obj2_act blockQBPass (obj1))
    (before obj1_act obj2_act))

```

Figure 2.18: A partial behaviour description of an American Football play from [65]. An agent (obj1) ‘snaps’ to the other team’s Quarterback and block their pass. Because one of the agents is explicitly defined the algorithm does not need to consider the actions of all agents when trying to match this behaviour.

knowledge to assist in the recognition process.

For example, Intille and Bobick noted that in team sports individual players are often assigned specific positions [65]. One can therefore refer to these positions (roles) in the behaviour specification to limit the number of agents’ actions that need to be considered when trying to match a behaviour. Intille and Bobick’s work considers the recognition of American Football plays. Figure 2.18 shows a partial definition for the *s51 play* behaviour and involves two agents. Note that one agent is explicitly defined as the opponents Quarterback. This helps reduce the search space and simplifies the inference process while remaining more extensible than Sukthankar and Sycara’s approach. It is however limited by the fact that it is a constraint based approach, so cannot reason about the likelihoods of competing hypotheses.

Fusier *et al.* have used a similar approach in their work recognising aircraft arrival behaviours (introduced earlier) [46]. Like Intille and Bobick they use constraint satisfaction, but distinguish between different agents by referring to agent types. Their aircraft arrival behaviour (Figure 2.9 on page 43) defines the roles *PI* (Person) and *VI* (Vehicle), which may be fulfilled by any entities of the appropriate types. As before, different hypotheses can not be compared probabilistically, and a further limitation is that in a scene with many entities, multiple agent-role matches might be obtained. While this is not necessarily a problem, it does require a resolution strategy if all matches are not to be accepted. In a constraint-based system developing such a strategy may be challenging, while comparing relative probabilities would be simpler for probabilistic models.

To overcome the limitations of all of these approaches a combination is proposed. The use of agent roles is important for specifying the types of agent that must fulfil behavioural components, while probabilistic inference is important for comparing competing hypotheses. This combined approach is one of the contributions of our work that distinguishes our work from others.

Summary

Although many researchers have developed techniques for recognising agent behaviour, fewer have considered multi-agent scenarios. As a result, this area of recognition has seen less progress, and the development of fewer generic algorithms. The probabilistic approaches that have been proposed allow competing hypotheses to be compared, but have been restricted to environments in which the agents involved are already known. However, this assumption cannot be made for many environments, including visual surveillance. In contrast, several approaches have been based upon constraint satisfaction, and in essence have assumed that all agents may be participants of a behaviour. In some cases behavioural components have been restricted to agents fulfilling particular role properties such as agent type and this helps to reduce the search space. However, because these approaches are not probabilistic, competing hypotheses cannot be compared.

To overcome these limitations a combination is proposed in this research. It has been identified that the use of agent roles is important for specifying the types of agent that must fulfil behavioural components, while probabilistic inference is important for comparing competing hypotheses. We thus present an approach that considers both of these aspects by using combinatorial search to identify the most likely agents involved in multi-agent behaviour. This approach is one of the key contributions of our work, and distinguishes it from other work in the field.

2.8 Conclusions

Recognising real-world agent behaviour requires not only an ability to recognise behaviour patterns, but also requires signal processing and low-level inference techniques. Consequently, this chapter has discussed a broad range of related work.

2.8.1 Video Processing

Processing videos of agents so that their behaviours can be identified is a challenging signal processing task. At its most basic level it requires the identification of foreground objects, object classification and tracking. To process complex scenes require an ability to track agents as they become occluded by static objects and other agents, and even further

complexities are added by groups of agents.

Because of the vast area of this research this chapter has only provided a brief overview of competing techniques. To perform foreground detection adaptive background subtraction has proven to be one of the more robust approaches. For instance, in complex scenes there may be multiple background objects (e.g. trees in front of buildings) which can be modelled in a number of ways, yet competing techniques (e.g. Optical Flow, temporal differencing) cannot accommodate multiple background objects. Furthermore, Optical Flow relies upon objects moving, so cannot identify static foreground objects.

Once foreground objects are detected, object classification is particularly useful for filtering clutter and removing uninteresting objects. For example, a sudden lighting change can produce erroneous foreground objects, but the shape and sizes of these object are rarely consistent with the objects of interest (e.g. person). Using geometric properties to classify objects has been demonstrated as a useful technique in removing such erroneously detected objects and those that are not required.

Finally, object tracking must be performed on objects of interest. Again, object tracking itself is an entire sub-field of research and this chapter has only provided a short overview of the common techniques used in video processing work. Tracking object contours is challenging because they require accurate instantiation, which is often hard to achieve when automatically detecting objects. Furthermore, such objects can only be tracked through partial occlusions, limiting their application in complex surveillance scenes. On the other hand, object features such as centroid can be combined with a motion model to track objects through occlusions, making the approach significantly more robust for surveillance applications. That said, significant occlusions still present challenging conditions, especially when tracking in 2D images.

2.8.2 Probabilistic Behaviour Recognition

At a fundamental level, Bayesian inference techniques allow us to perform probabilistic reasoning on sets of variables and is at the heart of many more complex techniques (e.g. Hidden Markov Models, Particle Filters). Variables can take different forms (e.g. binary, sets), but the power of Bayesian inference is that it allows us to infer the conditional likelihood of a set of variables taking certain values given another set of observed variables.

When modelling non-stationary (evolving) processes, of which agent behaviour is one,

one of the problems is that the underlying state of the process (agent) is unknown. However, the process (agent's actions) is observed, and from this must infer the hidden state. The Hidden Markov Model (HMM) is a generic Bayesian inference technique that is well suited to this problem, although it becomes inefficient for long processes. Furthermore, HMM inference is performed on windows of observations, so is not ideal for on-line inference. To perform online-inference of a long process the Bayes filter is a more appropriate choice. The Bayes filter is a recursive mechanism which allows probability estimates to be updated as each new observation is made.

In actual fact HMMs are a specialised form of dynamic Bayesian network, which allow the hidden state to be comprised of more than a single variable. Bayes filtering can be used to perform inference on more complex dynamic Bayesian networks as well as on HMMs, but a common problem is that as the complexity of the network increases, inference becomes intractable. This is especially true when networks include real-valued variables. To make inference tractable sampling can be used to perform approximate inference, as performed by a Particle Filter.

The need to reason about uncertainty has led to a large number of plan recognition researchers adopting probabilistic inference techniques. In behaviour recognition extensions of the Hidden Markov Model have proven popular, and so too have particle filter based algorithms. This is especially true of more recent work which has often focused on using noisy real-world observations, including video and GPS. However, there is a fundamental limitation with much of this work: the need to estimate model parameters from training corpora. All of the probabilistic models discussed require parameters in the form of prior and conditional probabilities which must be estimated through model training. Here-in lies the problem for data scarce domains such as visual surveillance and defence: annotated training corpora are rarely available so parameter estimation cannot be performed. Although some probabilistic models have been applied in these domains their applications have been constrained so as to make learning possible, or have relied upon the availability of experts to manually estimate parameters.

2.8.3 Alternative Recognition Approaches

Although state models have been popular, they are by no means the only approach for behaviour recognition. Indeed, researchers working with data more closely related to visual surveillance have often adopted alternative techniques for recognising behaviour. Video understanding is a popular topic within vision research, and is the area from which most 'automated surveillance' work can be found.

When comparing approaches from vision research to plan recognition there is a stark contrast in techniques. Vision researchers often only adopt probabilistic approaches for low-level processes like object tracking, and for recognising low-level behaviour such as walking, running and fighting. The temporal and spatial complexities of these behaviours are limited making training data easier to obtain. However, high-level behaviours often involve more complex temporal and spatial constraints, and it is frequently identified that suitable training data is unavailable in vision research. This has primarily led to the use of constraint-based semantic reasoning for recognising complex behaviour.

Constraint-based reasoning employs rules, relations and constraints to identify instances of behaviour. Temporal dependencies such as X before Y can be imposed, and through combination with object classification, rules can apply to different types of entity (e.g. person, vehicle). By segmenting the environment into different regions rules can also refer to specific locations within the scene (Agent A in Zone C), and is well suited to ordered environments. For example, in an aircraft arrival area the locations of vehicles, people and equipment are all limited by procedures, physical constraints and other factors. However, there are many environments that cannot be segmented so easily. Indeed, this is one of the key limitations with existing research. Furthermore, because the notion of probability has been lost, one cannot compare the likelihoods of competing behaviour hypotheses, nor consider observational uncertainty.

Alternative inference techniques such as grammars and logic models suffer similar limitations. Although probabilistic grammars can compare competing hypotheses they treat observations as facts. Prior research has not focused on their use with noisy real-world data, where their performance is more uncertain.

Vision researchers have also proposed two further alternatives: trajectory based inference and abnormal behaviour detection. In trajectory based inference agent motion is detected and tracked to derive agent trajectories. These trajectories are often used to detect anomalies like sudden changes in motion, or to detect trajectories that are uncommon. However, as with constraint satisfaction, trajectory approaches are most suited to ordered environments (e.g. traffic surveillance) where ‘normal’ trajectories are limited. Anomaly detection has also been applied to image regions to find anomalies, using a database of video to re-compose a novel scene. Areas that cannot be re-composed have a higher probability of anomaly, but this relies on anomalous behaviour being visually different from ‘normal’ behaviour. Like trajectory recognition, this limits the complexity of behaviours that can be recognised.

2.8.4 Feature Based vs. Sequence Based

Feature based classification has proven to be popular in both object detection and text document classification. In both of these fields there is an assumption that spatial constraints can be ignored. In the case of object detection pixel relationships are lost when converting the image into other domains (e.g. frequency, scale). Even when using local pixel features (e.g. small regions) global relations are lost. Similarly, in text document classification the frequency of word groups is considered rather than the grammatical structure of the sentences.

Feature-based behaviour recognition is generally applied by removing most or all of the temporal information associated with an action sequence. This is in stark contrast to the majority of probabilistic techniques representing behaviours as sequences of states, which rely heavily on temporal information.

A few researchers have applied feature based approaches to behaviour recognition. Success has been demonstrated using an approach based on text-document classification, but the removal of temporal information limited recognition to repetitive behaviours such as hand clapping. Clustering based approaches have also been presented, but have only been applied to low-level behaviour and are unlikely to scale well to more complex scenarios.

The work in this thesis has also been motivated by object recognition and text document classification, and propose that the limitations of these earlier approaches can be overcome by *replacing*, rather than removing, the exact temporal ordering of a behaviour. By definition, non-stationary processes evolve, but it is proposed that the nature of their evolution does not need to be modelled as strictly as prior probabilistic work has assumed. By replacing the temporal ordering the need for parameter estimation can be removed, and the techniques that have been developed for high-level behaviour inference become available to data-scarce domains.

Chapter 3

Feature-Based Behaviour Recognition

Nomenclature:

D	Desired feature	B^i	The i th behaviour
C	Set of currently achieved features	$Pr(B^i)$	Prior probability of B^i
c^k	The k th element of C	CPT	Conditional Probability Table
T	Set of target features	α	Set of detectable features
I	Behaviour Interruption	α^i	The i th element of α
A	Activity (feature) observed		
y_t	Generic observation at time t		
Z	Collective state encapsulating C, T, D, I		

This chapter introduces the idea that behaviours can be recognised from a sequence of observations without model parameter estimation. This allows recognition in domains where training corpora are unavailable and is a critical contribution of this chapter. The behaviours in question are assumed to be high-level and complex, and are generated as a result of an agent performing actions to achieve a high-level goal. Portions of behaviour often correspond to planning elements such as sub-goals, and thus the underlying concept is that the observance of a set of sub-goals should allow the over-all goal to be identified.

The next section will formally introduce the fundamental principles of feature-based behaviour recognition. This is followed by a graphical representation of the approach as a Bayesian Network which enables Bayesian inference to be employed. Although the

model does not represent the real temporal ordering of behavioural components, a weak temporal ordering can be imposed to take advantage of the fact that behaviours are sequential in nature. This weak temporal ordering is provided via a Dynamic Bayesian Network (DBN), and will be introduced in Section 3.3.

3.1 Principles of The ‘Bag-of-Features’

Motivation for feature-based behaviour recognition is drawn from object detection research, which is concerned with learning and detecting the presence of objects in static images and video. Identifying a robust modelling approach is a key challenge in this domain, as objects may undergo operations such as rotation, translation, and changes in appearance (e.g lighting). This challenge has led to the development of a number of different techniques for identifying invariant object features. To this end, it is not uncommon to transform an object image into the frequency or scale domains [88, 90], where invariant salient features can be more readily identified. Although this approach breaks the pixel relationships of objects, which is counter intuitive, this process does in fact work very well. This knowledge fuels the idea that perhaps the temporal relationships in behaviours can also be removed without hindering recognition.

Drawing parallels between object detection research and other application domains is not uncommon. Csurka *et al.* identified similarities between object detection and text categorisation [31]. In text document classification bags of keywords are used to represent a document class. Documents that discuss the same subject matter can then be detected by comparing their content to the known feature classes. Csurka *et al.* applied a similar idea to object detection, reporting promising results using bags of visual (SIFT [89]) features to represent objects. SIFT features are calculated for small image patches and thus the spatial orderings within the image are not maintained by the features.

The similarities between object recognition and behaviour recognition have also been highlighted before, with Patron *et al.*, identifying that many of the challenges are shared [115]. Recognising these similarities this chapter argues that like objects and text documents, human behaviours can also be represented using sets of salient features. These may be identified by breaking the temporal relationships between different behavioural components, and is akin to splitting pixel relationships for object detection [88, 90]. By removing these temporal constraints the model changes from “*Expect these ordered events*” to “*Expect these events*“, and means that model parameter estimation from corpora is no longer required. There are many domains in which large libraries of training corpora sim-

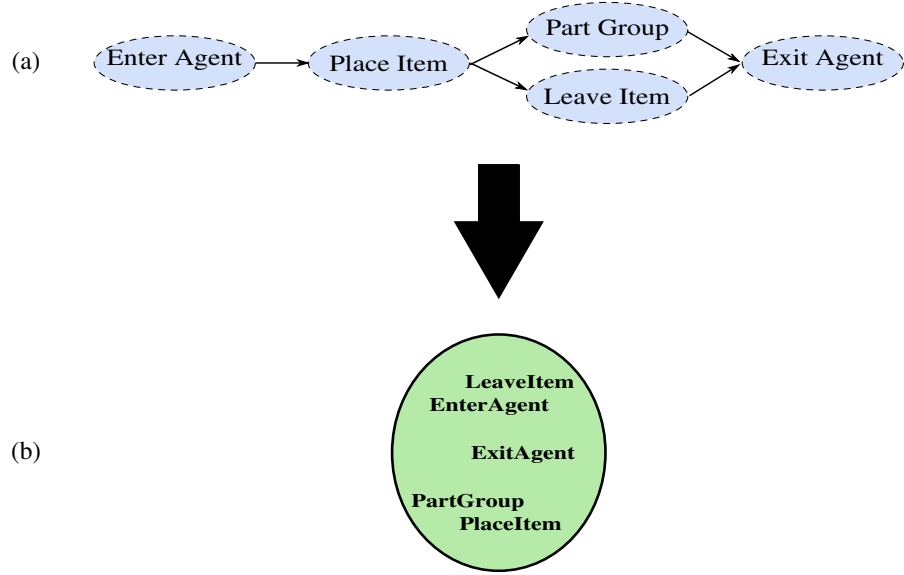


Figure 3.1: (a) The temporal representation of the Watched Item Behaviour. (b) The Bag-of-Features representation of the Watched Item behaviour.

ply don't exist, and thus this approach allows existing inference techniques to be applied to a new set of problem domains.

To give an example, Figure 3.1 shows two representations of the *Watched Item* behaviour. Recall that this behaviour is frequently seen at public transport hubs and represents the scenario where two people are travelling together. One traveller leaves their luggage in the custody of the other while they leave the scene, and thus their luggage is watched by their companion.

Behaviours are traditionally represented as a temporally ordered sequence of activities as shown in Figure 3.1a. However, in a feature-based approach the temporal constraints are removed, as illustrated in Figure 3.1b. The ellipse represents a complex behaviour as a bag of features with cardinality: one. Formally, denote a behaviour/bag by B^i where superscript denotes the i 'th behaviour in a collection, and will refer to B^i as the agent's target behaviour (synonymous with goal). Each bag element is drawn from the set of detectable features α , where a feature represents some activity.

The agent's progress towards a target behaviour can be monitored by tracking the features it generates. Fundamentally, the features should be consistent with B^i if B^i correctly represents the agent's behaviour. For instance, if feature α^i is observed but $\alpha^i \notin B^i$, then α^i must be a false detection, or B^i is not the agent's true behaviour.

As time increases more features of B^i should be generated. If it is assumed that each

element of a behaviour is only performed once, then the set of expected future features are the elements of B^i not yet observed. For example, if $B^i = \langle \gamma, \delta, \epsilon \rangle$ and feature γ has already been observed, then the set of expected features is $\langle \delta, \epsilon \rangle$. This set of expected future features is continually updated and can be used to apply a weak temporal ordering to the elements of B^i . The sets of expected features is conceptually similar to the ‘pending sets’ proposed by Goldman *et al.* in the context of Probabilistic Horn Abduction [120], although they use temporal constraints to add and remove elements from the set.

It is important to acknowledge that the temporal ordering of some features could be imposed without requiring parameter estimation. For instance, one cannot detect that an object has been removed before it has been placed, and an agent cannot exit the scene before they have entered. However, in many respects these constraints do not need to be modelled if it is assumed that real data is being observed. For example, the video processing sub-system cannot detect that an object has been removed if it has never been detected as placed and thus ‘incorrect’ orderings cannot arise. That said, there are undoubtedly behaviours that do contain temporal feature dependencies and this is one area in which future work might proceed.

If C is defined to be the set of currently observed features, and T as the target feature set for some behaviour B^i , then $T \setminus C$ is the set of expected (future) features. At each time step, features in $T \setminus C$ have equal probability, while all other features have zero probability. This probability distribution encapsulates the assumption that each feature is only truthfully generated once per behaviour, and is consistent with other work in the field (e.g. [76]). It should be noted that this limitation is not inherent to the approach, but rather time constraints did not allow for its removal within the confines of this research. Several suggestions for the removal of this assumption are proposed as future work (Chapter 11).

By saying ‘truthfully’ generated that is to say that a goal behaviour cannot require the same feature to be performed twice. That is not to say that the feature cannot be detected twice through miss-detection. Similarly, goal repetition does not contravene this assumption: If a goal is performed twice and thus generates two occurrences of the same feature, each feature is still classed as having been generated by a different goal (if only different in time). Crucially, if the goal-behaviour can be represented without repetition, then it can be represented by this approach.

3.1.1 Worked Example

Using the *Watched Item* behaviour from Figure 3.1 as an example, at time step $t = 0$ each of the 5 features (LeaveItem, EnterAgent, ExitAgent, PartGroup, PlaceItem) has equal probability and $C = \emptyset$. Carry forward that in this example, $P(\text{EnterAgent}) = \frac{1}{5}$. Let us assume that *EnterAgent* is observed at time-step 1 and thus at time-step 2 $C = \{\text{EnterAgent}\}$. Because of the single-occurrence assumption $P(\text{EnterAgent}) = 0$ at time-step 2. Again, the elements of T that are not in C get uniform probability and thus $\forall i \in T \setminus C : P(i) = \frac{1}{4}$. This process repeats until all elements of T have been observed. Note that any element that is not in T has a zero probability at all time-steps. Furthermore, when all elements of T have been observed no further agent activity is expected.

Note that it is assumed that an agent with goal behaviour T will not perform detectable features that are not components of T . Furthermore, for the example at hand it is assumed that false-positive (noisy) detections do not occur, although the next section will explain how false-positive observations are addressed.

3.2 Bayes Network Representation

This bag-of-features approach can be modelled as a Bayes Network. Consider the network in Figure 3.2a which represents the network for a single behaviour. As before, $T = B^i$ indicates that node T represents a target set of features for the i 'th behaviour. Similarly, node C is the set of observed features and is a subset of $B^i : \forall c^k \in C, c^k \in T$. The network also contains two new variables; nodes D and A . Node D represents the probability of a desired feature conditioned on T and C , while node A represents the feature detected and is conditionally dependent upon the agent's desire.

When modelling unknown agents it is not known what their goal behaviours are, nor which elements of those goals have been achieved, or which goal element they intend to perform next. Nodes C, T and D are thus latent (hidden) variables, while variable A is directly observable. If the variables C, T and D are collectively referred to as Z , and the observation A as y , then the network is reduced to the generic form: $Z \rightarrow y$. This gives rise to what is commonly referred to as the observation model: $P(y|Z)$, which expresses the probability of observing y given that the system is in state Z .

Having defined the edges and vertices of the network the only remaining parameter is

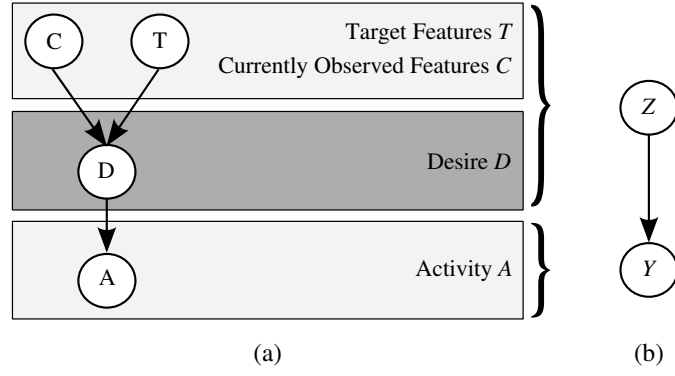


Figure 3.2: a) A Bayesian Network for representing behaviour using features. b) The generalised form

the conditional probability table (CPT). Various techniques exist for deriving the CPT for Bayes Nets, including model learning, and using probabilities defined by experts. However, a fundamental assumption of this thesis is that training data is unavailable, rendering model learning techniques unavailable. This thesis also makes no assumptions about the availability of an expert.

Instead, the CPT can be calculated algorithmically at run-time. This is possible due to the ‘single-occurrence-assumption’ made in section 3.1. Given the previous definitions of T and C , the conditional probability for D is:

$$p(D = \alpha^i) = p(D = \alpha^j) \forall_{i,j} : \alpha^i, \alpha^j \in C \setminus T \quad (3.1)$$

$$p(D = \alpha^k) = 0 \forall_k : \alpha^k \notin C \setminus T \quad (3.2)$$

For simplicity, assume that there is a one-to-one relationship between the agent’s desired activity and the performed activity¹. The observation model can then be defined by the function $E(A_t, D_t)$:

$$E(A_t, D_t) = p(A_t = \alpha^i | D_t = \alpha^j) \quad (3.3)$$

$$= 1 \quad : i = j \quad (3.4)$$

$$= 0 \quad : i \neq j \quad (3.5)$$

Finally, the prior probabilities must be defined for variables C and T . Again, because it is assumed that training data is unavailable, the uniform prior is applied to T . For C , an additional assumption is made that behaviours are fully observed, that is, elements

¹A one-to-one mapping is not required by the approach and this emission distribution could be replaced with another where appropriate.

are not performed before the agent enters the scene. Under this assumption the prior $P(C = \emptyset) = 1$.

The Bayes Net described so far represents a single behaviour in which the target feature set remains constant. However, there are many occasions where the goal of an agent might change. For instance, the completion of a goal is likely to lead to a new goal being chosen. Similarly, goals may be abandoned or postponed before their completion due to the influence of external factors. These factors may be observable (e.g. a locked door) or unobservable (internal changes to goal-prioritisation), and present interesting challenges.

In previous work researchers have identified goal abandonment by a sudden lack of goal evidence [50]. Others have specifically focused upon interleaved goals, where the set of observed actions can be partitioned into two sets, each supporting a different goal [63]. In this thesis an assumption is made that only one goal is pursued at a time, and thus interleaved goals remain out of scope of this research.

Chapter 4 will discuss how goal changes are detected within the bag-of-features framework, however, the current discussion will focus upon how goal-changes are integrated into the Bayes Network once they have been detected.

A new node I is integrated into the Bayes Network (Figure 3.3) to indicate that a behaviour interruption has occurred. Interruptions influence an agent's goal and thus a dependency is added between nodes I and T . This structure imposes the rule that the target features can only change when an interruption occurs. Similarly, a new dependency has been added to node C to indicate that the currently achieved features must be consistent with the target feature set. This dependency implies that an agent can only perform features that support the target behaviour and ensures that $\forall c^k \in C, c^k \in T$.

In addition to changes in the agent's goal behaviour, interruptions also occur when a false-positive observation has been made. Again, Chapter 4 will discuss how false-positive observations are determined and how they are processed by the inference algorithm. However, to summarise here, a false-positive interruption means the agent's desire is not constrained to $T \setminus C$, and correspondingly, there is a dependency between I and D .

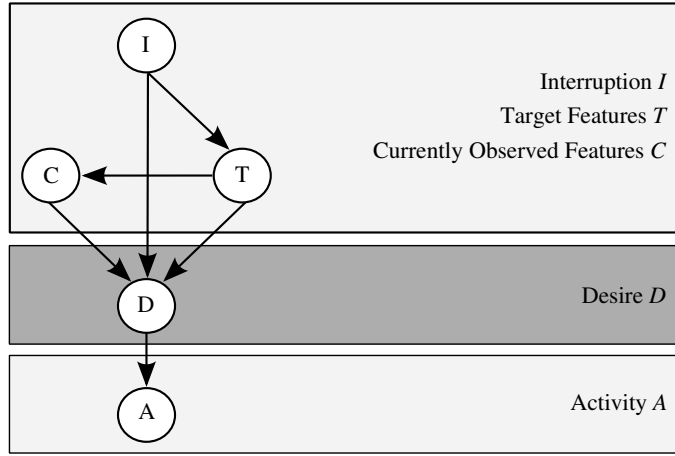


Figure 3.3: The addition of an ‘Interruption’ node I into the bag-of-features DBN

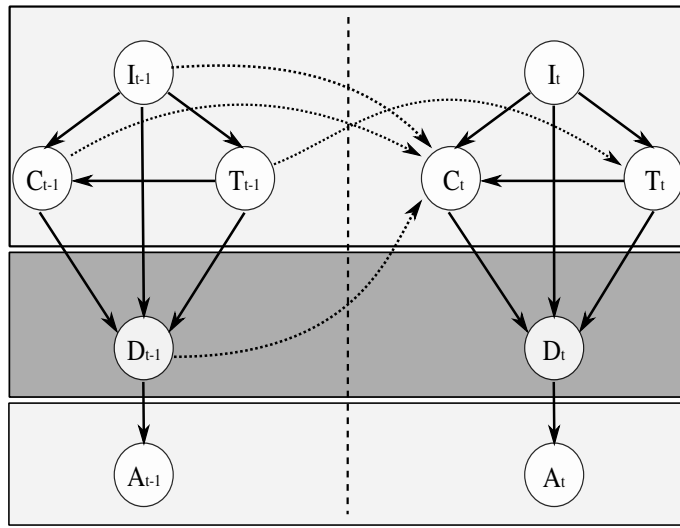


Figure 3.4: The full DBN for ‘bag-of-features’ inference

3.3 Adding A Temporal Model

The Bayes Network representation becomes a temporal model by converting it to a Dynamic Bayesian Network with hidden nodes $\{C, T, D, I\}$ and observable node A . Because the model is discrete the transition model is binary. To fully specify the DBN the prior, conditional and transitional dependencies are described below and formally represented in Table 3.1. When an agent changes their behaviour node $I = TPCh$ (**T**True-**P**ositive **C**hange) and when an activity is misdetected $I = FP$ (**F**alse-**P**ositive). The temporal dependencies encapsulated within the transition model are visually represented in Figure 3.4 via the dashed edges between the time-slices.

1. When there is no interruption ($I_t = TPS$) the target feature set remains the same.
2. When a change of behaviour is detected ($I_t = TPCh$) the target feature set is re-

Table 3.1: Prior, Conditional and Transition probabilities between time steps $t - 1$ and t for the DBN in Figure 3.4

Priors:	$P(C_1 = \emptyset) = 1$	$P(I_1 = TPS) = 1$	$P(T_1 = B^i) = B ^{-1}$
1	$P(T_t = T_{t-1} I_t = TPS) = 1$		Same behaviour
2	$P(T_t = B^i I_t = TPCh) = B ^{-1}$		Behaviour changed
3	$P(T_t = T_{t-1} I_t = FP) = 1$		Mis-detection
4	$P(C_t = C_{t-1} \cup D_{t-1} I_{t-1} = TPS) = 1$		when $D_{t-1} = A_{t-1}$
5	$P(C_t = C_{t-1} \cup D_{t-1} I_{t-1} = TPS) = 0$		when $D_{t-1} \neq A_{t-1}$
6	$P(C_t = \emptyset I_t = TPCh) = 1$		Behaviour changed
7	$P(C_t = C_{t-1} I_{t-1} = FP, I_{t-1} = TPS) = 1$		mis-detection at $t - 1$
8	$P(D_t = \alpha^i C_t, T_t, I_t \neq FP) = C_t \setminus T_t ^{-1}$	$\forall_i : \alpha^i \in C \setminus T$	
9	$P(D_t = \alpha^i C_t, T_t, I_t \neq FP) = 0$	$\forall_i : \alpha^i \ni C \setminus T$	
10	$P(D_t = \alpha^i C_t, T_t, I_t = FP) = \alpha ^{-1}$	$\forall_i : \alpha^i \in \alpha$	
11	$P(A_t = \alpha^i D_t = \alpha^i) = 1$		

initialised according to the uniform distribution

3. When a mis-detection occurs, the target feature set remains the same.
4. The set of currently observed features (C_t) gains the agent's last desire (D_{t-1}) when it matches the last observation (A_{t-1}) and there was no interruption.
5. The set of currently observed features (C_t) remains the same as C_{t-1} when the last desire (D_{t-1}) does not match the last observation (A_{t-1}) and there was no interruption.
6. When the target set has changed ($I_t = TPCh$) the set of currently observed features becomes the empty set.
7. The set of currently observed features (C_t) remains the same as C_{t-1} when the previous time-step was a mis-detection ($I_{t-1} = FP$).
8. The desire (D_t) has a uniform probability for all elements in $C \setminus T$ when $I \neq FP$.
9. The desire (D_t) has a zero probability for any element not in $C \setminus T$ when $I \neq FP$.
10. The desire (D_t) has uniform probability for all elements in α when $I = FP$.
11. There is a 1-to-1 relationship between desires and activities.

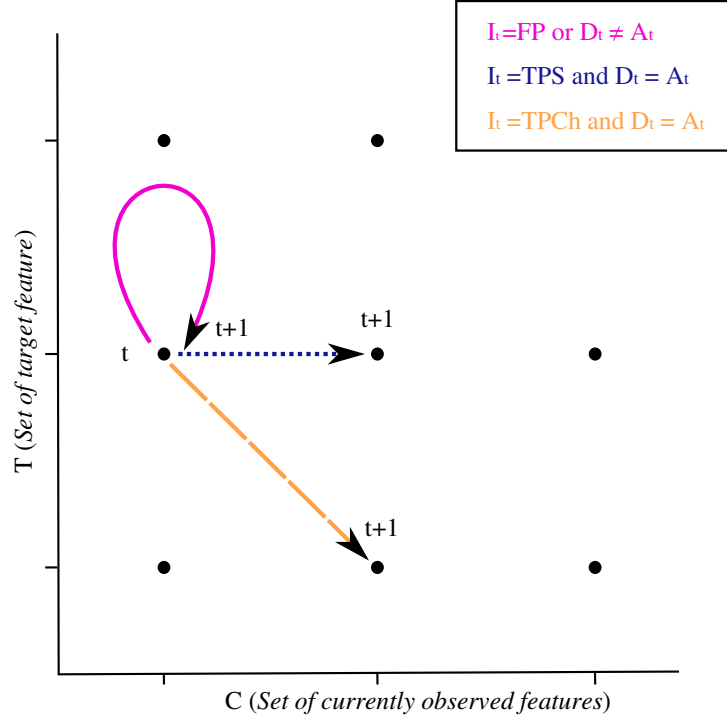


Figure 3.5: A visualisation of the state transition model

To help visualise the temporal model Figure 3.5 shows a two-dimensional representation of the state space. The vertical axis represents different values for T (the set of target feature) while the horizontal axis represents different values of C (the set of currently observed features). When either a *False-Positive* observation is assumed at time t ($I_t = FP$) or the desired feature does not match the observed feature ($D_t \neq A_t$), variables C_t and T_t transition to the same state at $t + 1$. Correspondingly, $C_{t+1} = C_t$ and $T_{t+1} = T_t$, as illustrated by a loop transition to the same state in the figure. When a *True-Positive Same* observation is assumed at time t ($I_t = TPS$) and the desired feature matches the observed feature ($D_t = A_t$), variable C_t will transition to a new state at $t + 1$ to indicate that D_t has been observed ($C_{t+1} = \{C_t \cup D_t\}$), while T_t will remain in the same state because the behaviour has not changed. This is shown in the figure via a horizontal only transition. Finally, when a *True-Positive Change* observation is assumed ($I_t = TPCh$) and the desired feature matches the observed feature ($D_t = A_t$) both C_t and T_t transition to new states at $t + 1$. As before, the horizontal transition represents a change in the state of C_t , although under these conditions $C_{t+1} = D_t$. Similarly, the vertical transition indicates a change in the target behaviour where the value of T_{t+1} is determined by the uniform distribution.

3.4 Summary

This chapter has introduced the fundamental principles of bag-of-features inference which uses a bag to model the individual activities of a behaviour. This bag is considered the agent's target feature set and is represented by node T in the Bayes Network. As the agent performs activities and generates features, we keep track of which elements of T have been observed using the currently observed feature set (node C). The agent's next desire (D in the Bayes Net) is drawn from the set of 'expected features' by conditioning upon T and C . Each element of the 'expected features' set has uniform probability, while all other features have zero probability. When the agent performs their desire they emit a detectable activity, denoted A .

Transitioning from one time-step to the next, the agent's desire becomes an element of C whenever an interruption has not occurred. The target feature set T remains constant throughout each time step unless a True-positive Change occurs, in which case T is reinitialised according to the uniform prior, and C becomes the empty set.

Having defined the Bayesian representation of the approach the next chapter will discuss how efficient inference may be performed on this structure by using Particle Filtering.

Chapter 4

Efficient Inference

Nomenclature:

D_t	Agent's desire at time t	\mathfrak{R}_t	Set of regeneration particles at time t
C_t	The set of currently achieved features at t	M	Number of particle in \mathfrak{R}_t
T_t	The set of target feature at time t	θ_t	Set of re-initialised particles ($\theta_t \subset \mathfrak{R}_t$)
A_t	The observed activity at time t	E	Number of particles in θ_\square
y_t	Generic observation at time t	\mathcal{F}_t	Particles assuming false-positive at time t
$y_{j \rightarrow k}$	y that can be explained by Beh. k but not j	V	Number of particles in \mathcal{F}_t
Y	A set of observations ($y_1, y_2, y_3, \dots, y_T$)	ℓ_t	Set of eligible particles at time t
Z	Generic state encapsulating C, T, D, I	L	Number of particle in ℓ_t
r_t	Sampled components of Z_t	π_t	Updated set of eligible particles at time t
z_t	Marginalised components of Z_t	Q	Number of particle in π_t
S	Number of observations in sequence	G	Number of (goal) behaviours
N	Number of particles	α	Set of detectable features
ξ_t^i	i th sampled particle at t	α^i	The i th element of α
ω_t^i	Weight of particle i at time t	α^{FWD}	A feature
		α^{BWD}	A feature that reverts α^{FWD}

Chapter 2 introduced Dynamic Bayesian Networks (DBNs), which were then used to formally represent bags-of-features in the previous chapter. This representation used a discrete state-space based on features, so exact recursive Bayesian estimation could be used to perform inference. However, prior work has noted that even in discrete cases,

large DBNs cause inference times that are exponential in the number of hidden nodes [33, 38].

Because one of the objectives of this research is to perform real-time behaviour recognition, it is important that inference time can be strictly controlled. Correspondingly, this chapter will focus on approximate recursive Bayesian estimation for bag-of-features inference. The foundations for approximate recursive Bayesian estimation have already been introduced in Chapter 2 with the Sequential Importance Sampling with Re-sampling (SIR) algorithm. The concept of Rao-Blackwellisation was also introduced, which utilises the structure of a model to reduce the number of variables that must be sampled.

Combing Rao-Blackwellisation with the SIR algorithm has produced promising algorithms in previous work (e.g. [24]), and is applied to the bag-of-features representation in the next section (4.1). This basic algorithm communicates the underlying procedure for bag-of-features inference. Once the basic algorithm has been introduced several enhancements will be presented in Section 4.2 leading up to the advanced algorithm. This more complex algorithm will be discussed in relation to three new concepts: partial behaviours, feature repetition and concatenated behaviours.

4.1 Basic Algorithm

Recall that the objective of Rao-Blackwellisation is to reduce the number of components that must be sampled. This is achieved by segmenting the variables into two sets: the sampled components, and the exact components. It is fundamental that the exact components can be easily marginalised out conditioned upon the sampled components, and thus the structure of a DBN must be analysed.

If one looks back at the DBN in Figure 3.4 on page 72 it can be seen that variable D_t is conditionally dependent upon variables $\{C, T, I\}_t$. Recall that variable D_t represents the agent's desire: the next feature to be performed, and is trivial to calculate conditioned upon $\{C, T, I\}_t$, as shown below:

$$P(D_t = \alpha^i | C_t, T_t, I_t \neq FP) = |C_t \setminus T_t|^{-1} \quad \forall_i : \alpha^i \in C \setminus T \quad (4.1)$$

$$P(D_t = \alpha^i | C_t, T_t, I_t \neq FP) = 0 \quad \forall_i : \alpha^i \notin C \setminus T \quad (4.2)$$

$$P(D_t = \alpha^i | C_t, T_t, I_t = FP) = |\alpha|^{-1} \quad \forall_i : \alpha^i \in \alpha \quad (4.3)$$

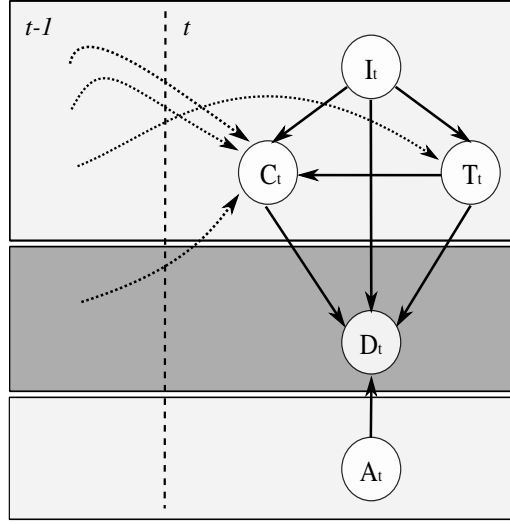


Figure 4.1: Incorporating the latest observation into the RB-Posterior reverses the edge between A_t and D_t

This dependency structure between nodes C, T, I and D is particularly useful for Rao-Blackwellisation, which can be applied by segmenting the state (Bayes Net) into two components. Define $r_t = \{C, T, I\}_t$ and $z_t = \{D\}_t$. Recall that r_t is the sampled component and z_t is the exact component. The Rao-Blackwellised Particle Filter (RBPF) calculates the posterior probability of $P(Z_t|y_{1:t})$ using these two components as follows:

$$P(Z_t|y_{1:t}) = P(y_t|z_t, r_t)P(z_t|r_t, y_{1:t})P(r_t|y_{1:t-1}) \quad (4.4)$$

Section 2.1.8 introduced the Rao-Blackwellised Posterior $P(z_t|r_t, y_{1:t-1})$ and highlighted that better inference could be achieved by incorporating the latest evidence into the posterior. This *evidence reversal* produced the alternate posterior $P(z_t|r_t, y_{1:t})$, which has the effect of reversing the edge between nodes A_t and D_t in the DBN. This is visually represented in Figure 4.1.

Using this new RB-Posterior, particles will only predict desires that are consistent with the latest observation. The RBPF therefore consists of N random samples of the form $\{\{r, z\}_{1:t}^i, \omega_t^i\}_{i=1}^N$ that characterise the posterior density $P(Z_t|y_{1:t})$, where each sample point $\{r, z\}_t^i$ has an associated weight ω_t^i such that $\sum_{i=1}^N \omega_t^i = 1$. The posterior density at time t is approximated as:

$$P(Z_t|y_{1:t}) \approx \sum_{i=1}^N \omega_t^i \delta(\{r, z\}_t, \{r, z\}_t^i) \quad (4.5)$$

$$\approx \sum_{i=1}^N \omega_t^i \delta(\{C, T, I, D\}_t, \{C, T, I, D\}_t^i) \quad (4.6)$$

As shown in Section 2.1.6, the Two-Kernel weight update equation is given by:

$$\omega_t^i \propto \omega_{t-1}^i \frac{Q(\{r, z\}_t, \{r, z\}_{t-1}, y_t)}{K(\{r, z\}_t, \{r, z\}_{t-1}, y_t)} \quad (4.7)$$

$$\propto \omega_{t-1}^i \frac{P(y_t | \{z, r\}_t) P(\{z, r\}_t | \{z, r\}_{t-1})}{P(\{z, r\}_t | \{z, r\}_{t-1}, y_t)} \quad (4.8)$$

Re-writing this in terms of the DBN gives:

$$\omega_t^i \propto \omega_{t-1}^i \frac{P(A_t | D_t^i) P(\{C, T, I, D\}_t^i | \{C, T, I, D\}_{t-1}^i)}{P(\{C, T, I, D\}_t^i | \{C, T, I, D\}_{t-1}^i, A_t)} \quad (4.9)$$

Unfortunately, the dependencies in Equation 4.9 are too complex to calculate accurately without parameter estimation. This is a problem for the approach, as it is assumed that training data is unavailable. However, the underlying principle of particle weighting is that more probable particles obtain higher weights than less probable particles. If a heuristic weight sufficiently encapsulates the dependencies of equation 4.9, model convergence should still occur. This thesis explores this premiss by defining a heuristic weight composed of two factors:

1. $P(Z_t^i | y_t)$: the true positive probability of the observation: $TP(y_t)$
2. The proportion of features in the target feature set that are currently achieved: $\frac{|C_t|}{|T_t|}$

Recall that there is a one-to-one relation between desires and activities, and thus any inconsistency between the two implies that y_t is a false detection. Under this circumstance the $P(Z_t^i | y_t)$ is the false-positive detection rate of y_t . Otherwise, the observation can be assumed to be a true positive detection with probability equal to the true-positive detection rate of y_t . This heuristic component will therefore give a lower weight to particles that are unlikely given the current observation, and higher weights to more likely particles (assuming the true-positive rate is greater than the false-positive rate).

The second factor considers the proportion of an agent's behaviour that a particle can explain. Consider two particles ξ^i and ξ^j , each with the same target feature set. The particle that can explain more observations has more elements in set C , and correspondingly, will obtain a greater factor than one with less elements in C . Therefore, particles that can explain more observations are favoured which is also consistent with the principles of particle weighting: more likely particles should obtain more weight.

Taken together, these factors both support the principles of particle weighting, and can be combined via Equation 4.10. Note that it must be ensured that a particle that cannot explain any observations ($C = \emptyset$) does not receive a zero weight because this is the initial particle state (no features have been observed). Weights of zero are prevented by adding 1 to both the numerator and denominator. An evaluation of this weighting strategy is given in Chapter 10.

$$\omega_t^i = \omega_{t-1}^i \times P(Z_t^i | y_t) \times \frac{|C_t^i| + 1}{|T_t^i| + 1} \quad (4.10)$$

Because the particle weights must sum to one, a normalisation step is required after the initial particle weights are calculated. This process is trivial.

4.1.1 Sample Regeneration

Section 2.1.8 stated that evidence reversal improved inference by incorporating the latest evidence (observation) into the RB-Posterior $P(z_t | r_t, y_{1:t})$. However, a limitation of this approach is that a false-positive detection will affect the distribution. Traditionally, particle filtering assumes that although observations may be noisy, false-positive observations (classification errors) do not occur. However, in a discrete filter such as the one described above, noise takes the form of false-positive detections (classification errors). If the latest observation is a false-positive, there is the potential for the following to occur:

$$P(D_t^i = d^j | C_t^i, T_t^i, I_t^i \neq FP, A_{1:t}^i) = 0 \quad \forall d^j \in C_t^i \setminus T_t^i \quad (4.11)$$

In essence, the sample point cannot explain $A_{1:t}$. Furthermore, there is the potential that no particle can explain $A_{1:t}$, which will lead to particle filter collapse (all particles have zero weight). In order to address false-positive detections one solution would be to add a noise model that predicts false-positive observations. This is similar to how many non-discrete filters operate. However, false-positive detections are not the only possible cause of filter collapse, with two additional causes being:

- The observed feature cannot be explained by any behaviour.

A DBN representing behaviour B^k can only model the states of an agent performing that behaviour. If the set of all behaviours (and the associated set of DBNs) does not contain feature α^i , then $P(D_t = A^i)$ must be zero for all behaviours.

- Sample impoverishment

If $\exists B^k : A_t \in B^k$, but $\neg \exists i : A_t \in C_t^i \setminus T_t^i$ then $P(D_t^i | A_t)$ must be zero for all i where i represents particle index.

The addition of a noise model cannot address these two additional causes of filter collapse, and thus a combined solution is provided via sample regeneration.

This is achieved by inserting an additional step into the RBPF algorithm to identify particles that will collapse during the next iteration. This can be achieved by identifying where $P(y_t | Z_t, y_{1:t-1}) = 0$, or more specifically, evaluating $P(A_t^i | \{C, T, I\}_t^i, A_{1:t-1}^i) \forall i$. The particles identified form the regeneration set \mathfrak{R} , while the remaining particles (where $P(y_t | Z_t, y_{1:t-1}) \neq 0$) form the eligible set ℓ , where $\{\mathfrak{R}_t \cup \ell_t\} = \{Z_t^i\}_{i=1}^N$ and $\{\mathfrak{R}_t \cap \ell_t\} = \emptyset$. The regeneration set encapsulates all particles that cannot explain the current observation. Note that when A_t cannot be explained by any behaviour (i.e. $A_t \not\subseteq \{B^k\}_{k=1}^G$) all particles will be contained within \mathfrak{R} .

A noise model can now be applied to \mathfrak{R} by sampling particles (without replacement) according to the false-positive probability of A_t , i.e. $1 - TP(A_t)$, where it is assumed the true positive rate of feature A_t is known. It should also be noted that the detection probability could alternatively be used if available. These sampled particles form the set of ‘false positive’ particles \mathcal{F} . Because $P(y_t | Z_t, y_{1:t-1}) = 0$ the particles in \mathcal{F} are given a weight λ , which replaces the heuristic in Equation 4.10, where λ is some small factor. A value of 0.01 was arbitrarily chosen and shown to work well during experimentation.

The remaining regeneration particles ($\mathfrak{R} \setminus \mathcal{F}$) represent a true-positive observation. Suppose that $A_t \subset \{B^k\}_{k=1}^G$ and thus sample impoverishment is the cause of all particles in $\mathfrak{R} \setminus \mathcal{F}$. The simplest mechanism to re-generate these particles is to re-initialise them according to the prior $P(Z_1)$ ¹

In section 3.3 the interruption variable was introduced as having an influence on the target feature set and currently observed features. It is in fact regeneration that changes the interruption variable for all particles in \mathfrak{R} .

¹A more efficient mechanism will be introduced in Section 4.2.

Worked Example

To assist understanding and demonstrate the effect of the algorithm consider an example with the following three behaviours:

1. Passing Through 1 (Beh. 1) - This behaviour represents an agent entering and leaving the scene. The target features are $\{EnterAgent, ExitAgent\}$.
2. Passing Through 2 (Beh 2.) - This behaviour represents an agent entering the scene, placing an object on the ground, removing the object again and then exiting the scene. The target features are $\{EnterAgent, PlaceObject, RemoveObject, ExitAgent\}$.
3. Abandon Object 2 (Beh 3.) - This behaviour represents an agent entering the scene, placing a luggage item on the ground and then exiting the scene. The target features are $\{EnterAgent, PlaceObject, ExitAgent\}$.

Assume that we wish to identify which of these behaviours is most likely to have produced a sequence of agent observations. For simplicity, we will focus on only two observations: *EnterAgent* followed by *PlaceObject*.

The first step of the algorithm is to initialise the particle filter, which in this example will involve initialising 600 particles. Recall from Table 3.1 (page 73) that the prior distribution initialises the currently achieved features to empty ($C = \emptyset$) and applies the uniform distribution to the target feature set (T). This can be seen in Figures 4.2 and 4.3, which should be viewed as two sides of one large horizontal figure. In these figures the numbers on the far left relate to the steps of Algorithm 4.1. To differentiate from these, written line numbers in the text will refer to ruled lines in the figure. The first three lines of the figures show the initial number of particles representing each behaviour and their initialised values for C and T .

When the first observation arrives (*EnterAgent*) particles that will collapse are identified by evaluating $P(y_t|Z_t, y_{1:t-1}) = 0$. At the current stage $P(EnterAgent|C, T) \neq 0$ for all particles, so no particles will collapse and variable $I = TPS$ to indicate a true-positive-same behaviour. At this point each particle can now calculate the Rao-Blackwellised (RB) posterior distribution of D by conditioning on variables C, T, I , and A . In this case $P(D = EnterAgent|C, T, I, A) = 1$ because no other element of T can explain A (the observation *EnterAgent*), as shown on the fifth line in the figures. In the next step of the algorithm each particle chooses a value for D according to the RB-Posterior distribution.

Each particle now has values for each variable and can be weighted using Equation 4.10. This is shown on the eighth line in the figures, which shows the weight that each particle will obtain. Note that the true positive rate for the *EnterAgent* feature is assumed to be 0.956 in this example and is taken from the performance of our real detector (introduced in Chapter 7 and used in our experiments). It is important to identify that particles representing Behaviour 1 obtain the highest weights, followed by Behaviour 3 and then Behaviour 2. This is because Behaviour 1 has the fewest elements in T (this will be discussed further later in the example). On the ninth line the N particle weights are normalised so that $\sum_{i=1}^N \omega_i^j = 1$.

At the next iteration of the algorithm (starting on the tenth line) the first step is to resample the particles with replacement. Upon completion the particles will be distributed according to the probability density estimate from line nine, and correspondingly, this will lead to a redistribution of particles such that 43% represent behaviour 1, 32% behaviour 3 and 25% behaviour 2. The tenth line shows an approximation of the associated numbers of particles.

As before, the algorithm then identifies particles that are about to collapse. This time, $P(\text{PlaceObject}|C, T) = 0$ for all particles representing behaviour 1 and they become members of regeneration set \mathfrak{R} . Of these particles, a proportion equal to the false-positive rate of the observation are selected to assume a false-positive hypothesis. In this example, *PlaceObject* has a true-positive detection rate of 0.73, and correspondingly, 27% of \mathfrak{R} (70 particles) are selected for the false-positive hypothesis. This is shown on line 13 of Figure 4.2. These particles are weighted with $\lambda = 0.01$ (line fourteen).

Recall that the remaining particles in \mathfrak{R} (≈ 188 for Behaviour 1) are reinitialised according to the priors. This means that their target feature set (T) variables are uniformly distributed between the behaviours, and currently achieved feature sets are emptied ($C = \emptyset$). Note that this operation was not performed for the particles making a false-positive assumption, which have still achieved the *EnterAgent* feature. At this point it is also logical to highlight why Figures 4.2 and 4.3 are separated as they are. Note that the Behaviour 1 particles that have been re-initialised into Behaviour 2 and 3 particles have achieved fewer features than their neighbours in Figure 4.3. This mean that these particles will obtain different weights and thus must be illustrated separately.

Having dealt with all particles in \mathfrak{R} the algorithm proceeds by calculating the RB-Posterior. Note that it does not matter what the distribution is for the false-positive particles because they have a fixed weight. For the ‘true-positive’ behaviour 1 particles $P(D = \alpha_i) = 0 \forall \alpha_i \in \alpha$, so in essence these particles have a zero distribution and cannot make a prediction. The

behaviour 2 and 3 particles can all explain the observation via $D = PlaceObject$, and thus this instantiation has probability = 1.

As before, the particles are weighted and note that the ‘true-positive’ behaviour 1 particles attain a weight of zero because they cannot explain the observation. In essence these 62 particles collapse and will never be resampled. Also note that the behaviour 2 and 3 particles in Figure 4.2 attain lower weights than their neighbours in Figure 4.3 due to the $\frac{|C|+1}{|T|+1}$ component of the weight formula. As in the previous iteration, the N particle weights are normalised so that $\sum_{i=1}^N \omega_t^i = 1$, as shown on the last line of iteration 2. Finally, the total weight attributed to each set of particles is shown.

In Figure 4.3 the same process is performed for the particles representing behaviours 2 and 3. The last line of Figure 4.3 shows the total weight for each behaviour hypothesis, calculated by summing the weights of the particles representing the same target behaviour. This allows us to estimate the probability of a target behaviour B^k as $P(T_2 = B^k | A_{1:2}) = \sum_{i=1}^N \delta(T_t^i, B^k)$. This is shown at the bottom of Figure 4.3 where one can see that $P(T = Beh1 | A_{1:2}) = 0.005$, $P(T = Beh2 | A_{1:2}) = 0.392$, and $P(T = Beh3 | A_{1:2}) = 0.603$. Given that behaviour 1 cannot explain all of the observations one would expect this behaviour to have the lowest probability. Similarly, more features for Behaviour 3 than Behaviour 2 have been observed, and thus Behaviour 3 is currently the most likely explanation for the observations.

This aspect of the approach deserves a little further discussion, as it is in contrast to more common approaches such as the Hidden Markov Model where one would expect compatible models to have the same probability. The bag-of-features approach is referred to as a ‘minimal explanation’ technique because it gives the highest probability to the shortest behaviour that can explain the set of observations. This can be observed in Figure 4.4 which shows the probability of behaviours as a sequence of observations arrives. This figure extends the previous example, although the Figure also includes probabilities for four additional behaviours used within our evaluation.



Consider the probabilities of Passing Through 1 (PT1), Passing Through 2 (PT2) and Abandon Object 2 (AO2) after the first observation. In an HMM approach with uniform priors all three behaviours would have a similar probability, which is in stark contrast to the bag-of-features results. Although at first this might seem like a flaw in the approach, this is not an undesirable effect. Assume that after one observation no further observations are made. If *PT1* were being observed, this indicates that one feature was missed, while two features must have been missed for *AO2* and three for *PT2*. If it is assumed that features can be reliably detected, it is true that *PT1* should be more probable.

Algorithm 4.1 The Rao-Blackwellised Particle Filter algorithm

```

1: Init: Generate  $[\{C, T\}_1^i, \omega_1^i]_{i=1}^N \sim P(Z_1)$  and  $\omega_1$ 
2: for  $t = 1$  to  $S$  do {where  $S$  is the length of the observation sequence}
3:   for  $i = 1$  to  $N$  do
4:     Resample  $\{C, T\}_t^i \sim \{\{D, C, T, I\}_{t-1}^i, \omega_{t-1}^i\}$  via Algorithm 2.3 on page 26
5:     Transition with  $P(r_t^i | Z_{t-1}^i)$  to obtain  $\{r\}_t^{i*} = \{C, T\}_t^{i*}$ 
6:   end for
7:   Partition into sets  $\{\mathfrak{R}_t^m\}_{m=1}^M$  (Regeneration) and  $\{\ell_t^l\}_{l=1}^L$  (Eligible)
8:   for  $m = 1 : M$  do
9:     if  $\text{random}() < 1 - TP(A_t)$  then
10:      Flag  $\mathfrak{R}_t^m$  as ‘false positive’ and weight 0.01.
11:      Move  $\mathfrak{R}_t^m$  into set  $\mathcal{F}_t$  where  $\mathcal{F}_t = \{\mathcal{F}_t^v\}_{v=1}^V$ 
12:     else
13:      Reset  $\mathfrak{R}_t^m$  according to the prior and move to set  $\Theta_t$  where  $\Theta_t = \{\Theta_t^e\}_{e=1}^E$ 
14:     end if
15:   end for
16:    $\{\pi_t^q\}_{q=1}^Q = \{\ell_t^l\}_{l=1}^L \cup \{\Theta_t^e\}_{e=1}^E$ 
17:   for  $q = 1 : Q$  do
18:     Calculate RB-Posterior:  $P(D_t^i | \{C, T, I\}_t^i, A_{1:t})$ 
19:     Predict  $\pi_t^q = \{r_t^{q*}, z_t^q\}$  from RB-Posterior giving  $\{D, C, T, I\}_t^q$ 
20:     Weight  $\pi_t^q$  according to Equation 4.10 on page 80
21:   end for
22:    $Z_t = \{\mathcal{F}_t^v\}_{v=1}^V \cup \{\pi_t^q\}_{q=1}^Q$ 
23:   Normalise  $Z_t$ 
24: end for

```

		<u>Behaviour 1</u>		
1.	No. Particles:	200		
	C:	{ }		
	T:	{ EnterAgent ExitAgent }		
3 - 6.	Distribution unaffected because all particles have uniform weight			
7.	 Obs 1: EnterAgent (TP =0.956)	P (D=Ent C, T, I, A) =1		
8 - 15.	M=0 so loop not entered			
19.	Prediction:	D=EnterAgent		
20.	Particle Weights: (C +1)/(T +1)xTP	$\frac{1}{3} \times 0.956 = 0.32$		
23.	Normalised Weights:	0.0021		
<u>End of iteration 1</u>				
3 - 6.	Resample:	43%		
	C:	{ EnterAgent }		
	T:	{ EnterAgent ExitAgent }		
	No. Particles:	258		
7.	 Obs 2: PlaceObject (TP =0.73)	P (A=Plc C, T, I, A) =0		
10 - 11.	(No. Particles:)	258		
	No. Particles Assume FP:	70	188	
13.	No. Particles Reset:			
	No. Particles:	70	62	
	C:	{ EnterAgent }	{ }	
	T:	{ EnterAgent ExitAgent }	{ EnterAgent ExitAgent }	
19.	Prediction:	FP	D={ }	
20.	Particle Weights: (C +1)/(T +1)xTP	0.01	0	
23.	Normalised Weights:	0.000074	0	
<u>End of iteration 2</u>				
	Total Weight:	0.00518 (A)	0 (B)	
		<u>Behaviour 2</u>		<u>Behaviour 3</u>
	No. Particles:	62	62	
	C:	{ }	{ }	
	T:	{ EnterAgent PlaceObject RemoveObject ExitAgent }	{ EnterAgent PlaceObject ExitAgent }	
19.	Prediction:	D=Place	D=Place	
20.	Particle Weights: (C +1)/(T +1)xTP	$\frac{1}{5} \times 0.73 = 0.15$	$\frac{1}{4} \times 0.73 = 0.18$	
23.	Normalised Weights:	0.00108	0.00135	
	Total Weight:	0.06708 (C)	0.08384 (D)	

Legend:
 Beh 1: Passing Through 1
 Beh 2: Passing Through 2
 Beh 3: Abandon Object 2

Cont...

Figure 4.2: A worked example of the basic inference algorithm (Part 1). Please refer to the text for a detailed discussion. Line numbers relate to Algorithm 4.1

Cont...



	Behaviour 1	Behaviour 2	Behaviour 3
1. No. Particles:	200	200	200
C:		{}	{}
T:		$\begin{Bmatrix} \text{EnterAgent} \\ \text{PlaceObject} \\ \text{RemoveObject} \\ \text{ExitAgent} \end{Bmatrix}$	$\begin{Bmatrix} \text{EnterAgent} \\ \text{PlaceObject} \\ \text{ExitAgent} \end{Bmatrix}$
3 - 6.	Distribution unaffected because all particles have uniform weight		
7.  Obs 1: EnterAgent (TP =0.956)		$P(D=\text{Ent} C, T, I, A) = 1$	$P(D=\text{Ent} C, T, I, A) = 1$
8 - 15.	M=0 so loop not entered		
19. Prediction:		D=EnterAgent	D=EnterAgent
20. Particle Weights: $(C +1)/(T +1) \times \text{TP}$		$\frac{1}{5} \times 0.956 = 0.19$	$\frac{1}{4} \times 0.956 = 0.24$
23. Normalised Weights:		0.0013	0.0016
	End of iteration 1		
3 - 6. Resample:		25%	32%
C:		{EnterAgent}	{EnterAgent}
T:		$\begin{Bmatrix} \text{EnterAgent} \\ \text{PlaceObject} \\ \text{RemoveObject} \\ \text{ExitAgent} \end{Bmatrix}$	$\begin{Bmatrix} \text{EnterAgent} \\ \text{PlaceObject} \\ \text{ExitAgent} \end{Bmatrix}$
No. Particles:		150	192
7 - 15.			
 Obs 2: PlaceObject (TP =0.73)	$P(D=\text{Plc} C, T, I, A) = 0$	$P(D=\text{Plc} C, T, I, A) = 1$	$P(D=\text{Plc} C, T, I, A) = 1$
No. Particles Assume FP:	<i>Beh. 1 cannot explain next observation so 27% Beh. 1 particles assume a false-positive obs. Remaining 73% are reset and distributed uniformly. Beh. 2&3 can both explain next obs.</i>		
No. Particles Reset:			
No. Particles:			
19. Prediction:		D=Place	D=Place
20. Particle Weights: $(C +1)/(T +1) \times \text{TP}$		$\frac{2}{5} \times 0.73 = 0.29$	$\frac{2}{4} \times 0.73 = 0.37$
23. Normalised Weights:		0.00216	0.0027
	End of iteration 2		
Total Weight:	A	0.325 + C	0.519 + D
P(T Obs):	0.005	=0.325+0.06708 0.392	=0.519+0.08384 0.603

Figure 4.3: A worked example of the basic inference algorithm (Part 2). Please refer to the text for a detailed discussion. Line numbers relate to Algorithm 4.1

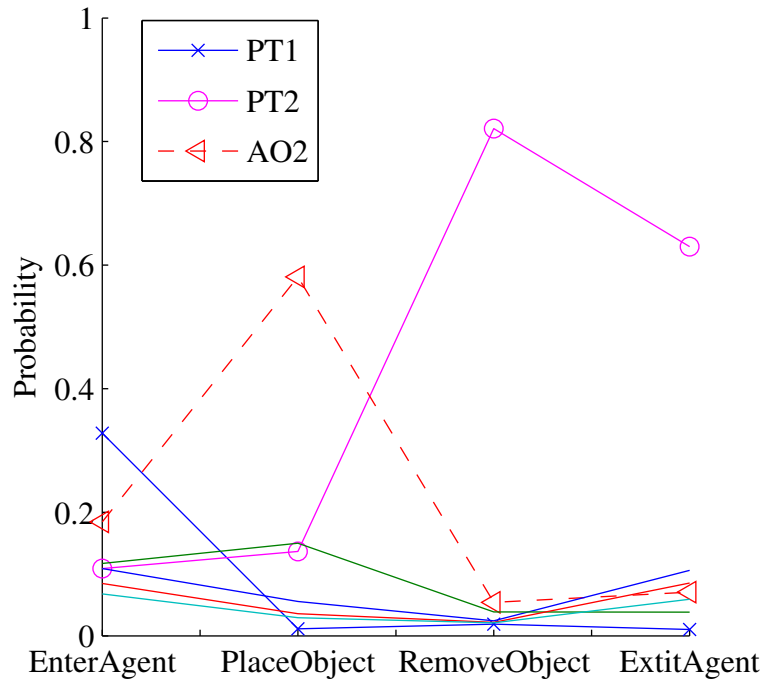


Figure 4.4: The minimal explanation changes as observations arrive. At the first observation (*EnterAgent*) *PT1* (Passing Through 1) gains the highest probability because it is the shortest explaining behaviour. By the second observation (*PlaceObject*) *PT1* can no longer explain the observations and reduces in probability, while *AO2* (Abandon Object 2) becomes more likely as the next shortest behaviour that can explain the observations. This happens again at observation three where *PT2* (Passing Through 2) becomes and remains the most probable explanation.

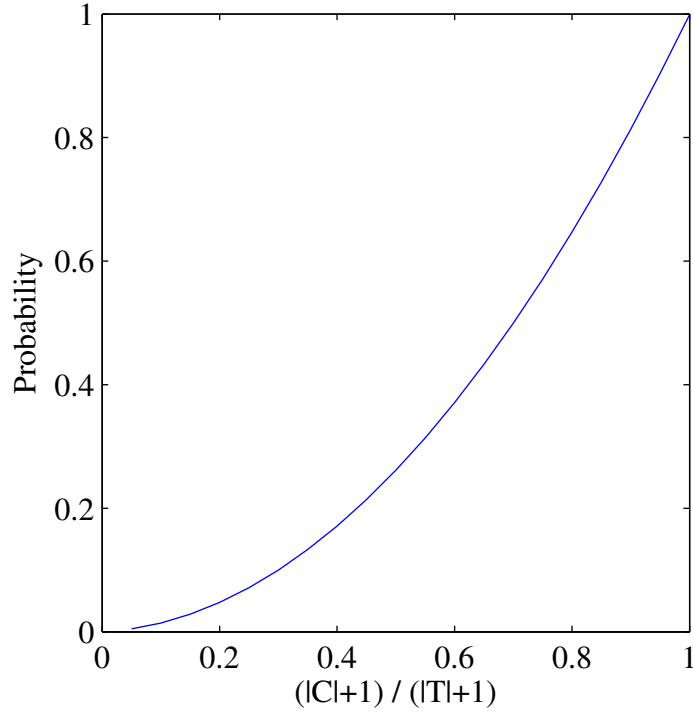


Figure 4.5: The cumulative distribution function used by Algorithm 4.1.

4.1.2 Summary

To draw this section together, the simple RBPF based algorithm is presented in Algorithm 4.1. After initialisation of C and T according to their priors the core algorithm is repeated for each observation time-step. The first stage of the algorithm re-samples the particles as discussed in Section 2.1.7. The re-sampling process itself involves the construction of a cumulative distribution function (CDF). If one considers the ratio $\frac{|C|+1}{|T|+1}$ as a random variable representing the proportion of a behaviour that has been performed then Figure 4.5 can be used to graphically represent such a CDF. In this figure the CDF represents a generic set of particles where it is assumed that each particle has a uniform weight from the previous time step and can explain the current feature. In reality this would not be the case, however it is much more difficult to visually represent more complex scenarios. Moving along the CDF particles with higher $\frac{|C|+1}{|T|+1}$ ratios have a higher probability of selection, and thus particles that can explain more observations will be selected more often.

Once re-sampled, the transition kernel is applied to the particles. Because the transition kernel is binary a particle will gain its last predicted desire such that $C_t^{i*} = \{C_t^i \cup D_{t-1}^i\}$ whenever a particle did not assume a false positive observation at the previous time-step. Note that one may be mistaken for thinking that a binary particle transition indicates that sampling is unnecessary, especially given that the state space itself is discrete. However,

it must be remembered that while particles either transition to the next state or die (have a weight of 0), the hypotheses that these particles represent may be different. For example, Figure 4.6 shows two particles representing two behaviours and an observation sequence of three features. In this example the first observed feature (a) can only be explained by behaviour $T1$, and thus any particle representing $T2$ must either assume a false-positive observation, assume a true-positive-change to $T1$ or a true-positive-change to $T2$. However, this last transition still cannot explain feature a and will ultimately gain a weight of zero and die (will never be re-sampled). A similar process occurs at observation two when feature a is again observed, and again at observation 3. It is important to note that in the final time-slice each particle represents a unique hypothesis, including those with the same final state. Thus, one can see that even for this relatively short observation sequence and only two behaviours the particle state-space grows quickly and thus sampling is required if efficient (real-time) inference is to be achieved.

The next step partitions the particles into the Eligible (ℓ) and Regeneration (\Re) sets according to $P(y_t|Z_t^i, y_{1:t-1}) = 0$ or $\neq 0$. The Regeneration set is processed first, with particles being flagged as ‘false positive’ observations according to the false-positive rate of the observation. The remaining elements of the Regeneration set have their state re-initialised according to the prior and are appended to the Eligible set.

For particles in the Eligible set, the Rao-Blackwellised posterior calculates the distribution of variable D_t^i and then selects a value according to that distribution. At this point a particle represents a full sample of (C, T, D, I) and can be weighted using Equation 4.10. Finally, the two sets of particles are joined and the normalised particle weights are calculated so that $\sum_{i=1}^N \omega^i = 1$.

4.2 Advanced Algorithm

The previous section outlined a basic algorithm for performing inference with the bag-of-features approach. This section now considers three methods for improving algorithm efficiency during particle regeneration.

1. Recall that Section 4.1.1 suggested that particles could be regenerated using the following priors:
 - Re-initialising T (the target feature set) according to the uniform distribution
 - Setting $C = \emptyset$

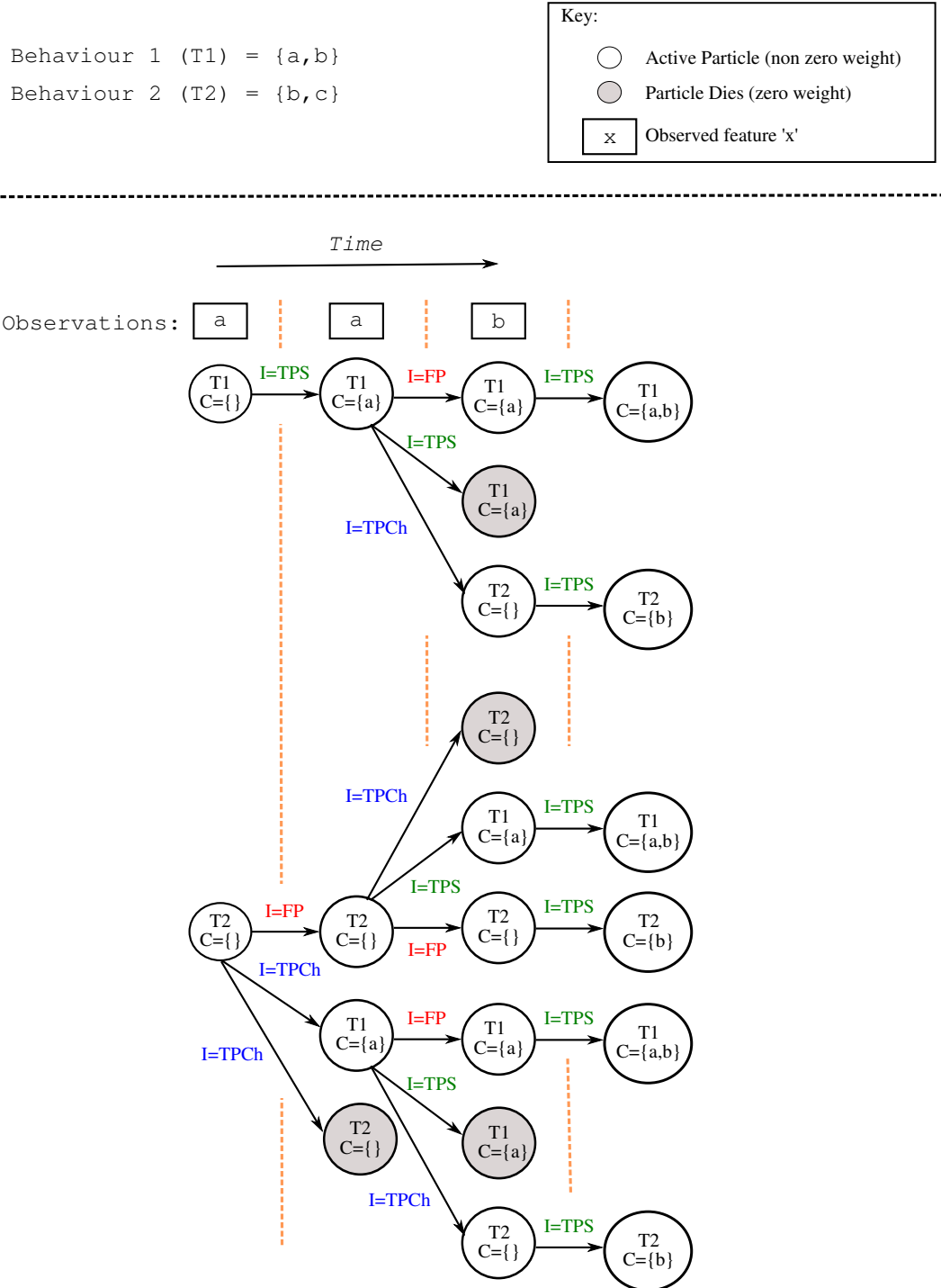


Figure 4.6: Example particle state growth for two particles and two behaviours while observing a 3 features sequence.

This method of regeneration has the effect of erasing the history of the agent, making an assumption that a new goal (behaviour) is being performed and no components of it have been observed. However, such an assumption is naive and fails to consider that some components of the second behaviour may have already been observed during the first behaviour. In other words, the observation stream might be a partial union of two individual behaviours.

2. Furthermore, in some circumstances the activities of an agent might be considered as repeatable. For example, consider a physical agent in a scene. Exiting the scene means the agent is no longer present, and it is therefore permissible to observe them re-enter. This kind of repeatable behaviour is not possible within the algorithm described thus far due to the single occurrence assumption, although an alteration can be made to allow some repetition.
3. Finally, a further enhancement can be made to more efficiently deal with concatenated behaviours. Like partial behaviours, concatenation also causes a stream of observations from two behaviours, with the key difference being that both behaviours are performed in their entirety. In this section all three of these scenarios will be addressed via three small enhancements.

4.2.1 Partial Behaviours

Partial behaviours are typically observed when an agent abandons one goal to perform another. Note that it is assumed that both goals are performed within the view of the sensors and thus the set of observations Y for a single agent can be considered $Y \subset B^j \cup B^k$, where B^j is the first behaviour and B^k is the second.

Denote $y_{j \rightarrow k}$ as the first observed feature from B^k that cannot also be explained by B^j . That is, $y_{j \rightarrow k} \in B^k \wedge y_{j \rightarrow k} \notin B^j$. Furthermore, recall that \mathfrak{R} is the set of particles that cannot explain the current observation, as will occur when $y_{j \rightarrow k}$ is observed. When $y_{j \rightarrow k}$ is observed it is expected that a large number of particles will be present in $\mathfrak{R} \setminus \mathcal{F}$ (recall that \mathcal{F} is the subset selected to represent false positive detections).

If particles are regenerated using the previously suggested scheme (i.e $C^i = \emptyset \forall i \in \mathfrak{R} \setminus \mathcal{F}$) elements of $C^i \in B^k$ will be removed. For example, assume that the *PlaceObject* feature is an element of both behaviours and that it has been observed before $y_{j \rightarrow k}$. When a particle is regenerated it loses the knowledge that *PlaceObject* has been observed.

This loss of knowledge in essence throws away information and will ultimately lead to suboptimal inference. To prevent this occurring the updated Algorithm 4.2 can be applied to only remove elements from C^i when they are not consistent with the new target behaviour T^i . The new algorithm thus keeps knowledge of all previously observed features that are relevant to the new behaviour B^k , and removes only those that are not.

4.2.2 Feature Repetition

Repeatable features are sequences of features that revert the state of the agent. For example, the features *PlaceObject* and *RemoveObject* can be considered repeatable when the object in question is the same. This poses the question, if *RemoveObject* is observed, should it be added to C^i , or should *PlaceObject* be removed from C^i ?

The answer to this question largely depends upon the application area, or more specifically, whether the behaviours being modelled include revertible features. If one needs to model a behaviour that includes a pair of revertible features then it will be necessary for both features to be added to C^i when they are observed. In the opposite case, where revertible features do not need to be modelled, it may be safe to take the alternate approach of removing the ‘reverted’ feature from C^i . The remainder of the discussion will focus on the more complicated scenario where the application needs to model revertible features.

The limitation of adding both features to C^i is that they cannot subsequently be repeated. This is because of the ‘single-occurrence-assumption’ that states that each behaviour is defined by a set of features without repeats. Yet it has just been acknowledged that placing and removing an object reverts the state of the agent, and one could argue that the agent should be permitted to place the object once more (or indeed, place and remove the object any number of times).

Consider the two generic features α^{FWD} and α^{BWD} , where α^{BWD} reverts α^{FWD} . Furthermore, let us assume that a behaviour B^k is defined such that: $\{\alpha^{FWD}, \alpha^{BWD}\} \subset B^k$, and that there exists a particle i such that $C_t^i = \{\alpha^{FWD}, \alpha^{BWD}\}$ and $T_t^i = B^k$. This particle represents the scenario in which a pair of revertible features have both been observed.

Because of the single occurrence assumption, $P(D_t^i = \alpha^{FWD}) = 0$. If the agent causes feature α^{FWD} to be observed again, or in other words, $A_t = \alpha^{FWD}$, then $P(D_t^i = \alpha^{FWD} | \{C, T, I\}_t^i, A_{1:t}) = 0$ and particle i will be regenerated. Thus there is the potential for revertible activities to be detected and processed during the regeneration step

of the algorithm.

Define $RevertedFeatures(\alpha^i)$ as the function that returns the features reverted by α^i . Following the example above, $RevertedFeatures(RemoveObject) = \{PlaceObject\}$. Correspondingly, the regeneration algorithm can be updated to regenerate $C_t^i : C_t^i = C_t^i \setminus \{RevertedFeatures(\alpha^i) \cup \alpha^i\}$, or in other words, to remove any sets of revertible features that exist in C_t^i .

By performing such an operation $P(D_t^i = \alpha^{FWD} | \{C, T, I\}_t^i, A_{1:t})$ will become non-zero, in effect allowing repeatable features². This updated approach can be summarised by Algorithm 4.3.

4.2.3 Behaviour Concatenation

Concatenation occurs when an agent performs one behaviour followed by another. Note that this is different from partial behaviours in that it is assumed that the preceding behaviour is performed in its entirety before the second behaviour begins. Concatenation is the simplest algorithm enhancement as it simply requires a check for $C_t^i = T_t^i$ during particle regeneration. If this condition is true T_t^i should be reset according to the prior as usual, but additionally, C_t^i should also be emptied to indicate no features from the new behaviour have been observed.

Having now defined these new algorithmic concepts this chapter is drawn to a close by summarising them in Advanced Algorithm 4.4.

4.3 Summary

This chapter has introduced two generic inference algorithm for feature-based behaviour recognition. At the beginning of the chapter the Basic Algorithm was described as a combination of the Rao-Blackwellised Particle Filter with the bag-of-features Dynamic Bayesian Network, and was summarised in Algorithm 4.1.

²Note that only sets of revertible features may be repeated. A single feature that is not reverted by another is still subject to the single occurrence assumption.

Algorithm 4.2 Improving particle reset to enable behaviour switching

```
1: for  $i = 1$  to  $M$  do {where  $M = |\mathfrak{R} \setminus \mathcal{F}|$ }
2:    $C^{i*} = \emptyset$ 
3:   for  $e = 1 : E$  do {where  $E = |C^i|$ }
4:     if  $C_e^i \in T^i$  then
5:        $C^{i*} = C^{i*} \cup \{c_e^i\}$  {Keep the previously observed feature}
6:     end if
7:   end for
8:    $C^i = C^{i*}$ 
9: end for
```

Algorithm 4.3 Particle ‘Reset and Revert’ algorithm

```
1: for  $i = 1$  to  $M$  do {where  $M = |\mathfrak{R} \setminus \mathcal{F}|$ }
2:    $C^{i*} = \emptyset$ 
3:   for  $e = 1 : E$  do {where  $E = |C^i|$ }
4:     if  $C_e^i \in T^i$  then
5:        $C^{i*} = C^{i*} \cup \{c_e^i\}$ 
6:     end if
7:   end for
8:   for  $e = 1 : E$  do {where  $E = |C^i|$ }
9:     if  $(c_e^i \in C^{i*} \wedge \text{RevertedFeatures}(c_e^i) \neq \emptyset)$  then
10:      {Remove any feature  $c_e^i$  reverts, and then  $c_e^i$ }
11:      for  $r \in \text{RevertedFeatures}(c_e^i)$  do
12:         $C^{i*} = C^{i*} \setminus r$ 
13:      end for
14:       $C^{i*} = C^{i*} \setminus c_e^i$ 
15:    end if
16:  end for
17:   $C^i = C^{i*}$ 
18: end for
```

Algorithm 4.4 The Advanced Rao-Blackwellised Particle Filter algorithm

```
1: Generate  $[\{C, T\}_0^i, \omega_0^i]_{i=1}^N \sim P(Z_0)$  and  $\omega_0$ 
2: for  $t = 1$  to  $S$  do {where  $S$  is the length of the observation sequence}
3:   for  $i = 1 : N$  do
4:     Resample  $\{C, T\}_t^i \sim \{\{D, C, T, I\}_{t-1}^i, \omega_{t-1}^i\}$  via Algorithm 2.3
5:     Transition with  $P(r_t^i | Z_{t-1}^i)$  to obtain  $\{r\}_t^{i*} = \{C, T\}_t^{i*}$ 
6:   end for
7:   Partition into sets  $\{\mathfrak{R}_t^m\}_{m=1}^M$  (Regeneration) and  $\{\ell_t^l\}_{l=1}^L$  (Eligible)
8:   for  $m = 1 : M$  do
9:     if  $\text{random}() < 1 - TP(A_t)$  then
10:      Flag  $\mathfrak{R}_t^m$  as ‘false positive’ and weight 0.01.
11:      Move  $\mathfrak{R}_t^m$  into set  $\mathcal{F}_t$  [where  $\mathcal{F}_t = \{\mathcal{F}_t^v\}_{v=1}^V$ ]
12:     else
13:       if  $C_t^m = T_t^m$  then
14:         Reset  $\mathfrak{R}_t^m$  according to the prior and move to set  $\Theta_t$ 
15:       else
16:         Reset  $\mathfrak{R}_t^m$  via Algorithm 4.3 and move to set  $\Theta_t$  [where  $\Theta_t = \{\Theta_t^e\}_{e=1}^E$ ]
17:       end if
18:     end if
19:   end for
20: end for
21:  $\{\pi_t^q\}_{q=1}^Q = \{\ell_t^l\}_{l=1}^L \cup \{\theta_t^e\}_{e=1}^E$ 
22: for  $q = 1 : Q$  do
23:   Calculate RB-Posterior:  $P(D_t^i | \{C, T, I\}_t^i, A_{1:t})$ 
24:   Predict  $\pi_t^q = \{r_t^{q*}, z_t^q\}$  from RB-Posterior giving  $\{D, C, T, I\}_t^q$ 
25:   Weight  $\pi_t^q$  according to Equation 4.10 on page 80
26: end for
27:  $Z_t = \{\mathcal{F}_t^v\}_{v=1}^V \cup \{\pi_t^q\}_{q=1}^Q$ 
28: Normalise  $Z_t$ 
```

The second half of the chapter presented some more complex behaviour concepts: partial behaviours, concatenated behaviour, and repeatable features. The basic algorithm is limited in its ability to deal with these concepts, but can be easily adapted to improve its capabilities. Algorithm 4.4 presented the updated inference procedure in pseudo code, and forms the final algorithm in this chapter. Having now described the theory of bag-of-features inference, the next chapter will consider how this theory can be extended to encapsulate behaviour hierarchies. This extension will build upon all of the ideas presented so far and requires very few changes to the algorithms.

Chapter 5

Hierarchical Recognition

Nomenclature:

D_t	Agent's desire at time t	Z^l	Generic state at level l
C_t	The set of currently achieved features at t	α	Set of detectable features
T_t	The set of target feature at time t	α^i	The i th element of α
A_t	The observed activity at time t	α_l	Set of detectable features at level l
y_t	Generic observation at time t	B_l^k	The complex feature k at level l
Z	Generic state encapsulating C, T, D, I	ω_t^i	Weight of particle i at time t
l	Hierarchy level	S	Total number of observations in sequence
\tilde{D}	Hierarchy depth		

In the previous chapters behaviours were defined as sets of features representing activities. In assuming the independence of behavioural features the strict temporal ordering of a behaviour was removed to give the bag-of-features representation.

This representation can be considered flat because it fails to represent the hierarchy often found in behavioural models. However, a more natural representation is to consider the individual sub-goals and activities that the behaviour encapsulates, as shown in Figure 5.1. In such a representation the overall behaviour is defined via a goal behaviour, which is decomposed into individual sub-goals and activities. Correspondingly, the bottom layer in the model is essentially the flat representation found in the earlier chapters.

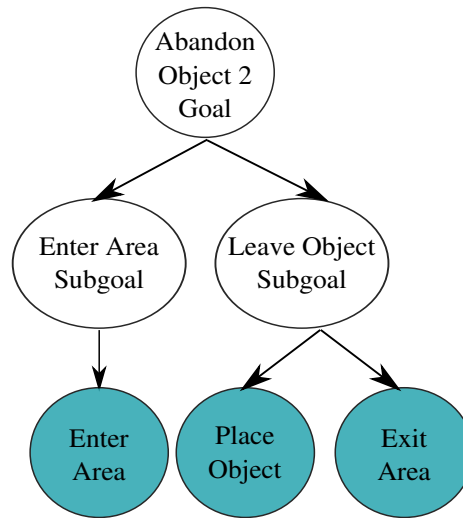


Figure 5.1: An example of how goal behaviours are hierarchically decomposed into sub-goals and activities

Hierarchical representations are not uncommon in prior work, with noted examples including [18, 24, 107, 108]. This wide-spread use is not only due to the ease with which behaviours can be represented hierarchically, but also because it can provide additional inference benefits. For instance, hierarchical composition not only partitions actions into meaningful subsets, but also promotes component re-use.

The merits of component re-use have been documented by Nguyen *et al.* in their work with the Hierarchical Hidden Markov Model [108]. They note that model parameters can be estimated on a component by component basis which reduces the number of parameters for each component. The reduction in state space improves estimation performance and allows more complicated structures to be represented. A further benefit is found when attempting to explain detections to an operator. There are potential benefits to explaining reasoning in terms of sub-goals, especially as the complexity of a behaviour increases to the point where a single (flat) collection of activities would be difficult to communicate.

In this chapter a behaviour hierarchy will be applied to the bag-of-features framework to achieve three objectives:

- Enable multi-agent behaviour to be modelled
- Aid behaviour specification via re-usable components
- Facilitate the explanation of recognised behaviour

This process of hierarchicalisation will be performed in two steps. Firstly, the principles of the bag-of-features approach will be re-defined in terms of a hierarchy, and will then be followed by a conversion of the filtering algorithm to utilise that hierarchy.

5.1 Representation

The underlying principles of feature based recognition remain the same as in Chapter 3. Recall from Chapter 1 that the term *primitive feature* (synonymous with activity) refers to features at the bottom of the behaviour hierarchy, while *complex features* are those that are composed of a number of primitive and/or complex features. To give a concrete example from Figure 5.1, *PlaceObject* is considered a primitive feature, while *LeaveObject* is a complex feature composed of the primitives *PlaceObject* and *ExitArea*. Similarly, *AbandonObject* is a complex feature composed of *EnterArea* and *LeaveObject*.

The DBN structure remains largely the same as before, with the key difference being that variable T , the target feature set, may now contain primitive or complex features, while in the previous chapters it only ever contained primitive features. Define B_l^k as the k 'th behaviour at level l in a hierarchy of depth \tilde{D} , where $l = 1$ at the top. At $l = \tilde{D}$ there are no complex features, and thus there are no target behaviours (and consequently no DBNs). At $l = \tilde{D} - 1$ in Figure 5.1 there are two behaviours: $B_{\tilde{D}-1}^1 = \text{EnterArea}$ and $B_{\tilde{D}-1}^2 = \text{LeaveObject}$. Because this additional notation is cramped the level will only be indicated where essential. When omitted it should be assumed that all variables are at the same level.

First, consider the distribution of D_t when $l = 1$. Recall that the agent's desire D_t is dependent upon the currently observed feature set C_t and the target feature set T_t . Furthermore, recall that the conditional distribution of D_t (when $I_t = TPS$) was defined by:

$$P(D_t = \alpha^i) = P(D_t = \alpha^j) \quad \forall_{i,j} : \alpha^i, \alpha^j \in C_t \setminus T_t \quad (5.1)$$

$$P(D_t = \alpha^k) = 0 \quad \forall_k : \alpha^k \notin C_t \setminus T_t \quad (5.2)$$

To aid with an example, if it is assumed that $C_t = \emptyset$, $T_t = \{\text{EnterArea}, \text{LeaveObject}\}$, and $I_t = TPS$, then:

$$P(D_t = \text{EnterArea} | \{C, T, I\}_t) = P(D_t = \text{LeaveObject} | \{C, T, I\}_t) = 0.5 \quad (5.3)$$

However, it should be clear from Figure 5.1 that the *LeaveObject* feature contains more primitives than *EnterArea* and thus when the temporal order is not modelled, a primitive from *LeaveObject* is more likely to be seen. Consequently $P(D_t = \text{EnterArea} | \{C, T, I\}_t)$ should be less than $P(D_t = \text{LeaveObject} | \{C, T, I\}_t)$, where the ideal posterior is:

$$P(D_t = \text{EnterArea} | \{C, T, I\}_t) = 0.3 \quad (5.4)$$

$$P(D_t = \text{LeaveObject} | \{C, T, I\}_t) = 0.6 \quad (5.5)$$

In other words, the probability of a complex feature is proportional to the number of primitives it generates. Formally, define $ch(\alpha^i)$ as the function that returns the children of α^i , and $numPrim(\alpha^i)$ as a recursive function that returns the number of primitives that can be generated from α^i . Note that the recursive element means that if α^i defines only complex features, those complex features will in-turn be evaluated. Such a function can be summarised in pseudo-code by Algorithm 5.1, and would return $numPrim(\text{AbandonObject}) = 3$.

Next, let us consider C_t . Because there are cases where $ch(\alpha^i) \neq 1$ it is incorrect to consider α^i as complete unless all of its primitives have been performed. To address this C_t must distinguish between complete and incomplete features, and can most easily be achieved by considering an entry in C_t as the chain between complex feature α^i and primitive feature α^j . For instance, $C_t = \{\text{PlaceObject} :: \text{LeaveObject}\}$ would represent the observance of *PlaceObject*, but not *ExitArea*. For simplicity, assume that the target entries in T_t are represented in a similar manner.

To update the distribution of D_t to include the behaviour hierarchy, define $prims(\alpha^i)$ as a further recursive function that returns the primitive chains of α^i , and $remain(C_t, \alpha^i)$ as the function that returns $|prims(\alpha^i)| - |prims(\alpha^i) \in C_t|$. As an example:

If	:	$C_t = \{$	$\text{EnterArea} :: \text{EnterArea}, \text{PlaceObject} :: \text{LeaveObject}\}$
		$T_t = \{$	$\text{EnterArea} :: \text{EnterArea}, \text{PlaceObject} :: \text{LeaveObject},$
			$\text{ExitArea} :: \text{LeaveObject}\}$
		$I_t =$	TPS
then	:		$remain(C_t, \text{EnterArea}) = 1 - 1 = 0$
			$remain(C_t, \text{LeaveObject}) = 2 - 1 = 1$

For simplicity, assume that $remain(C_t, T_t)$ returns the sum of remaining primitives for all features in T_t , and thus $remain(C_t, T_t) = 3 - 2 = 1$ in this example.

The distribution on D_t can now be updated so that for each feature α^i , $P(D_t = \alpha^i)$ is proportional to the number of primitives that have yet to be observed normalised by the the total number of primitives. Formally, the updated distribution of D_t becomes:

$$\forall_i \alpha^i \in T_t : P(D_t = \alpha^i) = \frac{\text{remain}(C_t, \alpha^i)}{\text{remain}(C_t, T_t)} \quad (5.6)$$

$$\forall_j \alpha_j \ni T_t : P(D_t = \alpha^j) = 0 \quad (5.7)$$

This can be solidified with the following examples:

If : $C_t = \{\}$ $I_t = TPS$	$T_t = \{$ <i>EnterArea</i> :: <i>EnterArea</i> , <i>PlaceObject</i> :: <i>LeaveObject</i> , <i>ExitArea</i> :: <i>LeaveObject</i> $\}$
then : $P(D_t = \textit{EnterArea})$ $P(D_t = \textit{LeaveObject})$	= $\frac{1}{3} = 0.\dot{3}$ = $\frac{2}{3} = 0.\dot{6}$
If : $C_t = \{\textit{EnterArea} :: \textit{EnterArea}\}$ $I_t = TPS$	$T_t = \{$ <i>EnterArea</i> :: <i>EnterArea</i> , <i>PlaceObject</i> :: <i>LeaveObject</i> , <i>ExitArea</i> :: <i>LeaveObject</i> $\}$
then : $P(D_t = \textit{EnterArea})$ $P(D_t = \textit{LeaveObject})$	= $\frac{0}{2} = 0$ = $\frac{2}{2} = 1$

Having now defined the updates to variables C_t, T_t and D_t it should be highlighted that variable A_t , which represents the observation, has the same form as D_t . That is, when D_t represents a complex feature so too does A_t , and inversely, when D_t represents a primitive feature A_t is primitive (as in the Chapter 3). Furthermore, in the original model the DBN transitioned such that $C_t = \{C_{t-1} \cup D_{t-1}\}$ when $D_{t-1} = A_{t-1}$. This is fundamentally the same in the new model, only D_{t-1} is a complex to primitive chain and the primitive element of D_{t-1} is compared to A_{t-1} .

The only remaining aspect that should be re-iterated is that in Chapter 3 a DBN represented a single behaviour, but now represents a complex feature. To model the behaviour in Figure 5.1 thus requires three DBNs: one to model the goal behaviour (Abandoned Object 2), and two to model the sub-goals *EnterArea* and *LeaveObject*. Taken in conjunction with other goal-behaviours and their associated sub-goals, the set of DBNs can

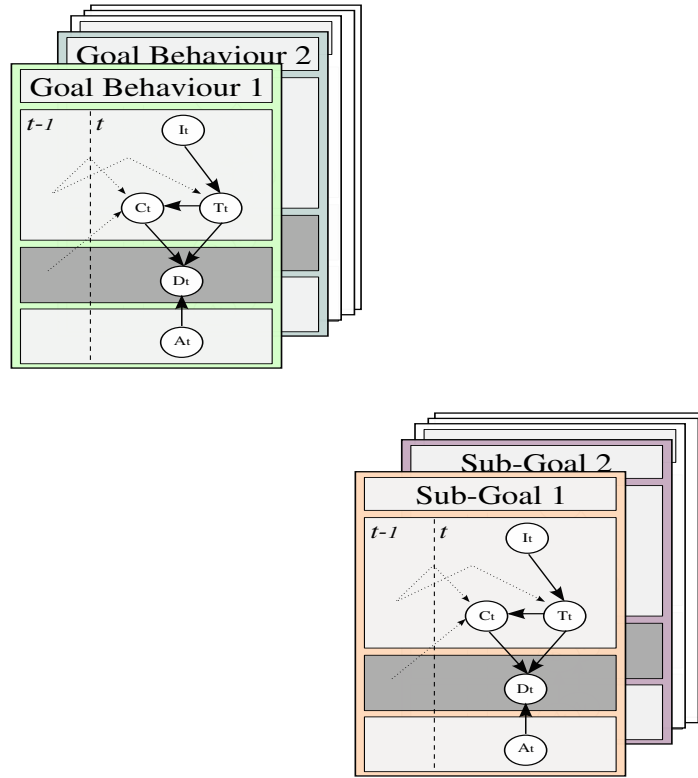


Figure 5.2: To model a set of behaviours requires two stacks of DBNS. One set for the goal behaviours, and one set for the sub-goals.

be graphically represented as in Figure 5.2.

5.2 The Hierarchical Filter

As in Chapter 4 the goal is to calculate the posterior $P(Z_T|y_{1:T})$. However, $Z_{1:T}$ previously consisted of a sequence of states up until time T at a single level, but must now represent the entire behaviour hierarchy. First, consider the joint probability below, where \tilde{D} is the depth of the behaviour hierarchy as in the previous section, and y_t^l is the observation at that layer.

$$P(Z_{1:T}, y_{1:T}) = \prod_{t=1}^T \prod_{l=1}^{\tilde{D}-1} P(y_t^l | Z_t^l) P(Z_t^l | Z_{t-1}^l, y_{1:t-1}^l) \quad (5.8)$$

Assume that once commenced, an agent performs a complex feature (goal/sub-goal) independently from its parent. In essence, this means that each layer of the hierarchy is decoupled and can be recognised independently using the approximate inference algorithms from Chapter 4. To recognise a set of hierarchical behaviours thus

requires a particle filter for each layer forming a collection of $\tilde{D} - 1$ filters approximating $[P(Z_T^l | y_{1:T})]_{l=1}^{\tilde{D}-1}$.

$P(Z_T^l | y_{1:T})$ is the posterior filtering distribution of a complex feature given the set of observations up until time T . To give a concrete example, consider when $l = 2$ in Figure 5.1. The associated filter can be used to estimate $P(\text{LeaveObject} | \{\text{PlaceObject}\})$, or in other words, the probability that sub-goal *LeaveObject* is being pursued given the sequence of observations (in this case there is only one: *PlaceObject*). Recall from Chapter 4 that particles are weighted by the heuristic:

$$\omega_t^i = \omega_{t-1}^i \times P(Z_t^i | y_t) \times \frac{|C_t^i| + 1}{|T_t^i| + 1} \quad (5.9)$$

To define $P(Z_t | y_t)$ in the hierarchical case one must first return to the behaviour decomposition from Section 5.1. Recall that $ch(\alpha_l^i)$ returns the children of complex feature α_l^i , where each child is itself a complex feature when $l < \tilde{D} - 1$. For a set of complex features at level l , a single particle filter approximates $P(Z_T^l = \alpha_l^i | y_{1:T})$, and thus $P(Z_T^l = \alpha_l^i | y_{1:T})$ can be recursively calculated for any level by starting at the bottom of the hierarchy ($\tilde{D} - 1$) where $P(Z_T^{\tilde{D}-1} | y_{1:T}^{\tilde{D}-1}) = TP(y_T)$. At each level $l < \tilde{D} - 1$, $P(Z_T | y_T)$ is approximated via $P(Z_T^{l+1} = \alpha_l^i | y_{1:T})$ and is estimated via the filter at $l + 1$. The over-all filtering procedure can then be performed as per Algorithms 5.2 and 5.3, where the function $stepFilter(f, y, p)$ provides observation y to filter f , and the posterior estimate from the previous level: $P(Z_T^{l+1} = \alpha_l^i | y_{1:T})$.

5.3 Example

To help illustrate Algorithms 5.2 and 5.3 Figures 5.1 on page 98 and 5.3 on page 106 will be used in an example. To generate the particle filters required Algorithm 5.2 recurses down each goal-tree defining a behaviour. Figure 5.1 showed the goal-tree for the *Abandon Object* behaviour and it is assumed that 1 of the k goal-trees is to be recognised. Starting at the root node, the behaviour's target features are defined to be the node's children. In this example the *Abandon Object 2* behaviour has the target feature set $\{\text{EnterArea}, \text{LeaveObject}\}$. Taken into context with the remaining $k - 1$ behaviours a total of k target feature sets are generated. These sets are represented by groups of particles in the Level 1 particles filter (bottom of Figure 5.3). In this particular filter $k = 3$ behaviours and thus there are three distinct groups of particles.

Algorithm 5.1 Recursively evaluate the number of primitive of α^i .

```
1: Prototype:  $[n] = \text{numPrim}(\alpha^i)$ 
2: Init:  $n = 0$ 
3: for  $c_e \in \text{ch}(\alpha^i)$  do
4:   if  $\text{ch}(c_e) = \emptyset$  then  $\{c_e \text{ is primitive}\}$ 
5:     return 1
6:   else
7:     for  $l_f \in \text{ch}(c_e)$  do
8:        $n = n + \text{numPrim}(l_f)$ 
9:     end for
10:   end if
11: end for
```

Algorithm 5.2 Hierarchical Filter Generation

```
1: Prototype:  $\text{generateFilters}(\{G^k\}_{k=1}^K, l, f)$ 
2: Input: Set of  $K$  goal tree root nodes, current level, the set of filters
3: Procedure:
4:  $T = \emptyset$ 
5: for  $k = 1 : K$  do
6:    $\{\text{Recurse down the goal tree}\}$ 
7:    $\text{generateFilters}(\text{ch}(G^k), l + 1, f)$ 
8:    $\{\text{Create a target feature set for this goal root}\}$ 
9:    $T^k = \text{ch}(G^k)$ 
10: end for
11:  $\{\text{Start the goal filter at this level}\}$ 
12:  $f^l = \text{startRaoBlackFilter}(\{T^k\}_{k=1}^K)$ 
```

Algorithm 5.3 The over-all inference procedure

```
1: Prototype:  $\text{bagOfFeatsInf}(\{G^k\}_{k=1}^K, D)$ 
2: Input: Set of  $K$  goal tree root nodes, tree depth  $D$ 
3: Init:
4:  $f = \emptyset$ 
5:  $\text{generateFilters}(\{G^k\}_{k=1}^K, 1, f)$ 
6: for  $t = 1 : S$  do  $\{\text{where } S \text{ is the number of observations in the sequences}\}$ 
7:    $\{\text{Re-initialise the posterior to the true-positive detection rate}\}$ 
8:    $P^D = P(Z_t | y_t) = TP$ 
9:   for  $l = D - 1 : 1$  do
10:     $P^l = \text{stepFilter}(f^l, y_t, P^{l+1})$ 
11:   end for
12: end for
```

Returning to Figure 5.1, Algorithm 5.2 also recurses down to the next level of each behaviour. Continuing the *Abandon Object* example, the *LeaveObject* feature has two children: *PlaceObject* and *ExitArea*. Correspondingly, the *LeaveObject* feature is represented by the target feature set $\{PlaceObject, ExitArea\}$. The other child of *Abandon Object* (*EnterArea*) only has one child: *EnterArea*, and thus its target feature set contains only one element: $\{EnterArea\}$. There are thus two target feature sets at this level which are represented by a Level 2 particle filter. This can be observed at the top of Figure 5.3. Note that this step occurs for each of the k behaviours to obtain k level 2 filters.

The process just described is performed in the initialisation step of Algorithm 5.3. The main iteration loop is then performed, stepping forward the particle filters for each observation. An example of the core particle filter algorithm has already been walked through in Chapter 4, which has relatively few changes in its hierarchical form. The key difference to note is that the level 1 particle filter predicts features that are not directly observable, and for which the true-positive detection rate ($P(Feature = \alpha^i | ObservedFeature = \alpha^i)$) is not directly available. The $P(Feature = \alpha^i | ObservedFeature = \alpha^i)$ can however be approximated via a probability density estimate from the level 2 particle filters. This is represented in Figure 5.3 via directed edges between the level 2 particle filters and the level 1 weighting process. The probability density of the Level 2 particle filter can thus be used to approximate the probability of an agent's sub-goal behaviour, and the level 1 filter approximates their complex goal behaviour.

5.4 Summary

This chapter has gone beyond the flat behaviour representation presented in the previous chapters and extended the approach to contain multiple-levels of abstraction. Observable activities are now referred to as primitive features, while goals and sub-goals are referred to as complex features. Our hierarchical representation uses a separate filter to model behaviour at each level of abstraction and thus a goal-tree of depth three uses two filters. The first filter receives activity observations (primitives) and calculates the posterior density of each sub-goal, the second filter then uses these estimates to calculate the probability of each goal.

Using a hierarchical representation brings a number of benefits. Firstly, at the definition stage behaviours can be represented more naturally and components can be re-used for different behaviours. Furthermore, the framework becomes capable of recognising abstract concepts such as sub-goals, which facilitates the explanation of detections to

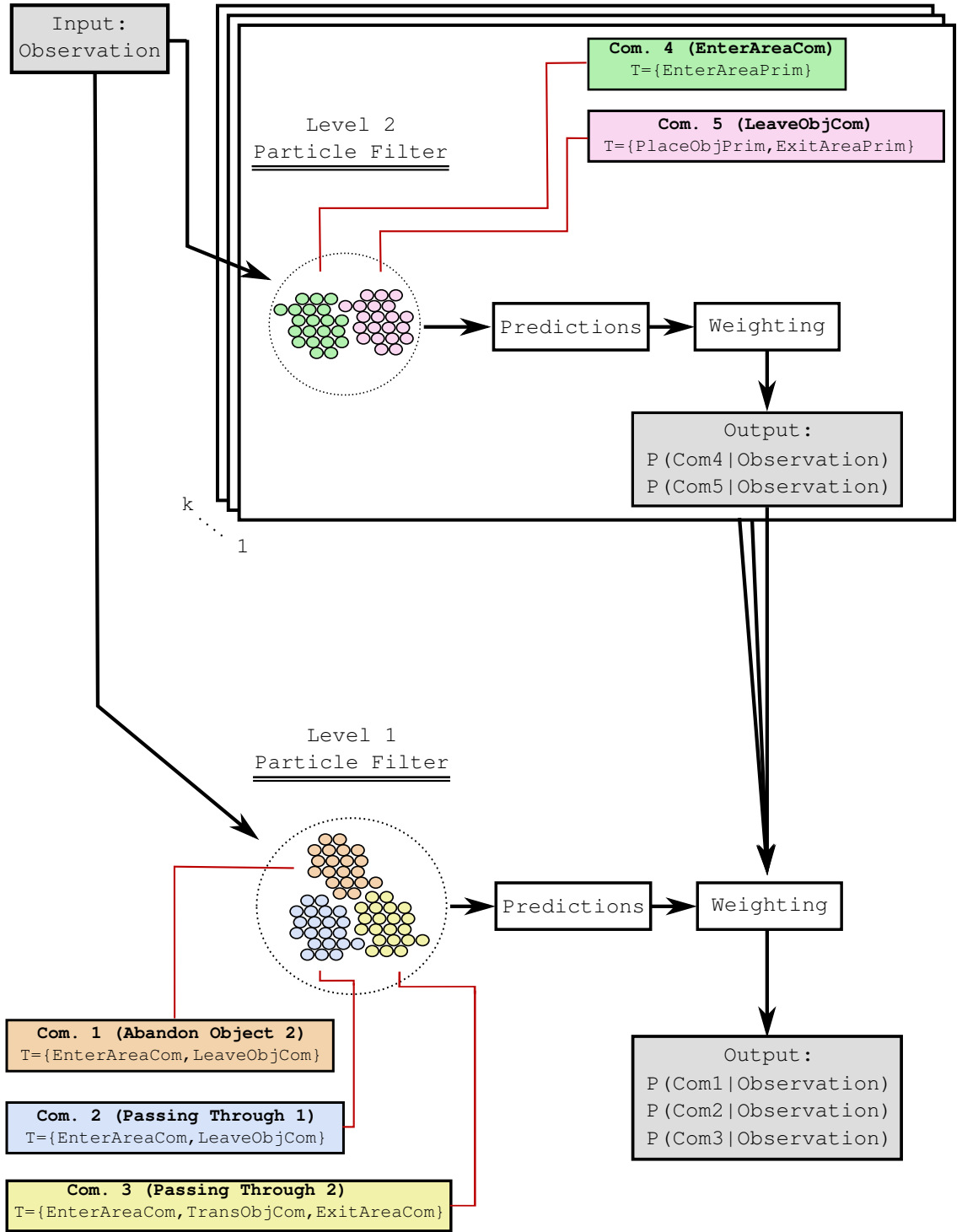


Figure 5.3: Algorithms 5.2 and 5.3 produce two levels of particle filters. The level 1 filter approximates the probability density of the agent's high-level behaviour, while the level 2 filters approximate the probability of different sub-goals. Note that for k high-level behaviours there are also k level 2 filters, but only one level 1 filter. Estimates from the level 2 filters are utilised by the level 1 weighting step.

operators. However, the greatest benefit is that it allows the recognition of multi-agent behaviour and this will become the focus in the next chapter.

Chapter 6

Multi-agent Behaviour Recognition

Nomenclature:

D_t	Agent's desire at time t	α^i	$\langle \text{Feature}, \text{Role} \rangle$ tuple
C_t	The set of currently achieved features at t	$\alpha_{\mathcal{F}}^i$	The feature component of α^i
T_t	The set of target feature at time t	$\alpha_{\mathcal{R}}^i$	The role component of α^i
$\mathfrak{R}1$	Agent 1	f	A Filter
$y_t^{\mathfrak{R}1}$	Observation for agent 1 at time t	$f^{\mathfrak{R}1,2}$	Filter for agents 1&2
$Z_t^{\mathfrak{R}1,2}$	State for multi-agent behaviour between agents 1&2	f_t^i	The i th particle in filter f at time t
S_t	State distribution for solo behaviours	B	The number of behaviours
M_t	State distribution for multi-agent behaviours	B_b^f	The b th behaviour in filter f
γ/ι	Example subgoals	B^s	Set of solo behaviours
N	Number of agents	B^m	Set of multi-agent behaviours
ω^i	Normalised weight for particle i	S	Set of particles representing B^s
ω^{i*}	Un-normalised weight for particle i	M	Set of particles representing B^m
		f_t^B	Filter representing behaviour type B at time t

A benefit of the hierarchical filter is that it can model multi-agent behaviours. This is achieved by assigning agent-roles to complex features with the implication that if sub-goals γ and ι are assigned roles 1 and 2 respectively, all sub-features of γ must be performed by one agent, and all sub-features of ι by another.

Let us consider the *Theft* example in Figure 6.1. In this behaviour the agent fulfilling role

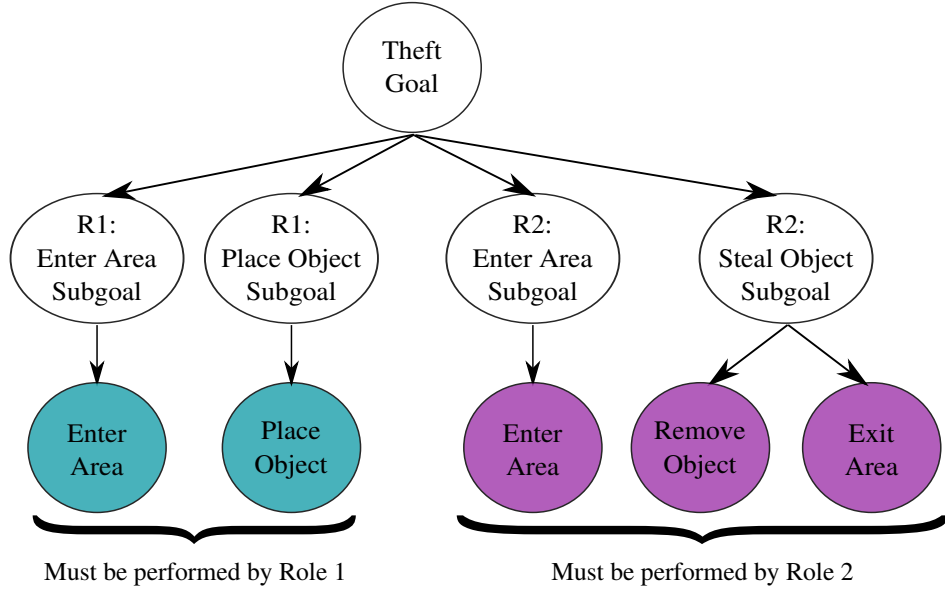


Figure 6.1: The *Theft* behaviour has two distinct agent roles

1 *enters* the scene and *places* an object. The agent fulfilling role 2 also *enters* the scene, and *steals* the object by *removing* it and *exiting*.

By assigning roles to sub-goals there is minimum impact on the bag-of-features representation. Indeed, the only alteration required to the underlying DBN is the addition of role information to variables C_t and T_t and an update to the posterior probability of D_t . This is most easily done by considering α^i as not only a feature, but as a feature/role tuple $\alpha^i = \langle \alpha_{\mathcal{F}}^i, \alpha_{\mathcal{R}}^i \rangle$. The distribution of D_t can then be represented as:

$$\forall_i \alpha^i \in T_t : P(D_t = \alpha^i) = \frac{\text{remain}(C_t, \alpha^i)}{\text{remain}(C_t, T_t, \alpha_{\mathcal{R}}^i)} \quad (6.1)$$

$$\forall_j \alpha_j \ni T_t : P(D_t = \alpha^j) = 0 \quad (6.2)$$

Where function $\text{remain}(C_t, T_t, \alpha_{\mathcal{R}}^i)$ returns the number of primitives for role $\alpha_{\mathcal{R}}^i \in T_t$ less the number of primitives for role $\alpha_{\mathcal{R}}^i \in C_t$, and $\text{remain}(C_t, \alpha^i)$ returns the number of primitives for feature α^i less the number of those primitives in C_t (this is the same as in Section 5.1). In other words, the probability is evenly distributed over the features for role \mathcal{R} that have not yet been observed.

Thus far it has been assumed that the actions of a single agent are being filtered. The addition of multi-agent behaviours complicates the filtering in a number of ways. Firstly, it should be clear that to recognise a multi-agent behaviour through filtering, the observa-

tions of both agents must be provided to the filter. This in itself is not difficult, but raises the important question: What happens when multi-agent and solo-agent behaviours exist?

To see when problems occur consider the following example. Denote a sequence of two observations $\{y_1^{\mathfrak{R}1} = \alpha^i, y_2^{\mathfrak{R}2} = \alpha^j\}$, where subscripts denote order and superscripts denote the agent generating the observation. Suppose that agents 1 and 2 are known to be performing a multi-agent behaviour B^m , but that the filter also contains a solo behaviour B^s . Under these conditions the top-level filter contains a set of $|M|$ particles representing B^m where $\{T_t^i\}_{i=1}^{|M|}$ will contain features for roles 1 and 2. There will also be a set of $|S|$ particles representing B^s where $\{T_t^i\}_{i=1}^{|S|}$ will contain features for role 1 only. For simplicity, assume that both behaviours can explain $\{\alpha_{\mathcal{F}}^i, \alpha_{\mathcal{F}}^j\}$.

When observation $y_1^{\mathfrak{R}1}$ arrives, both sets of particles will be able to explain the observation. However, when $y_2^{\mathfrak{R}2}$ arrives the particle sets will react differently. Because $1 \neq 2$, set M will attempt to find a second role in $\{T_t^i\}_{i=1}^{|M|}$ and will find one. However, when set S attempts to find a second role in $\{T_t^i\}_{i=1}^{|S|}$ it will not find one because they represent a solo behaviour. Consequently, the particles in set S will be regenerated as described in Chapter 4 while the particles in set M will not. Because the particles in M will be able to explain more observations they will gain more weight, resulting in $P(Z_2 = B^m) > P(Z_2 = B^s)$. Recall that it was assumed that the agents were performing a multi-agent behaviour and thus the filter gives the correct result in this scenario.

Now suppose that agents 3 and 4 are not performing multi-agent behaviour B^m , but are instead both performing B^s . Denote the new observations $\{y_1^{\mathfrak{R}3} = \alpha^i, y_2^{\mathfrak{R}4} = \alpha^i, y_3^{\mathfrak{R}3} = \alpha^j\}$. Note that both agents perform feature α^i . As before, when observation $y_1^{\mathfrak{R}3}$ arrives, both sets of particles will be able to explain the observation. But this time, when observation $y_2^{\mathfrak{R}4}$ arrives, neither set of particles will be able to explain the observation. The discussion will focus on set S as the more interesting set in this scenario. This set will have already assigned role 1 to agent 3, and thus the particles must be regenerated if they are to explain $y_2^{\mathfrak{R}4}$. However, this does not solve the problem because the inverse will occur when observation $y_3^{\mathfrak{R}3}$ arrives: role 1 will already be assigned to agent 4.

The solution to this problem is two-fold. Firstly, solo and joint behaviours must be partitioned into separate filters and the results merged to provide an overall posterior. Secondly, each filter must be associated with the agents being filtered and with an observation stream that provides observations of those agents only. In this work it is assumed that observations are associated with a target identifier, and thus the partitioning of observations is trivial. The second example above would require three filters:

- $f^{\mathfrak{R}3}$ Filtering solo behaviour for agent 3
 - ✧ *receiving observations from agent 3 only*
- $f^{\mathfrak{R}4}$ Filtering solo behaviour for agent 4
 - ✧ *receiving observations from agent 4 only*
- $f^{\mathfrak{R}3,4}$ Filtering multi-agent behaviour for agents 3 and 4
 - ✧ *receiving observations from agents 3 and 4*

6.1 Combing Filter Results

Having identified that solo and multi-agent behaviours must be filtered independently some mechanism is required to merge the results of the multi-agent filter M and solo filter S such that:

$$P(Z_t|y_{1:t}) \approx \sum_{i=1}^N \omega^i \delta(Z_t^i, Z_t) \quad (6.3)$$

$$\approx \sum_{m=1}^{|M|} \omega^m \delta(Z_t^m, Z_t) + \sum_{s=1}^{|S|} \omega^s \delta(Z_t^s, Z_t) \quad (6.4)$$

To keep similar notation to before, denote $P(S_t|y_{1:t})$ as the posterior probability of solo behaviours, and $P(M_t|y_{1:t})$ as the posterior for multi-agent behaviours such that:

$$P(S_t|y_{1:t}) \approx \sum_{s=1}^{|S|} \omega^s \delta(S_t^s, S_t) \quad (6.5)$$

$$P(M_t|y_{1:t}) \approx \sum_{m=1}^{|M|} \omega^m \delta(M_t^m, M_t) \quad (6.6)$$

Where as a normalised particle weight is denoted ω^i , let ω^{i*} denote the un-normalised particle weight. Also, if $f \in \{S, M\}$ is a filter representing the solo or multi-agent behaviours, and B^f is the set of associated behaviours, then f_t^i is the i 'th particle in filter f and $\delta(f_t^i, B_b^f) = 1$ when particle i represents the b 'th behaviour in B^f . The **un-normalised** weight of filter f can then be given by FW^f :

$$FW^f = \sum_{b=1}^{|B^f|} FW_b^f = \sum_{b=1}^{|B^f|} \sum_{i=1}^{|f|} \omega^{i*} \delta(f_t^i, B_b^f) \quad (6.7)$$

Where $|f|$ indicates the number of particles in filter f . To merge the posteriors from each filter requires a number of weighted average and re-normalisation steps. Re-normalisation is required because $\sum_{i=1}^N \omega^i = 1$ for each filter, yet the approximation of $P(Z_t|y_{1:t})$ needs to be normalised to be bounded by $[0, 1]$. Furthermore, a weighted average must be calculated to consider the un-normalised weights in each filter.

Step 1: Calculate the relative importance of each filter

The first step is to calculate the relative importance of each filter. For example, if $FW^S = 100$ and $FW^M = 50$ then the importance of each filter can be described as $Imp(FW^S) = \frac{100}{150} = 0.67$ and $Imp(FW^M) = \frac{50}{150} = 0.33$. However, it should be noted that this calculation is only correct if the number of particles in each filter is equal. If one filter has more particles than the other then the filter weights must first be equalised by normalising with the number of particles in each filter ($|f|$):

$$Eq(FW^f) = \frac{\sum_{b=1}^{|B^f|} FW_b^f}{|f|} = \frac{\sum_{b=1}^{|B^f|} \sum_{i=1}^{|f|} \omega^{i*} \delta(f_t^i, B_b^f)}{|f|} \quad (6.8)$$

In the example above suppose that both filters have 100 particles, then, the equalised filter weights will be the same as before:

$$Eq(FW^S) = \frac{100}{100} = 1 \quad (6.9)$$

$$Eq(FW^M) = \frac{50}{100} = 0.5 \quad (6.10)$$

$$\implies \text{Imp}(Eq(FW^S)) = \frac{Eq(FW^S)}{\sum_f Eq(FW^f)} = \frac{1}{1.5} = 0.67 \quad (6.11)$$

$$\implies \text{Imp}(Eq(FW^M)) = \frac{Eq(FW^M)}{\sum_f Eq(FW^f)} = \frac{0.5}{1.5} = 0.33 \quad (6.12)$$

Step 2: Re-normalise each behaviour

The second step is to recompute the likelihood of each filter behaviour B_b^f via re-normalisation. This is achieved by multiplying the filter importance by the original filter posterior:

$$P(Z_t = B_b^f | y_{1:t}) \approx \text{Imp}(Eq(FW^f)) * P(f_t = B_b^f | y_{1:t}) \quad (6.13)$$

To demonstrate the merge process in its entirety, Table 6.1 shows the original and combined probabilities for two filters, each with two behaviours. Note that at the end of the process behaviours $\{BehA, BehC, BehD\}$ all have the same posterior probability of 0.17. This is as one should expect, given that the particles for these behaviours all gained the same total weight of 25 in their individual filters.

6.2 Identifying Multi-agent Behaviour

The previous chapter introduced the hierarchical filter and it has been shown how the results of two (or more) filters can be merged. This allows multi-agent and solo behaviours to be filtered individually and yet produces a single estimate for the posterior $P(Z_T | y_{1:T})$. Until now the discussion has not considered how many agents are to be filtered, nor how those involved in multi-agent behaviour are known. These considerations are clearly application dependent and for some domains the number of agents and ‘collaborators’ are known and fixed. However, there are other domains in which this information is not known and must be derived. This section will introduce approaches for discovering multi-agent behaviour within the context of the hierarchical filter.

It has already been identified that each filter must be associated with the agent(s) being filtered and with an observation stream that provides observations of those agents only. Correspondingly, if there are two agents in the scene (1 and 2) then using the approach outlined produces three top-level filters; two filters for solo behaviours (one per agent), and one filter for the multi-agent behaviours. Initially, consider merging the results of

Filter S:			
$B_1^S = BehA$	$FW_1^S = 25$	$P(B_1^S y_{1:t})$	$= 0.25$
$B_2^S = BehB$	$FW_2^S = 75$	$P(B_2^S y_{1:t})$	$= 0.75$
Filter M:			
$B_1^M = BehC$	$FW_1^M = 25$	$P(B_1^M y_{1:t})$	$= 0.5$
$B_2^M = BehD$	$FW_2^M = 25$	$P(B_2^M y_{1:t})$	$= 0.5$
Step 1:			
FW^S	$= 25 + 75 = 100$		
FW^M	$= 25 + 25 = 50$		
$Eq(FW^S)$	$= \frac{FW^S}{NumParticles(S)}$	$= \frac{100}{100} = 1$	
$Eq(FW^M)$	$= \frac{FW^M}{NumParticles(M)}$	$= \frac{50}{100} = 0.5$	
$Imp(Eq(FW^S))$	$= \frac{Eq(FW^S)}{\sum_f Eq(FW^f)}$	$= \frac{1}{1.5} = 0.67$	
$Imp(Eq(FW^M))$	$= \frac{Eq(FW^M)}{\sum_f Eq(FW^f)}$	$= \frac{0.5}{1.5} = 0.33$	
Step 2:			
$P(BehA y_{1:t})$	$= Imp(Eq(FW^S)) \times P(B_1^S y_{1:t})$		
	$= 0.67 \times 0.25$		
	$= \mathbf{0.17}$		
$P(BehB y_{1:t})$	$= Imp(Eq(FW^S)) \times P(B_2^S y_{1:t})$		
	$= 0.67 \times 0.75$		
	$= \mathbf{0.5}$		
$P(BehC y_{1:t})$	$= Imp(Eq(FW^M)) \times P(B_1^M y_{1:t})$		
	$= 0.33 \times 0.5$		
	$= \mathbf{0.17}$		
$P(BehD y_{1:t})$	$= Imp(Eq(FW^M)) \times P(B_2^M y_{1:t})$		
	$= 0.33 \times 0.5$		
	$= \mathbf{0.17}$		

Table 6.1: Worked example of filter merging

the solo filter for agent 1 and the multi-agent filter 1,2. In essence the merged filter results estimate the overall posterior of agent 1 performing solo behaviours vs multi-agent behaviours with agent 2.

Now assume that agents 1 and 2 are not in fact engaged in multi-agent behaviour. One would expect that there will be a solo behaviour B_b^S that is more probable than any other behaviour: $P(Z_t = B_b^S | y_{1:t}) > P(Z_t = B_{b'}^f | y_{1:t}) \forall_{f,b'} b \neq b'$. The same holds true if the results for solo filter 2 are merged with multi-agent filter 1,2. If the behaviour with the highest posterior is selected as the most likely explanation of an agent's behaviour, and both behaviours are solo, then it has been determined that agents 1 and 2 are not performing multi-agent behaviour.

This raises the question: what happens if the most likely behaviour for agent 1 is solo but the most likely behaviour for agent 2 is multi-agent? This predicament can be solved by computing the joint probability of each combination and selecting the combination with the higher joint (Approach 1).

Approach 1	
if:	$\begin{aligned} & \max(P(Z_t^{\mathfrak{R}1} = B_{b1}^S y_{1:t}) \times \max(P(Z_t^{\mathfrak{R}2} = B_{b2}^S y_{1:t})) \forall_{b1,b2} \\ & > \\ & \max(P(Z_t^{\mathfrak{R}1,2} = B_b^M y_{1:t}) \times P(Z_t^{\mathfrak{R}2,1} = B_b^M y_{1:t})) \forall_b \end{aligned}$
then:	1 and 2 solo
else:	1 and 2 multi-agent

Note that $P(Z_t^{\mathfrak{R}1,2} = B_b^M | y_{1:t}) \neq P(Z_t^{\mathfrak{R}2,1} = B_b^M | y_{1:t})$ because $P(Z_t^{\mathfrak{R}1,2} = B_b^M | y_{1:t})$ is a merge of agent 1 solo behaviours with 1,2 multi-agent, while $P(Z_t^{\mathfrak{R}2,1} = B_b^M | y_{1:t})$ merges agent 2 solo behaviours. In each case the probability has been normalised and thus the joint probability $P(Z_t^{\mathfrak{R}1,2} = B_b^M | y_{1:t}) \times P(Z_t^{\mathfrak{R}2,1} = B_b^M | y_{1:t})$ must be considered to ensure that both agents are considered. If $\max(P(Z_t^{\mathfrak{R}1,2} = B_b^M | y_{1:t}), P(Z_t^{\mathfrak{R}2,1} = B_b^M | y_{1:t}))$ was compared to $\max(P(Z_t^{\mathfrak{R}2} = B_{b2}^S | y_{1:t}))$ then there is potential for the behaviour with maximal probability to not be shared by both agents.

For example, Table 6.2 shows some hypothetical probability estimates for agents 1 and 2. The most likely solo behaviours are B_1^S for agent 1 and B_2^S for agent 2, having a joint probability of 0.32. If $\max(P(Z_t^{\mathfrak{R}1,2} = B_b^M | y_{1:t}), P(Z_t^{\mathfrak{R}2,1} = B_b^M | y_{1:t}))$ was used to identify the most likely multi-agent behaviour then $P(B_1^M) = 0.5$ and correspondingly

Agent $\mathfrak{R}1$ Solo Behaviours	Agent $\mathfrak{R}2$ Solo Behaviours
$P(Z_t^{\mathfrak{R}1} = B_1^S y_{1:t}) = 0.4$	$P(Z_t^{\mathfrak{R}2} = B_2^S y_{1:t}) = 0.8$
$P(Z_t^{\mathfrak{R}1} = B_2^S y_{1:t}) = 0.1$	
Agent $\mathfrak{R}1, 2$ Multi-agent Behaviours	Agent $\mathfrak{R}2, 1$ Multi-agent Behaviours
$P(Z_t^{\mathfrak{R}1,2} = B_1^M y_{1:t}) = 0.5$	$P(Z_t^{\mathfrak{R}2,1} = B_1^M y_{1:t}) = 0.2$

Table 6.2: Example probability distributions for two agents.

$P(B_1^M)$ would be chosen as the most likely behaviour. However, this is clearly wrong because $P(Z_t^{\mathfrak{R}2,1} = B_1^M | y_{1:t}) = 0.2$, and when the most likely behaviour for agent 2 is determined it will be solo behaviour B_2^S . In this situation agent 1 is engaged in multi-agent behaviour with agent 2 but agent 2 is not involved. This is clearly nonsense. However, if $\max(P(Z_t^{\mathfrak{R}1,2} = B_b^M | y_{1:t}) \times P(Z_t^{\mathfrak{R}2,1} = B_b^M | y_{1:t}))$ is used instead, a value of 0.1 is obtained and the individual solo behaviours will be chosen as being most probable.

An alternative approach would be to compare the integrated probability of solo behaviour with the integrated probability of multi-agent behaviour (Approach 2). Then, having selected the most likely sets of behaviours, choosing the individual most likely behaviours for each agent from those sets.

Approach 2	
if:	$\int P(Z_t^{\mathfrak{R}1} = B_b^S y_{1:t}) db + \int P(Z_t^{\mathfrak{R}2} = B_b^S y_{1:t}) db$ $>$ $\int P(Z_t^{\mathfrak{R}1,2} = B_b^M y_{1:t}) db + \int P(Z_t^{\mathfrak{R}2,1} = B_b^M y_{1:t}) db$
then:	A and B solo
else:	A and B multi-agent

Approach 1 has been adopted in this research due to the simplicity of its implementation, though it is hypothesised that both approaches should give similar performance.

Approach 1 will identify whether agents 1 and 2 are engaged in multi-agent behaviour and by extension the same process can be applied to scenarios with more than two agents. The only difference is that the joint probabilities must include all agents, not just pairs of agents. Otherwise, an agent might be selected as engaged in multi-agent behaviour with more than one agent. This process of calculating the joint probabilities can be considered an optimal assignment problem, where the goal is to maximise the joint probability of all agent behaviours.

If the application domain demands that the most likely behaviour be identified each time a feature is observed, then filter merging and behaviour selection optimisation must be performed for each observation. Filter merging can be performed in $O(BN)$ time, where N is the number of agents in the system and B is the number of behaviours. In practice this operation is performed quite quickly (full run-time analysis is presented in Chapter 8), although subsection 6.4 identifies some fundamental limitations with the underlying approach of having multiple filters.

Behaviour selection optimisation is significantly more expensive and has a best-case run-time of $O(B^2N^2)$. Although this is feasible for small B and N , this combinatorial problem has an inherent scaling issue.

Heuristic Methods

Where as combinatorial optimisation generates possible solutions and identifies the best one, heuristic methods find good, but not necessarily optimal solutions. Because they are not guaranteed to find the optimal solution the complexity of heuristic algorithms is lower, at the trade-off that only a good solution is provided.

There are a number of heuristic algorithms for performing combinatorial search, with two popular approaches being Genetic Algorithms [99] and Simulated Annealing [70]. Genetic Algorithms are inspired by the principles of evolution: survival of the fittest. Through the process of natural selection organisms adapt to optimise their chances of survival. Better equipped organisms survive longer and have the opportunity to multiply, while weaker ones do not. As organisms multiply, random mutations occur that change the genetic make-up of children. If those mutations prove useful to survival then the children also survive longer, giving those mutations the opportunity to spread via the children's own offspring.

A genetic algorithm (GA) consists of a population of candidate solutions for a given optimisation problem. Members of the population are randomly selected to re-produce based upon some function of fitness that characterises how good the solution is. The 'offspring' of these solutions are constructed by combining aspects from both parent solutions (cross-over), and with the addition of small random changes (mutations) new solutions are generated. Each candidate solution in the population has its own fitness, and as evolution continues 'fitter' (better) solutions increase and are explored further, while weaker solutions 'die out' from the population.

Genetic Algorithms suffer from a number of problems that make their use limited. Firstly, each solution must be represented as a bit-string, which is unnatural for many problems. Furthermore, the process of exploration via cross-over and mutation is inefficient. These operations do not use any knowledge of the problem at hand, and thus fail to make progress in any structured way. As a result, many new (child) solutions are inferior, and progress towards better solutions is slow [128].

Simulated Annealing is an alternative heuristic method that draws its inspiration from the physical process of cooling metal [128]. When metal is cooled too quickly it becomes brittle, while a gradual reduction in temperature provides a better result. In thermodynamics the state of a system is determined by the energy of its particles, where higher temperatures increase energy. For the problem at hand, one can represent the probability of transitioning from lower state energy E_i to higher energy E_j via the probability:

$$P(E_i, E_j, T_{emp}) = e^{-\frac{(e_j - e_i)}{K_B T_{emp}}} \quad (6.14)$$

where K_B is a constant. This distribution has several important attributes. Firstly, as the temperature decreases, so too does the probability of transitioning from a lower energy state to a higher energy state. Secondly, smaller jumps in energy are more likely than bigger jumps.

The Simulated Annealing algorithm uses a similar concept, where physical particles are replaced by assignments that have an energy represented by cost. Ordinarily, the objective of simulated annealing is to minimise the overall cost of the system. However, for the problem at hand, the assignment energy is associated with its probability, and thus higher probabilities are preferred. The objective thus becomes to maximise the probability of the system. To reflect this goal, the system transition probability is updated by removing the negative from the exponent:

$$P(E_i, E_j, T_{emp}) = e^{\frac{(e_j - e_i)}{K_B T_{emp}}} \quad (6.15)$$

Changing an assignment is akin to a particle changing states. With the new goal of maximising assignment probabilities, any transition that leads to an increase should be accepted, while negative transitions penalised. However, if negative transitions were always rejected it is highly likely that the system would become stuck in a local maxima.

Instead, negative transitions are accepted according to the probability distribution 6.15.

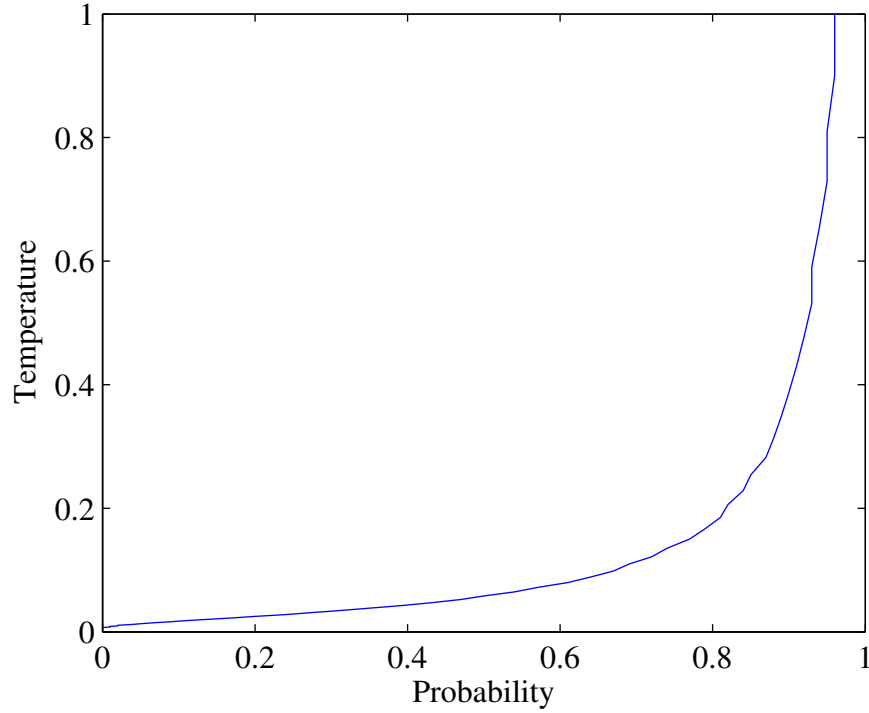


Figure 6.2: The probability of accepting a negative transition (-0.02) as the temperature changes

That is, when the temperature is high, an assignment change that reduces the system’s overall probability will be accepted with a high probability. However, as the system temperature reduces, the probability of accepting a negative change decreases. This is illustrated in Figure 6.2 which shows the probability of accepting a negative transition of -0.02 as the system temperature decreases.

Algorithm 6.1 summarises the application of Simulated Annealing to the multi-agent detection problem. It is provided with the merged posterior estimates for all agent combinations and returns the best assignment found. The transition step changes the assignment of one or more agents, where an assignment might be that agents 1 and 2 are engaged in multi-agent behaviour, or that agents 1 and 2 are solo. If the joint probability of the new assignment with all other (unaltered) assignments leads to a higher probability, the change is accepted. If it leads to a lower probability, it is accepted according to the annealing distribution in Equation 6.15. Changes to the assignment are proposed for a specified number of iterations before decreasing the temperature. The whole process then repeats until the solution becomes stable. Details of the exact implementation of this algorithm can be found in Chapter 7.

To illustrate the algorithm with an example, Figure 6.3 shows a simple trace of one iteration of the algorithm. The algorithm is invoked each observation and is sequentially

Algorithm 6.1 Simulated Annealing for Multi-Agent Behaviour Detection

```
1: Prototype:  $[S] = \text{simAnnealing}([P(Z_t|y_{1:t})])$ 
2: Init:
3:  $S = \text{initialSolution}()$ 
4:  $T_{emp} = \text{initTemperature}()$  {Typically 1}
5:  $K_B = \text{InitialKb}()$  {Typically 1-10}
6:  $\alpha = \text{initTemperatureDecrementRate}()$  {Typically in  $[0.8, 0.99]$ }
7: repeat
8:   for  $i = 1 : \text{numIterations}$  do {Typically 100-1000}
9:      $C = \text{generateRandomTransition}(S)$ 
10:    if  $P(C) \geq P(S)$  then
11:       $S = C$ 
12:    else
13:      if  $e^{\frac{(P(C)-P(S))}{K_B T}} > \text{random}([0, 1])$  then
14:         $S = C$ 
15:      end if
16:    end if
17:  end for
18:  {Reduce the temperature by  $\alpha$ }
19:   $T_{emp} = T_{emp} \times \alpha$ 
20: until no change in  $S$ 
```

executed after the particle filters have generated the probability density estimates for each agent/behaviour. This is shown in line 1 of the example where it is assumed there are a total of two agents and three possible behaviours. In lines 2-6 the algorithm is initialised as can be seen in the figure. In this example the initial solution is generated by assuming that all agents are performing solo behaviours, and thus their most probable solo behaviours can be observed in the figure. On line 8 the first iteration of the algorithm begins, and a random solution transition is generated on line 9. In this example the transition is that agents 1 and 2 are involved in multi-agent behaviour and thus both agent assignments are changed in the candidate solution C . After calculating the joint probability of the original (initial) and new (candidate) solutions one can see on line 10 that the new solution is an improvement, and thus the transition is accepted and forms the new (working) solution (line 11). The algorithm would then repeat for a fixed number of iterations before decrementing the temperature, and this process in turn would repeat until a stable solution was found.

Although in this simple example only two agents are present and only one multi-agent behaviour exists, a more complex scenario could involve many more agents. A transition might switch a multi-agent behaviour to a solo behaviour (for both agents), or could alter the agent:agent assignment. Such an alteration might swap agents 1 and 2 being involved in behaviour 3 for an the assignments: agent 1 and agent 3 (and their most probably multi-agent behaviour), and agent 2 being changed to a solo behaviour.

1. Input: Probability estimates from particle filters for $t=1:S$ for all agents and all behaviours

Agent 1 Solo Behaviours	Agent 2 Solo Behaviours
P(Behaviour1) = 0.1	P(Behaviour1) = 0.2
P(Behaviour2) = 0.4	P(Behaviour2) = 0

Agent 1,2 MultiAgent Behaviours	Agent 2,1 MultiAgent Behaviours
P(Behaviour3) = 0.5	P(Behaviour3) = 0.8

- 2 - 6. Initialisation:

$S =$

	Agent 1	Agent 2
Behaviour	Beh. 2	Beh. 1
Probability	0.4	0.2

$Temp = 1$

$K_b = 1$

$alpha = 0.9$

8. Iteration $i = 1$ begins

9. Generate Random Transition:

	Agent 1	Agent 2
Behaviour	Beh. 3	Beh. 3
Probability	0.4	0.2

$$P(S) = 0.4 * 0.2 = 0.08$$



	Agent 1	Agent 2
Behaviour	Beh. 3	Beh. 3
Probability	0.5	0.8

$$P(C) = 0.5 * 0.8 = 0.4$$

$= C$

10. $P(C) \geq P(S)$

11. $S = C$ (Accept the new solution)

- 17 - 18. Algorithm repeats for fixed number of iterations before reducing temperature

$$Temp = Temp * alpha = 1 * 0.9 = 0.9$$

20. Algorithm repeats until solution becomes stable

Figure 6.3: A simple example trace of the simulated annealing algorithm.

6.3 Predicting Agent Behaviour

The simulated annealing algorithm not only identifies which agents are engaged in multi-agent behaviour, but also identifies what behaviour each agent is performing such that the overall probability of the observations is maximised. At its simplest, predictions can be generated based on this information alone. However, the annealing algorithm does not indicate how much of a behaviour has been observed, nor how stable that assignment is. Because it is likely that such information would be extremely useful for many applications, this section considers how it might be obtained from the filter.

The inherent structure of the DBN provides some very useful information about what behaviour has been observed. The set of currently observed features tracks not only the observations, but also the behavioural roles that they satisfy. By extracting this information from all particles in a filter a generalised picture can be formed and used as part of the prediction process.

Recognising where in the behaviour the agent is, or how much of it has been performed, is likely to be extremely useful for determining how reliable the current prediction (most likely behaviour) is. If only a quarter of a behaviour's features have been observed then the confidence in that prediction should remain reasonably low. Yet if three-quarters of a behaviour have been observed our confidence should be increased. For any given behaviour B_b , the status distribution can be represented as:

$$P(C_t|T_t = B_b) \approx \sum_{i=1}^N \omega^i \delta(C_t^i, C_t) \delta(T_t^i, B_b) \forall_i \in f_t^B \quad (6.16)$$

where f_t^B represents the filter for behaviour type B (e.g. solo) at time t . From this posterior one can determine the probability that a subset of features have been observed.

It may be desirable for predictions to only be communicated to an operator when at least half of a behaviour has been observed. This could be trivially achieved as follows:

$$\exists_n : P(C_t^n|B_b) > Th \wedge \frac{|C_t^n|}{numPrim(B_b)} > 0.5 \quad (6.17)$$

where Th is some certainty threshold. A similar approach can be taken in determining to

which role agents have been assigned, allowing a prediction to be explained to an operator in terms of ‘who did what’.

Although the filtering distribution itself tells us how stable a behaviour prediction is, what it does not tell us is the stability of the solo/multi-agent assignment. For this, the output of the simulated annealing algorithm must be tracked. Assignments that frequently change between observations are likely to be unreliable, and thus the associated behaviour predictions, which are restricted by the assignment, are also unreliable. That is because a ‘multi-agent’ assignment between agents 1 and 2 enforces predictions based on the multi-agent filter $f^{\mathcal{R}_{1,2}}$, and thus discounts solo behaviours and multi-agent behaviours with other agents.

Because behaviour prediction is restricted by multi-agent assignments the overall prediction process is reliant upon good simulated annealing results. This makes the annealing process a critical point in the architecture, as it has the ability to negate much of the efforts of the particle filters. Such a property is undesirable and represents a limitation of this approach, although no scalable alternatives are apparent.

6.4 Limitations

Throughout this chapter it has been hinted at that the process of identifying multi-agent behaviour presents a scalability issues. When the behaviours being filtered are restricted to a maximum of two agents the solution scales well if the agents performing a multi-agent behaviour are known. But this is not so true if those involved in multi-agent behaviour are not known. In a three-agent environment there are three potential pairs of ‘collaborators’ and thus three filters are required just to represent the multi-agent behaviours. If only the goal behaviour filters for each solo/pair are considered, then three agents requires six filters, four agents requires ten, and five agents requires fifteen filters. It should be clear that this is polynomial growth and thus presents a scaling issue.

What’s more, this issue arises when behaviours are restricted to pairs of agents. The growth rate is larger if behaviours can consist of three or more agents. Chapter 8 will analyse these growth rates in more detail to identify at what point they becomes unviable.

6.5 Summary

This chapter has presented an approach for recognising multi-agent behaviour without prior knowledge of agent groupings. At its core combinatorial search is used to find the most likely behaviours for all agents. To achieve this the posterior probability of each multi-agent behaviour for each pair of agents must be calculated, in addition to calculating the posterior of each agent performing solo behaviour. The posterior filtering density of each filter can be combined to give a joint estimate, which is then provided to the optimisation algorithm.

Simulated Annealing is a heuristic optimisation algorithm that is efficient and delivers good solutions. This algorithm is used to identify the best joint probability of all agents performing behaviours which not only tells us the most likely behaviour for each agent, but also identifies those involved in multi-agent behaviour. Because the approach is built upon combinatorial search it suffers from an inherent scaling issue. The number of filters required grows exponentially with the number of agents and this will impact on run-time.

As this chapter concludes the theoretical description of our approach has now been completed. The next chapter will discuss how this framework was implemented and evaluated and then proceed to discussing the results in the chapters that follow.

Chapter 7

Implementation

Nomenclature:

Bl_t	Set of foreground pixel blobs	N	Number of agents
El_t	Subset of Bl_t matching ellipsoids (diameter: 0.4m, height: 1.8m)	T_{emp}	Initial temperature
$SmBl_t$	Subset of Bl_t with $0.3 \leq \text{height} \ \& \ \text{width} \leq 1\text{m}$	α	Temperature decrement rate
r	Likelihood ratio	I	Number of simulated annealing iterations
B^k	The k th behaviour	C	Set of observed features

In the previous chapters the bag-of-features approach has been introduced via generic, domain independent algorithms. To evaluate these algorithms they will now be applied to the domain of automated visual surveillance. This is one domain in which annotated corpora is rarely available due to both privacy concerns, and the time consuming nature of annotating video. As discussed in Chapter 2, the majority of existing research in this area has focused upon non-probabilistic event matching techniques which are unable to reason about uncertainty. The application of our approach to this area thus represents a large step-forwards in probabilistic automated visual surveillance.

The implementation framework can be broadly separated into three levels: Image Processing, Reasoning, and Operator Interaction. Within each layer a number of ordered processes are performed, while different layers are strictly de-coupled to facilitate the integration and replacement of different processing techniques. The overall framework is

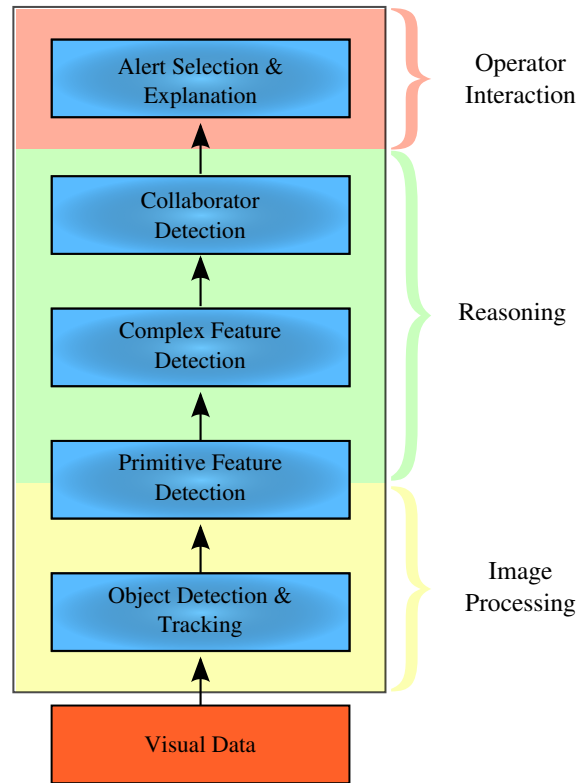


Figure 7.1: The 3-Layer Implementation Framework

visually represented in Figure 7.1. It should be noted that the implementation does not attempt to inform or improve lower-level recognition as a result of high-level inference, although such an integration could certainly provide interesting future work.

This chapter’s layout broadly follows that of Figure 7.1. The first section will introduce the components required to recognise primitive features from raw video data. This is followed by implementation details for the Hierarchical Rao-Blackwellised Particle Filter (introduced in Chapter 5), and the multi-agent behaviour detection algorithm (introduced in Chapter 6). Section 7.3 will then discuss the criteria by which predictions are made and behaviour explanations generated. Finally, the chapter will conclude with a discussion of the primitive feature simulator which is used to generate substantially more test cases than could have been viably generated from video alone. The simulated data is used in conjunction with real video data throughout the evaluation.

7.1 Image Processing

The image processing stage consists of two steps. Firstly, Object Detection and Tracking provides information about scene objects. The objects of interest are foreground objects, that is, they enter the scene and are separate from the static background. The movements of foreground objects are tracked and allow simple semantic events to be detected during the second step. These events are represented as primitive features, and provide input to the Reasoning layer of the framework.

7.1.1 Object Detection and Tracking

Static cameras allow foreground pixels to be identified using background subtraction. This technique was introduced earlier (Section 2.2), but to summarise, it compares the current video frame with a known background frame. Pixels that are different according to some threshold are classed as foreground pixels, and connected foreground pixels give foreground blobs. These blobs will collectively be referred to as Bl_t . Because calibrated cameras are used the size and location of each blob can be projected into real-world coordinates and used to calculate the proximity of different objects. This will prove useful for primitive feature detection in Section 7.1.2.

Two separate trackers operate on Bl_t :

Person Tracker

The person tracker consists of a set of SIR filters [54]. These filters implement Sequential Importance Sampling with Re-sampling, a technique that has already been introduced generically in Section 2.1. The implementation used has been provided ‘as is’ by [84] and uses one hundred particles to represent the person’s position on the ground plane, velocity, and direction of travel. For each video frame, the blobs (groups of foreground pixels) that contain people are quickly identified from Bl_t using ellipsoid detection, which identifies ellipsoids with height $\approx 1.8m$ and diameter $\approx 0.4m$. Denote the set of these blobs as El_t .

For each element of El_t that cannot be explained by an existing filter, a new filter is

instantiated to track that person. The start of a new track thus indicates the entry of an agent into the scene, while the absence of a track indicates that the person associated with that tracker has been lost. In order to address the temporary occlusion of a person (e.g. people crossing paths), particles also contain a visibility variable (0/1) to indicate the person's disappearance. This variable applies to all particles in the filter. By combining this variable with a time limiting threshold, the filter continues to predict the person's location for short occlusions, while longer occlusions will cause the track to actually terminate.

In order to distinguish between agents each tracker is initialised with a unique identifier. Tracker output thus consists of time indexed tuples containing the person's ID, their coordinates within the image frame, and their real-world coordinates.

Object Tracker

The second tracking component consists of an object detector which is used to detect static objects having the appearance of luggage. The object detector uses a number of heuristics to identify static 'luggage like' objects. The first heuristic is applied to the size of the foreground blob and constrains its real-world width/height (x) to: $0.3m \leq x \leq 1m$. This range encompass a variety of items while excluding overly large or small blobs. Excluding very-small blobs is particularly important in counteracting the effects of lighting changes, which spuriously create small foreground blobs. The use of appearance based heuristics is not unprecedented in automated video surveillance research, with other examples including [91, 84, 130].

The second heuristic used by the object tracker is location and time based. For each element in the set $SmbI_t$, which denotes the set of blobs satisfying the size constraints, an element is classified as a stationary luggage item if its centroid remains within $0.3m$ of its original position for one second. A successful classification initialises the tracking of the object, to which a unique object identifier is associated. As with the person tracker, object tracks are comprised of the unique object ID and the object's image frame/real-world coordinates.

A static object continues to be tracked while its position remains stationary. That is, while the blob's centroid remains within the $0.3m$ sphere around it's origin. If the centroid moves outside this region for more than one second, the object track is terminated.

Module	Description
Agent Tracker	Detects the entry/departure of people from the scene.
Object Tracker	Upon luggage detection, associates that luggage with the closest person. Object removal is associated with the last closest agent.
Group Tracker	Identifies when people are in close proximity, and split from a single location.

Table 7.1: The primitive feature detection modules

Parameter Sensitivity

It should be acknowledged that the trackers are reasonably sensitive to the heuristic parameters discussed. In the case of the person tracker small children cannot be detected, although outside of this it performs reasonably well at tracking a range of different height individuals (160 – 183cm). The object tracker too can be sensitive and these parameters were drawn both from other work in the field and through manual tuning. This thesis does not suggest that these trackers would be robust in more complex situations and their detection accuracies will be discussed in Chapter 9.

7.1.2 Primitive Feature Detection

Primitive features are correlated with a set of simple video events that can be detected from video tracking information. Detection is handled via three separate modules which encode simple semantic rules. Each module outputs observations in the form of primitive features, which are provided as input to the reasoning layer of the framework. The modules are summarised in Table 7.1.

The first module processes person tracking data to identify when tracks begin and end. The commencement of a track produces the observance of an *EnterAgent* primitive, while the termination of a track produces an *ExitAgent* primitive. It should be noted that no temporal information is to be assumed when a primitive is detected. For instance, the occurrence of *EnterAgent* does not indicate that the agent has recently entered the scene, but merely that the agent has been identified as a person and is now being tracked. In severely occluded scenarios, such as when people enter as a group, individual agents are often not identified until the viewing angle changes or the group structure changes. It is therefore possible for *EnterAgent* features to be generated for agents who have been

present but undetected in the scene. In a similar vein, *ExitAgent* primitives only indicate the absence of an agent previously detected, and could be caused by long occlusions rather than actual scene departure.

The second module processes both object and person tracking data. Its primary purpose is to identify when objects are detected or lost, generating *ObjectPlaced* and *ObjectRemoved* primitives. Unlike the previous two primitives, object related primitives cannot automatically be associated with an agent. To achieve an association this implementation uses the naive approach by using agent tracking information to infer object ownership. That is, the agent closest to the object is chosen as the activity performer. This approach is consistent with other work in the field (e.g. [91]). There are clearly limitations with this approach, and although not explored within this research, there is huge scope for utilising more accurate activity detection techniques to improve detection accuracy.

Finally, it should be highlighted that *ObjectPlaced* and *ObjectRemoved* are not only associated with the closest agent, but are also associated with the object identifier that caused the event. In this way it is clear that *ObjectPlaced(obj₁)* and *ObjectRemoved(obj₁)* refer to the same object but that *ObjectPlaced(obj₁)* and *ObjectRemoved(obj₂)* do not.

The final module is referred to as the Group Tracker and uses agent proximity to determine when groups of agents form and split (disband). Again, this technique is naive and could be replaced with more advanced techniques that actually detect group interaction. The development of such algorithms is beyond the scope of this research, although some suggestions for future work are presented in Chapter 11.

The Group Tracker uses a number of heuristics to identify group formation/disbandment. Firstly, groups of agents in this context are assumed to be within close proximity, and thus agents must be within two metres. Secondly, groups are assumed to have similar motion. A set of agents are therefore only classed as grouped if they remain in close proximity for three or more seconds. Thus, for a set of agents having proximity greater than two metres, who then close their proximity below this threshold and remain close, a *GroupFormed* primitive will be generated. Inversely, if the proximity between grouped agents exceeds the threshold for more than two seconds, a *GroupSplit* primitive is generated. Finally, the remaining scenario is when multiple agents enter the scene in close proximity. Under these circumstances a *GroupFormed* primitive is not generated because the agents are already grouped.

Chapter 9 will show that these modules deliver varying degrees of accuracy on the real video datasets, and it is important to acknowledge that they would be insufficient for more

complex video. The focus of this research is high-level inference and thus state-of-the-art video processing techniques may not have been used. However, the modularity of the framework allows any component to be swapped, and thus readily supports the adoption of improved video processing algorithms.

7.2 Reasoning

This section focuses on the reasoning layer of the framework and contains two components: Complex feature detection and multi-agent behaviour detection. Complex feature detection has been introduced in its generic form in Chapter 5 and requires very few implementation specific considerations. However Section 7.2.1 will fully specify the set of behaviours used in the implementation. Similarly, generic collaborator detection was introduced in Chapter 6 and contains several parameters that must be tuned. Section 7.2.2 will detail the parameter values used in addition to presenting a slightly modified algorithm.

7.2.1 Complex Feature Detection

Seven behaviours were identified to be used throughout the evaluation. Four of these behaviours can be directly observed in the publicly available PETS 2006 dataset. This dataset was originally produced for low-level video event recognition. Although no existing research has considered complex events within the data, its widespread use within the video community, and its realism, make it a good choice of dataset for future comparisons between different approaches.

In addition to the four directly observable behaviours, a fifth behaviour can be generated by merging tracking information from two separate videos. This *synthetic* scenario encapsulates all of the low-level classification errors (noise) and realism of the component videos and thus synthesis very realistic data. The use of synthetic scenarios is not uncommon in prior work (e.g. [27, 62]), where it is often recognised that extending existing datasets is a difficult task. Visually, the synthetic behaviour is very similar to others, yet has important semantic differences that distinguish it as unique. Finally, two further behaviours were defined which are not present nor synthesised in the PETS 2006 dataset. Again, these behaviours were chosen due to their similarity to others, while having distinctly different semantic explanations. In addition to the PETS dataset, a second video

Name	Description	Present in PETS	Present in HW dataset
Passing Through 1 (PT1)	Person enters and leaves the scene	Y	Y
Passing Through 2 (PT2)	Persons enters, temporarily places luggage, then leaves the scene (with luggage)	Y	Y
Watched Item (WI)	Two people enter the scene as a group. One places luggage and leaves the scene without it. The other person remains in the scene.	S	Y
Abandoned Object 1 (AO1)	Two people enter the scene independently. The people temporarily form a group. One person places luggage and leaves the scene without it.	Y	Y
Abandoned Object 2 (AO2)	Person enters the scene, places luggage and leaves without it.	Y	Y
Theft (Th)	Person enters the scene and places luggage. Second person enters scene and removes other agent's luggage, leaving the scene with it.	N	Y
Hand-Off (HO)	Person enters the scene, places luggage and leaves without it. Second person enters the scene, removes luggage and leaves the scene with it.	N	Y

Table 7.2: Goal behaviours used in the evaluation

dataset containing all the behaviours was gathered for the evaluation (termed the HW data) and is fully discussed in Chapter 9.

Table 7.2 summarises the details of each of the seven behaviours, including whether they are directly observable in each video dataset. Each behaviour is represented by a complex feature in the Hierarchical Rao-Blackwellised Particle Filter. The structure of each Behaviour can be found in Appendix A.

7.2.2 Collaborator Detection

Collaborator detection is heavily based upon the Simulated Annealing algorithm presented in Chapter 6. There are three parameters to this algorithm:

- T_{emp} : Initial temperature (value used: 1)
- α : Temperature decrement rate (value used: 0.9)
- I : The number of iterations (value used: 50)

During experimentation it was found that several short runs (in terms of iterations) produced better results than longer runs. The algorithm was thus altered to perform several cycles, as summarised in Algorithm 7.1. In essence, an additional loop has been inserted around the algorithm to initialise the solution in different ways. On the first cycle the solution is initialised by assigning each agent to the solo behaviour with the highest probability. However, at the end of each cycle the solution utility (joint probability of all assignments) is compared against the best solution so far, storing the new assignment when appropriate. This assignment is then used to re-initialise the solution on the next cycle.

7.3 Operator Interaction

The Operator Interaction layer provides a window through which operators gain insight into the behaviours that have been detected. In essence it is responsible for filtering the information generated during inference to provide a succinct summary of interesting behaviour. Interesting behaviour can be defined by two attributes: the criteria defining ‘interestingness’, and the behavioural aspects that are important to an operator. The interestingness of a behaviour thus determines whether a prediction should be presented to the operator, while the important behavioural aspects determine what details should be encompassed by a prediction.

Algorithm 7.1 Multi-Cycle Simulated Annealing for Collaboration Detection

```
1: Prototype:  $[S] = \text{simAnnealing}([P(Z_t|y_{1:t})])$ 
2: Init:
3:  $S = \text{initialSolution}()$ 
4:  $Su = \text{utility}(S)$ 
5:  $T = \text{initTemperature}()$  {Typically 1}
6:  $\alpha = \text{initTemperatureDecrementRate}()$  {Typically in  $[0.8, 0.99]$ }
7:  $Bu = 0$  {Best solution utility}
8:  $Bs = \emptyset$  {Best solution}
9: for  $cy = 1 : 10$  do
10:   if  $cy > 1$  then
11:     if  $Su < Bu$  then
12:        $S = Bs$ 
13:     end if
14:   end if
15:   repeat
16:     for  $i = 1 : \text{numIterations}$  do {Typically 100-1000}
17:        $C = \text{generateRandomTransition}(S)$ 
18:       if  $P(C) \geq P(S)$  then
19:          $S = C$ 
20:       else
21:         if  $e^{\frac{P(C)-P(S)}{K_B T}} > \text{random}([0, 1])$  then
22:            $S = C$ 
23:         end if
24:       end if
25:     end for
26:      $T = T \times \alpha$  {Reduce the temperature by  $\alpha$ }
27:   until no change in  $S$ 
28:   if  $\text{utility}(S) > Bu$  then
29:      $Bu = \text{utility}(S)$ 
30:      $Bs = S$ 
31:   end if
32: end for
```

7.3.1 Prediction Criteria

In many applications, including visual surveillance, some behaviours can be considered uninteresting. That is, they are expected and are of no cause for alarm. However, other behaviours, though routine, may be more concerning and are of particular interest to operators. Of the seven behaviour used throughout the validation three involve normal transport-hub behaviour (*PT1*, *PT2*, *WI*). The remaining four behaviours are more suspicious and involve object abandonment/theft, and should therefore be monitored more closely (*AO1*, *AO2*, *TH*, *HO*). One can thus begin to build a set of criteria for ‘interesting’ behaviours based upon the application domain.

Another criteria that can be considered when making predictions is how far a behaviour has progressed. The bag-of-feature approach is a ‘minimum explanation’ system that favours shorter explanations, as discussed in Chapter 4. This means that for a set of observed activities and two behaviours that can explain those activities, the behaviour with fewer components will be more probable. Even after a single observation, one (very short) behaviour may be substantially more probable than the others, so it is important that when making predictions the level of behaviour completion is considered in conjunction with the behaviour probability.

The experimental evaluation uses both of these criteria for behaviour prediction. Firstly, predictions are restricted to those behaviours that have been observed in their entirety. That is, all features representing a behaviour have been observed. This information can be deduced from the hierarchical particle filter by calculating the mean number of elements of variable *C* for the most probable behaviour. Secondly, all behaviours are selected for communication once they have been observed, although they are partitioned into two sets. The first set represents less interesting behaviours and it is ensured that the information conveyed for these predictions is minimal. The second set contains concerning behaviours, where additional details are presented to the operator (see next section).

7.3.2 Prediction Detail

All predictions have a basic form to indicate the agent and behaviour detected:

“Agent X has performed Behaviour B. Certainty: C”

The certainty is drawn from the set $\{low, medium, high\}$ and is determined via a numerical mapping of the likelihood ratio. The likelihood ratio is simply the probability of the behaviour divided by the next most probable behaviour within the same class. (e.g. solo behaviours). So for instance, where two behaviours have the same probability a ratio of 1 is produced and is mapped to *low* certainty. The mapping rules chosen were :

$$r < 1.5 \quad : \quad low \quad (7.1)$$

$$r \geq 1.5 \quad : \quad medium \quad (7.2)$$

$$r > 2.0 \quad : \quad high \quad (7.3)$$

As a demonstration, suppose that $P(B^1) = 0.25$ and $P(B^2) = 0.5$, giving a likelihood ratio $r = \frac{0.5}{0.25} = 2$. This implies a high certainty, which seems reasonable given that B^2 is twice as likely as B^1 given the evidence. Similarly, consider when $P(B^1) = 0.25$ and $P(B^2) = 0.3$. These probabilities are relatively similar and give a ratio of only 1.2, indicating a low certainty. Again, this seems reasonable.

For predictions from the concerning set of behaviours additional behaviour specific details are provided to the operator. These are specified via templates using feature chains to identify key events. For example, the *Abandon Object 1* template is:

“Agent < ROLE – 0 > has left luggage with agent < ROLE – 1 >, but they are not known companions. < ROLE – 0 > may have abandoned an object. The item was placed around frame < 0 : ObjectPlacedPrimitive : LeaveObjectComplex > and abandoned around frame < 0 : ExitAgentPrimitive : LeaveObjectComplex >.”

Note that the use of feature chains to specify events is consistent with the *currently achieved* feature set variable *C*. It is trivial to implement the hierarchical particle filter

in such a way as to store the time that a feature was observed, and thus the *mean* time can be extracted and placed into the template string during the prediction details stage. This approach allows relevant event times to be communicated to the operator during prediction.

7.4 Primitive Feature Simulator

Section 7.1 described a framework for processing raw video data into primitive feature observations. Due to the limited availability of real video datasets, a primitive feature simulator was also developed to mimic the low-level video processing, allowing more scenarios to be generated than could be feasibly recorded by actors. The simulator thus used the set of detectable behaviours as a plan library from which potential observation sequences could be generated. Similar approaches have also been taken in prior work such as [78], and allows the rapid generation of potential observation sequences.

In order to preserve realism the simulator incorporated two levels of low-level classification errors. Firstly, an over-all classification error insertion rate determined the probability that a spurious observation would be inserted into the observation stream. A rate of 10% would thus ensure that one in ten observations was erroneous, where classification errors were in the form of any primitive feature observation. The distribution of classification error types was determined via a second parameter associated with each primitive feature type. This parameter was configured to be similar to the error rate of the real primitive feature detection modules introduced in Section 7.1 and thus ensured that classification errors had a realistic distribution. Similarly, the over-all classification error rate was configured at 10% to be similar to the real-video detections.

In addition to primitive feature classification errors the simulator also aided realism by ensuring multiple agents were concurrently present in the scene. This configurable parameter allowed any number of agents to be simulated concurrently. Concurrency in this context means that the observation stream contained primitive features for N agents, and thus all agents are potential collaborators. Algorithm 7.2 illustrates the simulation process via pseudo code. The algorithm is called with an error rate and model so that classification errors (discrete noise) can be inserted, in addition to the number of agents the scenario should contain, and the set of all (known) behaviours from which agent behaviours should be chosen. The algorithm performs three steps:

1. Agent behaviour generation: Multiple observation traces are generated until N agents have been simulated. Generation commences by randomly selecting a behaviour from the set of known behaviour, from which an observation sequence can be produced. Unique agent identifiers are generated as necessary and added to the set of agents, and thus a behaviour requiring two agents will cause the addition of two new agents to be added to this set, and a two-agent observation trace to be produced. This trace is added to the set of scenarios and the process repeats (selecting a new random behaviour) until a total of N agents have been simulated.
2. Single stream generation: The scenarios produced during the first step are now interspersed to provide a single observation sequence. This interspersion is random, although the sequential ordering of observations remains consistent. As an example, a behaviour trace with one agent and two observations may be split such that the first observation becomes the first/second/third/etc. observation in the output sequence, and the second observation may become the last/second from last/etc. The output of this step is thus a single sequence of observations containing each agent behaviour trace in a sequentially consistent, but randomly interspersed order. An example of this is shown in Figure 7.2 for three fictitious behaviours and four agents.
3. Error insertion: The final step of the algorithm is to apply the error model to the single observation sequence. This is performed by selecting random points in the observation stream into which classification errors should be inserted. For each error location the error model is used to determine the observation (feature) to insert, which is then attributed to a randomly selected agent from the list of generated agents. The final observation stream (S) is thus a mixture of N agent traces and an ER proportion of classification errors. Again, Figure 7.2 shows an example of this process, in which one classification error is inserted into the final observation stream.

7.5 Summary

The components that will be used for the evaluation have now been introduced. The chapter began with a discussion of the image processing framework, where the first level dealt with object and person detection and their tracking between video frames. This tracking information provided input to the second level of processing where activities were detected and their associated primitive features were emitted. These features formed input to the reasoning layer of the framework where high-level inference is performed

Individual Behaviour Traces:

Agents	Behaviour Trace
1	Ag1(Prim1), Ag1(Prim2), Ag1(Prim3)
2 & 3	Ag2(Prim1), Ag3(Prim1), Ag2(Prim2), Ag2(Prim3), Ag3(Prim2), Ag2(Prim4)
4	Ag4(Prim1), Ag4(Prim2)



*Conversion To Single
Observation Sequence*

Single Observation Trace:

Time-step	1	2	3	4	5	6	7
Observation	Ag2(Prim1)	Ag1(Prim1)	Ag3(Prim1)	Ag4(Prim1)	Ag1(Prim2)	Ag2(Prim2)	Ag2(Prim3)

Time-step	8	9	10	11
Observation	Ag4(Prim2)	Ag1(Prim3)	Ag3(Prim2)	Ag2(Prim4)



*Addition Of
Classification Errors*

Final Observation Trace:

Time-step	1	2	3	4	5	6	7
Observation	Ag2(Prim1)	Ag1(Prim1)	Ag3(Prim1)	Ag4(Prim1)	Ag1(Prim2)	Ag2(Prim2)	Ag2(Prim3)

Time-step	8	9	10	11	12
Observation	Ag4(Prim2)	Ag2(Prim2)	Ag1(Prim3)	Ag3(Prim2)	Ag2(Prim4)

Figure 7.2: Primitive Feature Simulator Example: Individual agent behaviours are merged into a single observation stream upon which the error model is then applied to insert classification errors.

using bag-of-features inference. The Operator Interaction layer forms the top layer of the implementation framework and serves as an interface between high-level inference and human operators. This layer determines when behaviour predictions should be made by determining when a behaviour terminates. If the behaviour is regarded as interesting the detection report includes pertinent information such as event times, while less interesting behaviours are merely reported as having occurred and by who.

In addition to the video data the evaluation also makes use of simulated data. This data is generated by the primitive feature simulator which produces streams of primitive feature observations. To ensure realism primitive feature classification errors are inserted into the observation streams with a similar distribution to the real data. Furthermore, the number of concurrent agents can be manually selected allowing complex multi-agent scene to be simulated.

Algorithm 7.2 Generating a simulated scenario

```
1: Prototype:
2:  $[S] = \text{GenerateScenario}(ER, EM, N, AllB)$ 
3: {where ER=Error Rate, EM=Error Model, N = Number of agents, AllB = Set of all
   known behaviours}
4:
5: Init:
6:  $numberOfAgents = 0$ 
7:  $scenarioIndex = 0$ 
8:  $scenarios[] = \emptyset$ 
9:  $agents[] = \emptyset$ 
10:  $observationSequence = \emptyset$ 
11:  $S = \emptyset$ 
12: {Step 1: Generate scenarios for N agents}
13: repeat
14:    $beh = \text{SelectRandomBehaviour}(AllB)$ 
15:    $numberOfAgents = numberOfAgents + numberOfAgentsRequired(beh)$ 
16:    $seq = \text{GenerateObservationTrace}(beh, agents)$ 
17:    $scenarios[scenarioIndex] = seq$ 
18:    $scenarioIndex = scenarioIndex + 1$ 
19: until  $numberOfAgents \geq N$ 
20: {Step 2: Merge the scenarios into a single observation sequence}
21: repeat
22:    $i = \text{RandomInteger}(scenarios.Size())$ 
23:    $currentSeq = scenarios[i]$ 
24:    $observation = currentSeq.RemoveFirstElement()$ 
25:    $observationSequence.Append(observation)$ 
26:   if  $currentSeq.Size()=0$  then
27:      $scenarios = \text{RemoveEmptyIndex}(scenarios, i)$ 
28:   end if
29: until  $scenarios.Size() = 0$ 
30: {Step 3: Apply the error model to the sequence at the appropriate error level}
31:  $S = observationSequence$ 
32: for  $i = 1 : observationSequence.Size()$  do
33:    $r = \text{RandomDouble}()$ 
34:   if  $r < ER$  then
35:      $classificationError = \text{GenerateRandomError}(agents, EM)$ 
36:      $S.Insert(classificationError, i)$ 
37:   end if
38: end for
39: return S
```

Chapter 8

Experimental Evaluation via Simulation

Nomenclature:

D_t	Agent's desire at time t	FN	False negative rate
A_t	Observed activity at time t	DT	Density threshold
TP	True positive rate	N	Number of particles
FP	False positive rate		

The performance of bag-of-features inference on a high-level behaviour recognition task is now considered. This evaluation is performed in the automated surveillance domain and can be partitioned into two stages: experiments on simulated data, and experiments on real video data (Chapter 9). Because suitable video surveillance datasets are limited the simulated data allows data variability to be increased and thus more rigorous experiments can be performed. These results are then combined with the experiments performed on real video data to demonstrate end-to-end processing and validation of the overall recognition framework.

All of the experiments make use of the behaviours introduced in the previous chapter and summarised in Table 8.1. Again, the structure of these behaviours can also be found in Appendix A. For brevity the results and discussion will generally refer to behaviour short names. Multi-agent observation sequences were generated using the primitive feature simulator. Each simulated sequence contained a varying number of agents (10 – 20) and low-level classification errors in the form of random primitive features. There are two

Full Behaviour Name	Short Name Notation	Multi-agent/Solo
Passing Through 1	PT1	S
Passing Through 2	PT2	S
Watched Item	WI	M
Abandon Object 1	AO1	M
Abandon Object 2	AO2	S
Theft	TH	M
Hand-Off	HO	M

Table 8.1: Summary of evaluation behaviours

Feature Name	Probability
EnterAgent	0.03
ExitAgent	0.03
PlaceObject	0.21
RemoveObject	0.4
FormGroup	0.32
SplitGroup	0.01

Table 8.2: Classification error distribution model

components to the classification error model:

- Rate - the proportion of features inserted into the observation sequences. A rate of 10% was used for all the experiments in this chapter except where stated otherwise (primarily Section 8.3).
- Distribution model - the distribution of the features selected for insertion. To facilitate comparison of results with the video based experiments the distribution model was configured to be similar to the video event recognition modules. This distribution is summarised in Table 8.2.

Chapter 4 identified the two parameters used during bag-of-features inference. The first of these is the true-positive detection rate $P(D|A)$ which is required for each primitive feature, and the second is the number of particles in each filter. As with the classification

Feature Name	Probability of true-positive
EnterAgent	0.96
ExitAgent	0.96
PlaceObject	0.73
RemoveObject	0.5
FormGroup	0.6
SplitGroup	0.99

Table 8.3: Primitive feature true-positive detection rates

error model, the true-positive detection rates were configured to be similar to the video event recognition modules (summarised in Table 8.3). In the majority of experiments (except where stated) 220 particles were used per complex feature. This ensured that the number of particles in a filter scaled evenly with the number of complex features it represented and ensured good state-space representation. Correspondingly, a filter with three complex features would utilise 660 particles. 220 particles was empirically derived as a good trade-off between accuracy and speed in Section 8.3.

In the majority of experiments the results are used to generate the number of true positive (TP), false positive (FP) and false negative (FN) classifications. From these one can derive several evaluation metrics that are commonly used in behaviour recognition research (e.g. [18, 136, 104, 20, 19, 125]):

- *Precision* - The positive prediction rate calculated as:

$$\frac{\text{Number of correct classifications for behaviour}}{\text{Total number of classifications for behaviour}} = \frac{TP}{TP + FP} \quad (8.1)$$

- *Recall* - The true positive rate (sensitivity) calculated as:

$$\frac{\text{Number of correct classifications for behaviour}}{\text{Number of instances of behaviour}} = \frac{TP}{TP + FN} \quad (8.2)$$

- *F-Score* - A weighted average of a classifier's precision and recall with range [0 : 1], where 1 is optimal. It is calculated as:

$$\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (8.3)$$

It is important to highlight that the precision metric defined in 8.1 does not consider the number of false negatives, and thus it is possible for a classifier to achieve a precision of 1 even if 99% of the classifications are false negatives, with the remaining 1% being true-positives. This logic often defies conventional wisdom, which might expect a precision of 0.01 in such a scenario. Taken in isolation the precision defined above can be misleading, and thus its combination with the recall metric (8.2) is essential. Recall specifically focuses on the number of false-negatives a classifier makes while ignoring false-positive classifications.

Using precision and recall facilitates performance comparisons with prior work, and additionally, helps to identify where a classifier's performance is hindered. For example, a classifier with low recall but high precision communicates that:

- The classifier frequently fails to classify instances of a class...
- ...But when classifications are made they are normally correct (true positives)

In this example the metrics might indicate that a trained model has been over trained, or that a classification threshold is too high. This level of analysis can be important when trying to analyse where a classifier is failing, and is also useful for comparing and communicating the performance of a classifier. One of the aims of this research (as outlined in Chapter 1) was to deliver low false-positive rates, and thus the precision metric is particularly useful for evaluating this aspect. In contrast, when the overall performance of a classifier is required the F-Score metric is useful because it encapsulates both precision and recall, and is more aligned with the conventional way in which classifier performance is often considered.

8.1 Recognition Accuracy

Hypothesis: End-of-simulation prediction will be more accurate than in-sequence prediction.

This section evaluates recognition accuracy at two distinct time points; during the simulation (when a behaviour terminates), and at the end of the simulation (post-sequence). The hypothesis is that post-sequence prediction will provide more accurate predictions because all agent behaviours will have been observed in their entirety and thus more information is available to the inference process.

Experimental Design

For each time-point the ground truth will be compared to the most likely multi-agent/behaviour assignments. This will allow each classifier's F-Score to be calculated. Support for and against the hypothesis will then be derived by comparing the two F-Scores produced.

The first set of experiments will consider post-sequence prediction and will be achieved by suspending all predictions until the end of the simulation. This will ensure that all behaviours have been observed in their entirety. In contrast, the second set of experiments will use a prediction criteria to determine when each behaviour terminates. When a termination is detected a collaborator/behaviour prediction will be made, providing earlier recognition than in the first approach.

To ensure that the prediction time was isolated as the only changing variable a single dataset was shared between both sets of experiments and all other parameters (e.g. number of particles, primitive feature true-positive rates) remained constant. The dataset (generated by algorithm 7.2 on page 141) contained a total of 490 behaviour traces, 70 for each behaviour. The simulation algorithm was seeded with a target number of agents of 10, and thus each scenario generated contained 10-20 concurrent agents, each performing a (potentially different) solo behaviour or one half of a multi-agent behaviour. As per the previous description of the algorithm, each behaviour trace is interspersed to provide a single observation sequence containing a mixture of all agent observations. The simulator was seeded with an error level of 0.1 and thus each simulation contained 10% of primitive feature classification errors (distributed per Table 8.2) to represent the errors that low-level classification algorithms would make when processing real sensor data. Because the

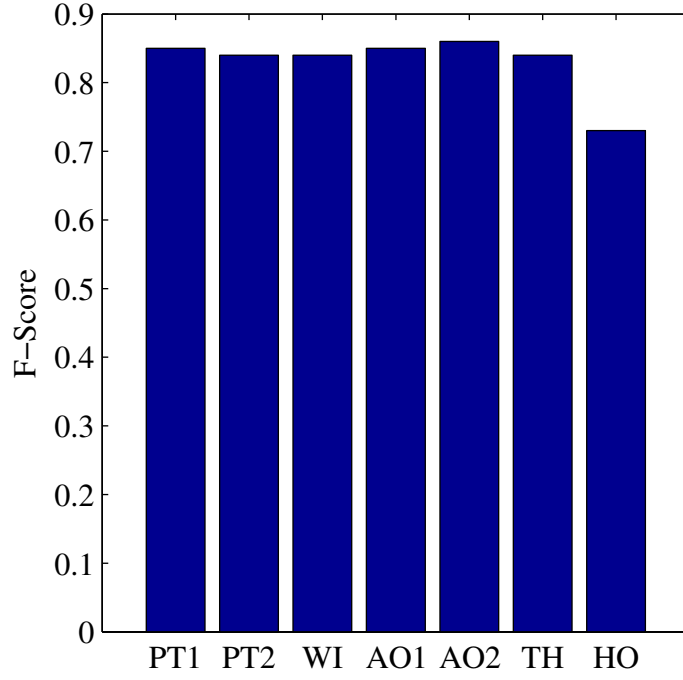


Figure 8.1: F-Score for each behaviour prediction

simulation data contains low-level errors it is not anticipated that perfect recognition will be achieved in either set of experiments, although a high degree of recognition (≥ 0.7) can be reasonably expected.

For each experiment the inference algorithm was executed with 220 particles per complex feature and utilised the primitive feature true-positive rates defined in Table 8.3.

8.1.1 Post-Sequence Prediction

Figure 8.1 shows the F-Score for each behaviour, with the mean F-Score being 0.83. The poorest performer can be clearly seen as the *HO* behaviour, which yields an F-Score of 0.73. This is consistent with the confusion observed in Table 8.4 where all behaviours show instances of confusion with *HO*. Indeed, the recall of *HO* remains relatively high at 0.79, but its F-Score is impacted by a precision of only 0.68. This is substantially lower than all of the other behaviours, where the next lowest precision is 0.8 (*AO1*).

Table 8.4 shows the confusion matrix for the seven behaviours. Each column represents instances of a predicted behaviour, while each row represents the instances of an actual

	<i>PT1</i>	<i>PT2</i>	<i>WI</i>	<i>AO1</i>	<i>AO2</i>	<i>TH</i>	<i>HO</i>
<i>PT1</i>	0.74	0.06	0	0.04	0.01	0	0.14
<i>PT2</i>	0	0.89	0	0.04	0.01	0	0.06
<i>WI</i>	0	0	0.77	0.09	0.02	0.08	0.05
<i>AO1</i>	0	0	0.01	0.91	0	0.01	0.06
<i>AO2</i>	0	0.04	0.01	0.04	0.89	0	0.01
<i>TH</i>	0	0.01	0.01	0.01	0.09	0.81	0.06
<i>HO</i>	0	0.10	0.01	0	0.06	0.03	0.79

Table 8.4: Behaviour confusion matrix for post-sequence prediction

behaviour. The top left entry shows that *PT1* is correctly classified as *PT1* in 74% of cases, while the top right entry shows that *PT1* is misclassified as *HO* in 14% of cases. A cursory scan of the table will show that *PT1* is in fact the most misclassified behaviour while *AO1* is the least.

8.1.2 In-Sequence Prediction

Figure 8.2 compares the F-Score for each post-sequence behaviour prediction against the in-sequence predictions. The most striking result in this figure is that the F-Score increases by as much as 0.2 (*HO*), and shows a minimum improvement of 0.05 (*AO2/TH*). Correspondingly, the average F-Score increases from 0.83 to 0.92, a 9% improvement. Table 8.5 shows that behaviour confusion is much smaller for in-sequence prediction with *WI* and *HO* being the most confused behaviours (7% confusion). On the other hand, *PT1* is no longer confused with any behaviour, while it was the most commonly confused during post-sequence prediction. The mean level of confusion for any behaviour is only 4%.

These results suggest that the hypothesis should be rejected because post-sequence prediction does not deliver a higher level of recognition performance than in-sequence prediction. Investigating the reason for these unexpected results identified that low-level classification errors caused incorrect predictions to be made during post-sequence prediction. To illustrate why this occurs Figure 8.1.2 shows three hypothetical behaviours and an observation trace for two agents. Behaviours 1 and 3 are both solo behaviours while behaviour 2 is a multi-agent behaviour involving two actions for one agent and one multi-agent feature performed by both agents (feature: *E1,2*).

In the observation traces Agent 1 is performing behaviour 1, but note that two classification errors also occur: *F1* and *E1,2*. Agent 2 is performing behaviour 3, however, their

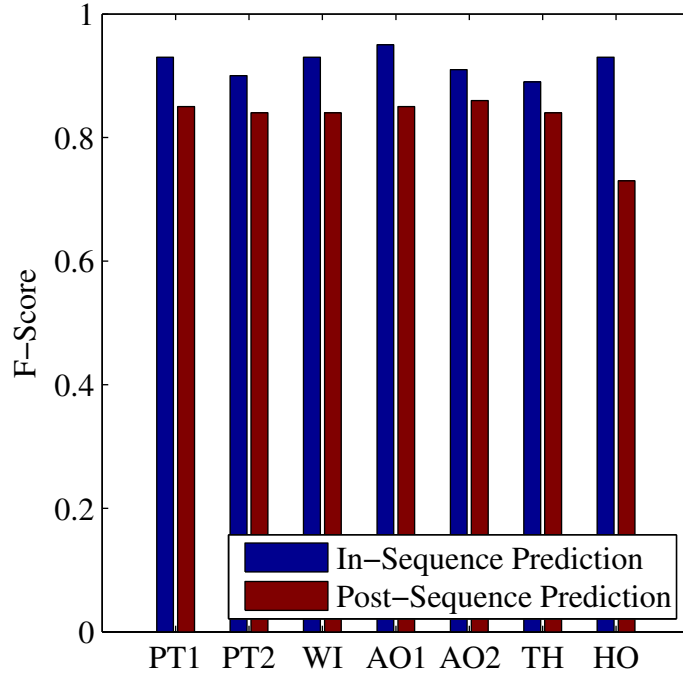


Figure 8.2: In-sequence prediction F-Score vs. post-sequence F-Score

observation trace also contains an erroneously detected $E_{1,2}$ feature. In such an example in-sequence prediction would detect that Agent 1 completes Behaviour 1 after observing $B : 1$, although the observance of $F : 1$ is likely to reduce the probability of $B : 1$ over observing the behaviour without any classification errors. Similarly, Behaviour 3 would be detected for Agent 2 after observing feature $D : 1$. No other behaviours are performed in their entirety so in-sequence prediction would make a total of two predictions, both of which would be correct.

In contrast, post-sequence prediction would include the $E_{1,2}$ classification error when determining the most likely behaviour for each agent. Thus, even though Agent 1 and 2 have performed complete behaviours, if the joint probability for all of their actions yields a higher likelihood for joint behaviour 2 then this will be predicted for both agents. In this example the presence of the additional $F1$ classification error means that the probability of behaviour 1 for agent 1 is lower than it would be otherwise, making it more likely for behaviour 2 to yield a higher probability. Thus, the presence of classification errors (which could in theory occur even after the agent has left the scene) means that the most likely behaviour can change. However, the prediction criteria used by in-sequence prediction helps to filter out the effect of these classification errors, thus delivering improved performance. Going forwards, the rest of the results will report in-sequence predictions.

	<i>PT1</i>	<i>PT2</i>	<i>WI</i>	<i>AO1</i>	<i>AO2</i>	<i>TH</i>	<i>HO</i>
<i>PT1</i>	1	0	0	0	0	0	0
<i>PT2</i>	0	0.96	0	0	0.04	0	0
<i>WI</i>	0	0.04	0.93	0.03	0	0	0
<i>AO1</i>	0	0	0	0.97	0.03	0	0
<i>AO2</i>	0	0.03	0	0	0.97	0	0
<i>TH</i>	0	0	0	0	0.04	0.96	0
<i>HO</i>	0	0	0	0	0.07	0	0.93

Table 8.5: Behaviour confusion matrix for in-sequence prediction

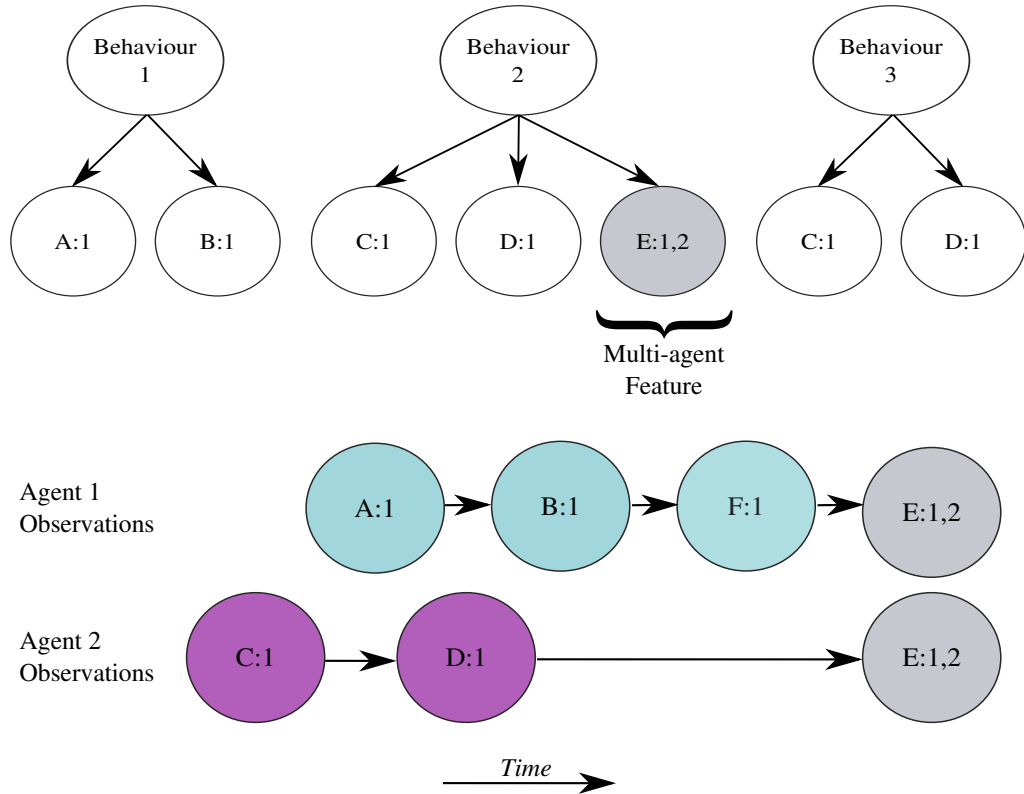


Figure 8.3: Example observation trace in which an erroneous multi-agent feature (E1,2) occurs.

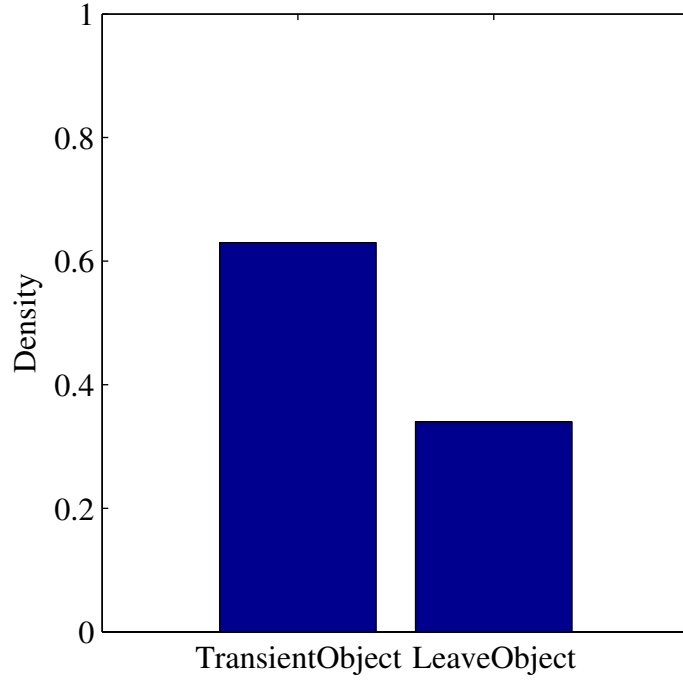


Figure 8.4: The density of particle weights associated with two explanations (complex features) for the set of primitives: $\{PlaceObject, RemoveObject\}$. One third of the particle weights are assigned to the assumption that *RemoveObject* was a false positive and explain the *PlaceObject* primitive with a *LeaveObject* complex. The remaining two thirds of particle weight assume that both observations were true positives and explain them via the complex feature *TransientObject*.

The event detail extracted during post-sequence predictions were found to be optimal. That is, by selecting explanations using a weight density threshold (DT) of 0.5 the filters always identified the correct complex features during explanation generation. For example, Figure 8.4 shows the density of particle weights associated with two explanations for the set of primitives $\{PlaceObject, RemoveObject\}$. Because $P(TransientObject) > DT$ it is selected as the most likely explanation and those particles are used to identify the event times. The explanations that were generated for behaviour predictions contained the correct event times in all of the cases examined.

8.2 Behaviour Variability

This section is comprised of five experiments which consider the effect of behaviour variability on recognition performance. In this context variability refers to behaviours that are different from those defined and include ones that are entirely different as well as small mutations of known behaviours. Each experiment focuses on a particular hypothesis as defined below:

1. Unknown behaviours are not misclassified
2. Behaviours can be recognised when their features are observed in a different order to the behaviour definition
3. When two concatenated behaviours are observed each behaviour can be recognised
4. When one (partial) behaviour is switched to another (partial behaviour) the second behaviour can be recognised
5. Behaviours can be recognised in the presence of superfluous (repeated) features

8.2.1 Unknown Behaviours

Hypothesis: Unknown behaviours are not misclassified

In contrast to Section 8.1, which showed that known behaviours could be recognised, it is hypothesised that unknown behaviours are not misclassified (no prediction is made). An unknown behaviour is any sequence of activities that is vastly different from those defined.

Experimental Design

To test this hypothesis a set of unknown behaviours were required on which inference could be performed. The null hypothesis gains support if the inference algorithm misclassifies the behaviours as known, while a lack of classifications supports the hypothesis. To provide observation traces containing unknown behaviours a modified version of the

primitive feature simulator was used. This modified algorithm can be found in listing 8.1 on page 174.

The first difference between this algorithm and the original is that the set of all complex features is added as a new parameter (*AllC*). The remaining parameters are as defined before: *ER*=Error rate, *EM*=Error model, *N*=Number of agents, *AllB*=Set of all behaviours. The algorithm also contains a new *Step 0*, which generates a new (random) behaviour and inserts it into the set of all behaviours. This random behaviour is used to represent a novel (unknown) behaviour, with its length determined at random to be between two and eight complex features in length (line 5).

It is important to note that a complex feature may require multiple agents. As in Chapter 6, agent roles are used to distinguish between different agents within a behaviour. A solo behaviour involves one role while a two agent multi-agent behaviour will involve two roles. It should be highlighted that line 8 chooses a random role from those generated, and thus a multi-agent behaviour can be generated containing both solo and multi-agent features for both agents. At the end of *Step 0* the new behaviour is added to the set of all behaviours and can be selected for simulation during *Step 1* of the algorithm. *Steps 1-3* of the algorithm are the same as in the original (see page 141).

To isolate the experimental variables the same inference parameters and dataset are used for all generated behaviour traces. Using algorithm 8.1 to generate the test dataset a total of 240 behaviour traces were produced. This consisted of 30 traces for each behaviour (including 30 unknown behaviours). As before, the simulation algorithm was seeded with a target number of agents of 10, and thus each scenario generated contained 10-20 concurrent agents, each performing a (potentially different) solo behaviour or one half of a multi-agent behaviour. As with the original algorithm, each behaviour trace was interspersed to provide a single observation sequence containing a mixture of all agent observations. The simulator was also seeded with an error level of 0.1, so each simulation contained 10% of primitive feature classification errors (distributed per Table 8.2).

As before, inference on each scenario was performed using 220 particles per complex feature and utilised the primitive feature true-positive rates defined in Table 8.3. The inference algorithm was not aware of the random behaviours generated during the dataset generation process.

To evaluate the hypothesis the results of the experiment will be compared against those in Section 8.1.2, in which only known behaviours were present. Because the simulation data contains low-level errors it is not anticipated that perfect recognition will be achieved for

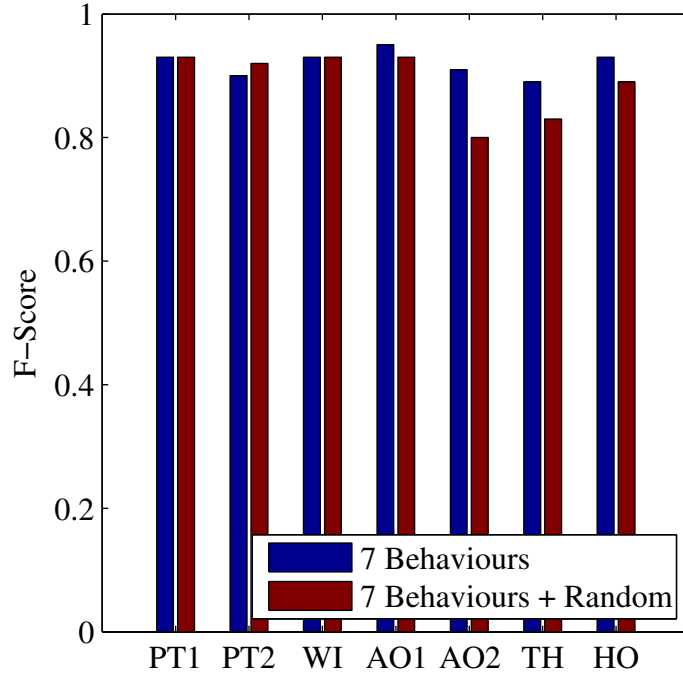


Figure 8.5: Comparing F-Scores when unknown behaviours are observed. Unknown behaviours have minimal impact on recognition performance. Of the 30 unknown (random) behaviours only 4 (13%) were misclassified, with no classifications made for the remaining 87%.

the **known** behaviours, although it is anticipated that performance will be aligned with the results of Section 8.1.2 if the hypothesis is correct. If the hypothesis is incorrect this will be evidenced by lower recognition F-Scores, caused by the misclassification of the unknown behaviours.

Results

Figure 8.5 compares the recognition F-Scores for each behaviour with those obtained in Section 8.1.2. One can see that there are no major differences in the results although *AO2* and *TH* both show a slightly more significant drop in performance over the other behaviours. Further analysis shows that these differences are a feature of the dataset rather than the influence of misclassified ‘unknown’ behaviours. The average F-Score for the known behaviours was 0.89, representing a 3% drop in performance over observing known behaviours only. These results provide a small amount of support the null hypothesis because 13% of the unknown behaviours *are* misclassified. However, one can see that only a small proportion of misclassification occur.

8.2.2 Ordering Mutation

Hypothesis: Behaviours can be recognised when their features are observed in a different order to the behaviour definition

It is hypothesised that recognition of a behaviour will be unaffected by small changes to the primitive feature ordering during observation.

Experimental Design

To test this hypothesis requires a comparison of recognition performance under two conditions:

- Observations matching the defined behaviour order
- Observation with small changes to the primitive feature ordering.

Similar inference recognition performance under both conditions will provide support for the hypothesis, while a significant difference in performance will support the null hypothesis.

Recall that a complex feature represents a behaviour sub-goal as a sequence of contiguous primitives. For example, the complex feature *LeaveObject* is modelled by the contiguous primitives $\{PlaceObject, ExitAgent\}$. The validation behaviours contain four complex features with two or more primitives: *LeaveObject*, *TransientObject*, *PersonInteraction* and *StealObject*. Of these only *PersonInteraction* and *LeaveObject* exist in scenarios that allow the contiguous order to be interrupted and thus this experiment evaluates order mutations on the *WI* and *AO1* behaviours only. Recognition performance of the other five behaviours will be ignored in this experiment, although instances of these behaviours were present in the scenarios in order to ensure that the test conditions were consistent with the other experiments.

For this experiment a new dataset was generated in which the ordering of the *WI* and *AO1* primitives was manipulated. This manipulation was done via an interactive tool that allowed a specific observation ordering to be constructed. Figure 8.6 shows the pro-

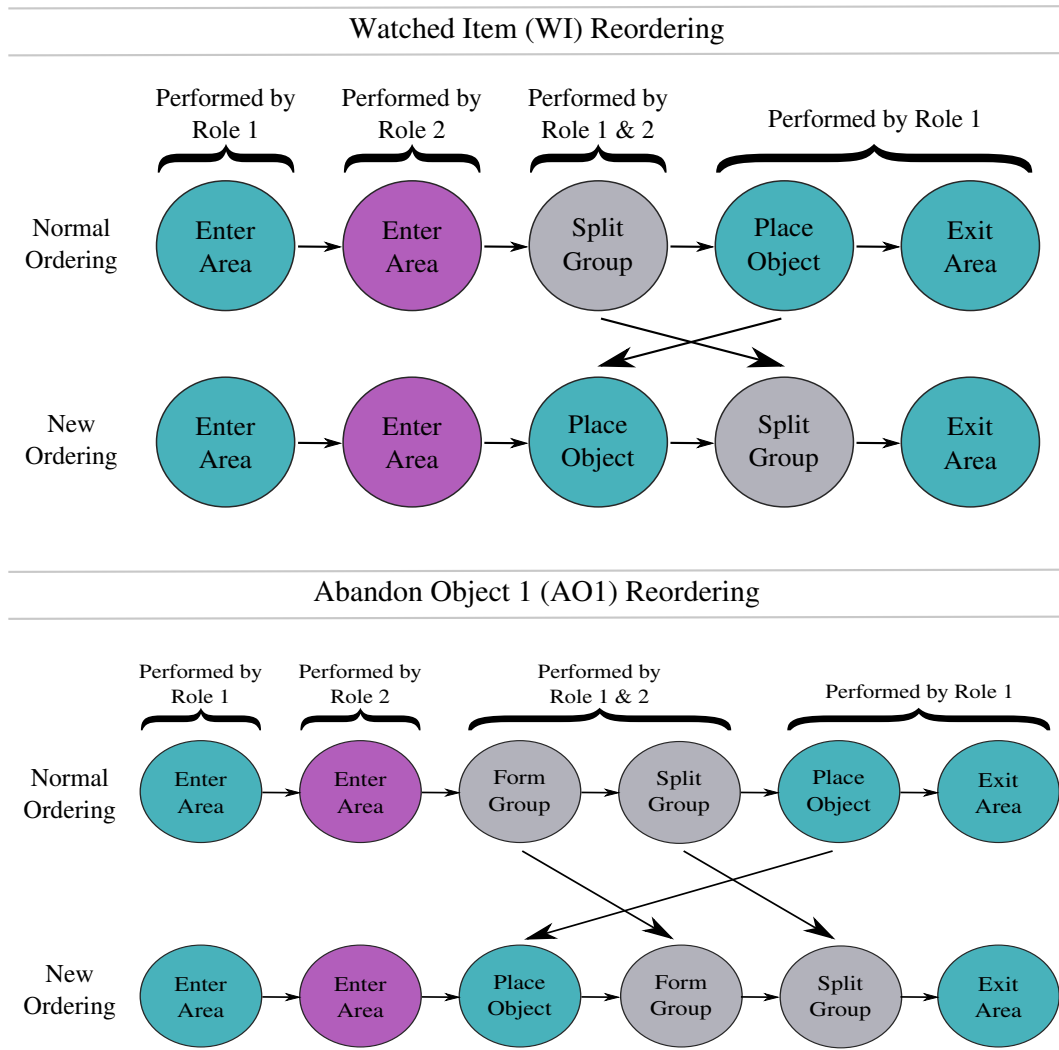


Figure 8.6: The primitive features in the *WI* and *AO1* behaviours being reordered

cesses that were performed during this operation, while all other behaviours remained unchanged. For the purpose of this experiment these new orderings replaced the standard orderings produced by the primitive feature simulator, although it should be highlighted that the final sequential ordering of observations were interspersed between different agents as with the other experiments. The simulator was seeded with a target number of agents of 4, and thus each scenario generated contained 4 – 8 concurrent agents. An error level of 0.1 was also used, in addition to the standard classification error distribution (Table 8.2). The resulting data contained 24 instances of each behaviour giving a total of 168 behaviour traces. Inference was performed using 220 particles per complex feature and utilised the primitive feature true-positive rates defined in Table 8.3.

Results

The experiments showed that interrupting contiguous behaviours had no impact on overall behaviour recognition performance for the *AO1* and *WI* behaviours, but did effect the probability distributions. This is shown in Figures 8.7 and 8.8, where example probabilities for each complex feature for *WI* and *AO1* sequences are presented. In Figure 8.7a the observation order is the same as that modelled, while part (b) shows a re-ordering of the two *LeaveObject* components (*PlaceObject*, *ExitAgent*). The x-axis represents each observed feature with time increasing from left to right, while the y-axis shows the probability for each complex feature. Looking at the last column in the graphs one can see that there is relatively little difference in the likelihood of the *LeaveObject* complex, although there is a slightly broader distribution of probability in part (b) where the modelled order was disrupted by **one** observation (*SplitGroup*).

Similarly, Figure 8.8 shows the probability of each complex feature as *AO1* sequences are observed. Again, part (a) shows observations in the modelled order, while part (b) shows a mutated order, where this time, *LeaveObject* is disrupted by **two** observations (*FormGroup*, *SplitGroup*). Again, the last bar shows that the mutation only causes a small change in the probability of the *LeaveObject* complex feature with a slight broadening of the distribution to include other complex features. These results support the hypothesis that behaviours can be recognised when their features are observed in a different order to the behaviour definition. However, it should also be acknowledged that this support is restricted to the *AO1* and *WI* behaviours included in the experiment, although it may be reasonably expected that additional behaviours would offer similar support.

8.2.3 Behaviour Switching and Concatenation

Hypothesis 1: When two concatenated behaviours are observed each behaviour can be recognised

The inference algorithm has been designed such that when a behaviour terminates via completion (all primitives have been observed), a new behaviour can be initialised. This means that if two concatenated behaviours are observed, both components should be correctly recognised and predicted by the inference algorithm.

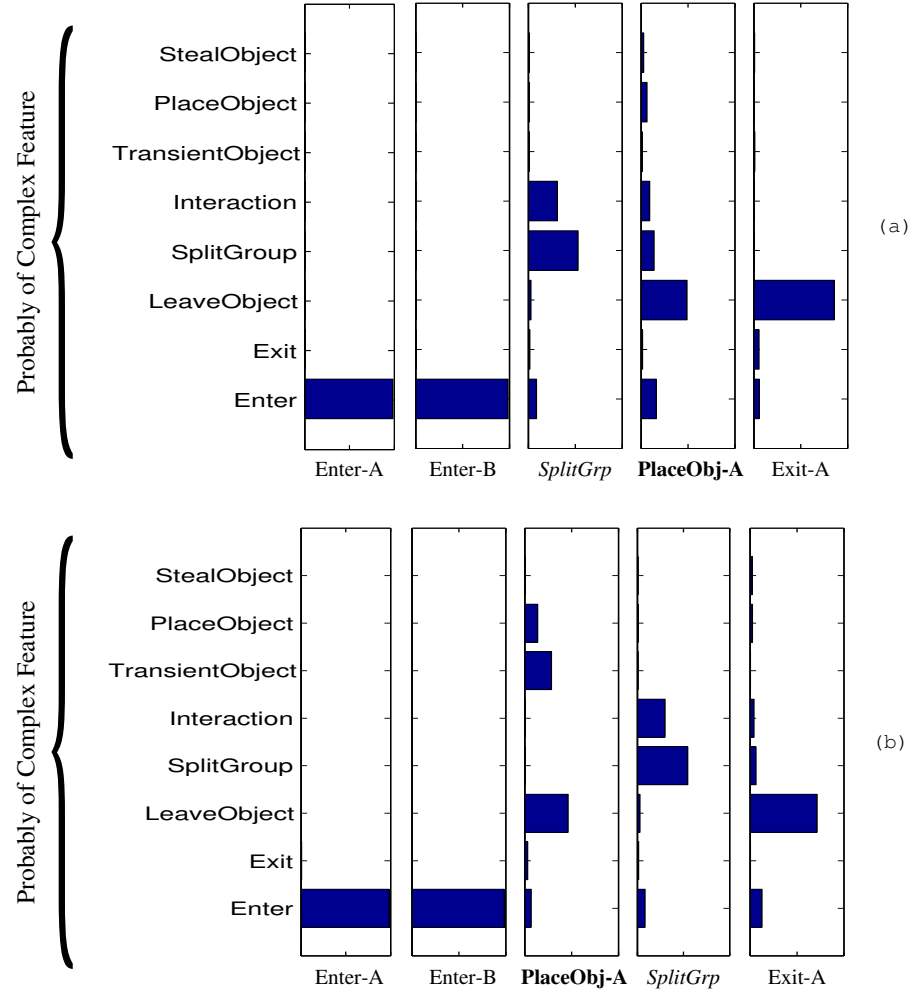


Figure 8.7: Probability density for each complex feature as observations are made (Behaviour: WI). (a) is consistent with the model order, (b) shows an alternative order. When *PlaceObject* and *ExitAgent* are observed contiguously the *LeaveObject* feature (which is comprised of *PlaceObject* and *ExitAgent*) has a higher probability density than when they are separated by another observation.

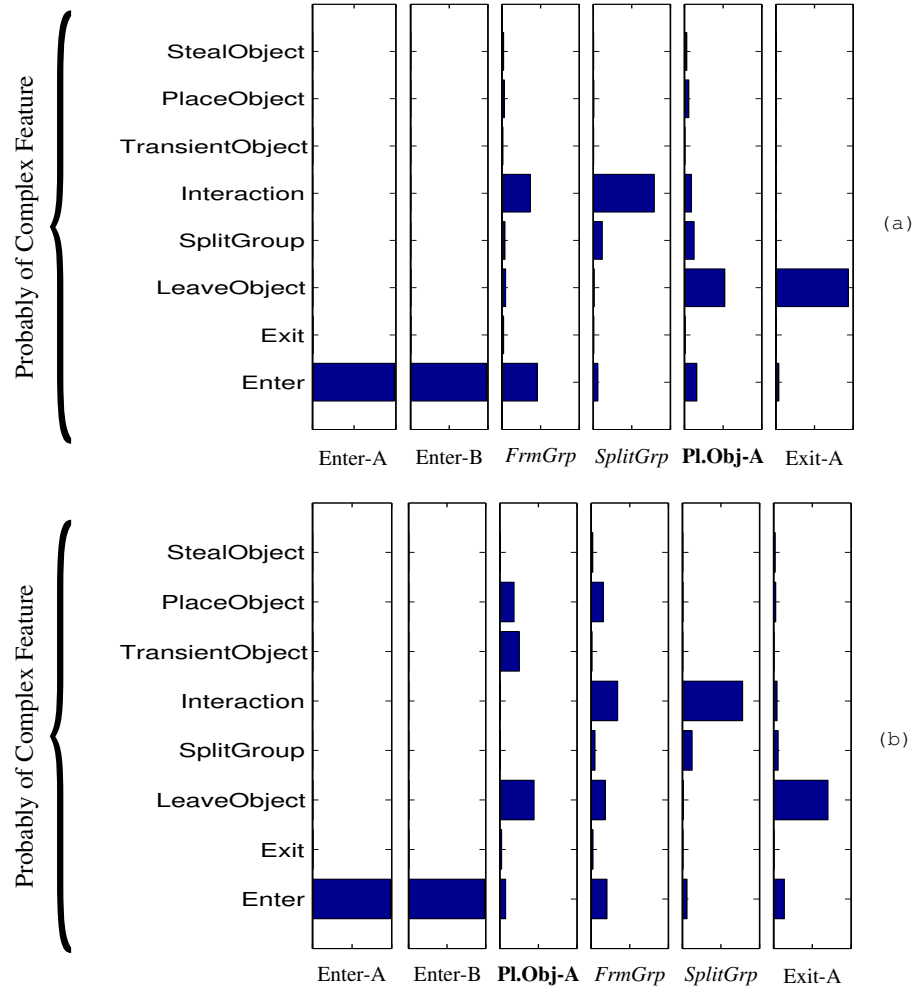


Figure 8.8: Probability density for each complex feature as observations are made (Behaviour: AO1). (a) is consistent with the model order, (b) shows an alternative order. When *PlaceObject* and *ExitAgent* are observed contiguously the *LeaveObject* feature (which is comprised of *PlaceObject* and *ExitAgent*) has a higher probability density than when they are separated by **two** observations.

Experimental Design 1:

This hypothesis can be tested by performing inference on a dataset in which both concatenated and un-concatenated behaviours exist. By comparing recognition performance of the second (concatenated) behaviour against scenarios in which only un-concatenated scenarios are present it will be possible to determine whether concatenated behaviours can be recognised, and if so, how this recognition compares to un-concatenated scenarios. Ensuring that concatenation is the only non-static variable is also a necessary step, and thus, in addition to using the same inference parameters, each combination of behaviours must be considered.

To perform this experiment a new dataset was generated using an interactive tool in combination with Algorithm 7.2. The tool allowed individual behaviours to be selected for concatenation and produced a single (concatenated) behaviour trace. Additional behaviours were then generated using Algorithm 7.2 with an error rate of 0. The output of each process was then mixed and noise added to give a final output scenario. This process can be summarised by Algorithm 8.2 on page 175.

As with earlier experiments an error level of 0.1 was used in addition to the standard error distribution model. Similarly, the simulation was seeded with a target number of agents of 10. The resulting dataset consisted of 770 observation sequences where each individual behaviour was represented 54 times in each position (first/second).

The anticipated result of the experiment was that the recognition F-Score of the first behaviour classifier was comparable with when the behaviour was not concatenated, while the second behaviour classifier should achieve a lower F-Score. This expectation is due to the sampling nature of the inference algorithm.

Results 1:

Figure 8.9 shows that when concatenating behaviours the detection F-Score of the second behaviour is lower (0.73) than the first (0.92). This represents a 19% drop in performance but supports the hypothesis that the second behaviour in a concatenated scenario can be recognised.

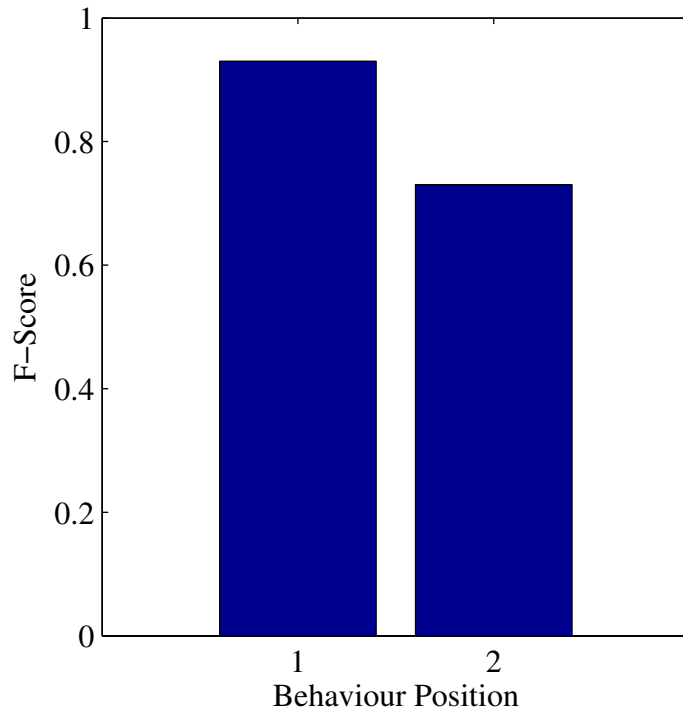


Figure 8.9: When two behaviours are observed directly after one another the second behaviour suffers a 24% loss of recall. In combination with a 4% drop in precision this leads to an overall F-Score of 0.73.

Hypothesis 2: When one (partial) behaviour is switched to another (partial behaviour) the second behaviour can be recognised

The second hypothesis is based on the convergence property of the inference process. At the point at which the first behaviour is abandoned a high proportion of particles will be tracking this behaviour, but through particle reset, these particles should converge to the second behaviour.

Experimental Design 2:

This hypothesis can be tested by performing inference on behaviour traces containing switched behaviours. If, at the end of a switched behaviour trace all features of the second behaviour have been observed (through a union of the first and second behaviours), then the behaviour should be recognised. Therefore, this hypothesis can be tested generating data containing switched behaviours, in addition to standard behaviours, and comparing recognition performance of the final behaviour in each switched scenario with the recognition performance of each behaviour being performed in an isolated scenario (not switched).

The approach taken for generating such a dataset closely mimics Algorithm 8.2, with a key difference being that on line 9 a check is inserted to ensure that the behaviours are not the same, and do not represent behaviour *PT1*. The *PT1* behaviour is excluded from this experiment because of its complete overlap with all other behaviours, while we ensure that the behaviours are different because the goal is switch from one behaviour to another. The second change to the algorithm is a replacement of the function call *GetConcatTrace* with a call to the function *GetSwitchedTrace*(*behaviour1*, *behaviour2*). This function generates a partial observation sequence of *behaviour1* followed by *behaviour2*, where the termination point of *behaviour1* is randomly selected, as is the beginning of *behaviour2*. This resulting simulation algorithm is summarised in Algorithm 8.3 on page 176.

For simulation an error level of 0.1 was used, in addition to the standard error distribution model and a target number of agents of 10. The output dataset consisted of 240 observation sequences where each behaviour (except *PT1*) was represented by 40 examples.

Results 2:

When observing switched behaviours the first behaviour is never completed and thus the prediction criteria are never met. Correspondingly, the first behaviour is never predicted. However, the second behaviour is completed and a prediction is made. An F-Score of 0.68 is achieved for recognition of the second behaviour, representing a 24% decrease in performance over observing a standard single behaviour. Although recognition is degraded over the base case, an F-Score of 0.68 does provide support for hypothesis 2.

8.2.4 Superfluous Activity

The final experiment in this section considers the effect of superfluous activities on prediction performance. Superfluous activities do not contribute towards the achievement of a goal behaviour but cannot be considered classification errors because the activities are correctly detected. An example might be an agent who places and removes an object several times before abandoning it, where the placement and removal are not considered as contributing to the abandoning object goal.

Hypothesis: Behaviours can be recognised in the presence of superfluous (repeated) features

The inference algorithm has been designed such that when a particle is reset, reversed features are removed from the set of observed features. This means that the algorithm should be able to recognise behaviours with repeated components.

Experimental Design:

To test this hypothesis superfluous features were inserted into all behaviour scenarios except PT1, which was excluded from the experiment. Specifically, the simulation algorithm was updated to insert contiguous instances of *{PlaceObject and RemoveObject}*, and *{FormGroup and SplitGroup}* into each behaviour trace. 50% of instances involved single pair insertion, while the remaining 50% involved two pair insertion. This version of the simulation process is summarised in Algorithm 8.4 on page 177.

As in the previous experiments, an error level of 0.1 was used, in addition to the standard error distribution model and a target number of agents of 10. The resulting dataset consisted of 252 behaviour traces in which each behaviour was represented by 36 examples. The inference algorithm was configured with the standard parameters.

By comparing the recognition performance of these behaviours with their recognition performance under standard conditions (Section 8.1.2) support for or against the hypothesis will be obtained.

Results:

The results in Figure 8.10 show that accuracy is badly effected by the additional features with the lowest F-Score being 0.39. However, there is also a large range in F-Scores, with the highest performer (AO1) still achieving an accuracy of 0.86. The average F-Score is 0.64, which is significantly lower than the performance obtained in Section 8.1.2 (0.92). Nevertheless, the results provide some support the hypothesis that behaviours can be recognised in the presence of superfluous features, albeit with a negative impact on performance.

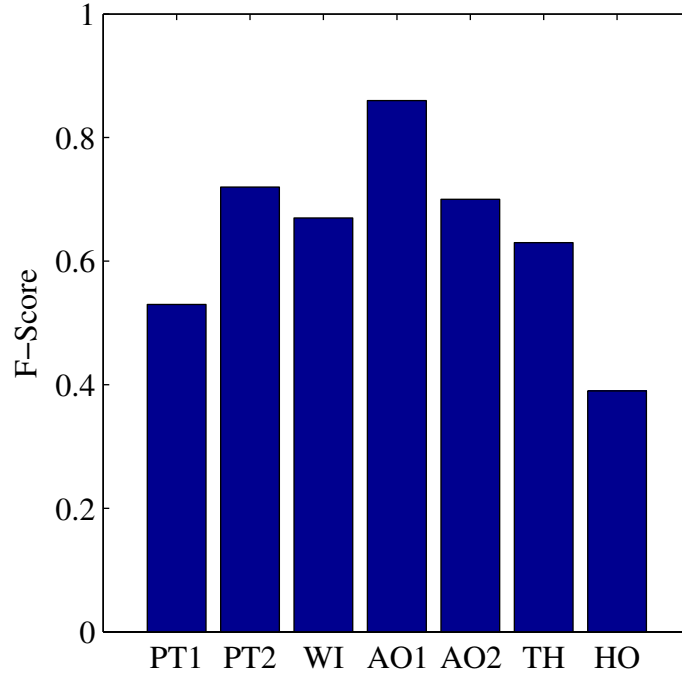


Figure 8.10: Pairs of superfluous activities (e.g. $\{PlaceObject, RemoveObject\}$) cause a reduction in mean F-Score from 0.92 to 0.64.

8.3 Effect of Feature Classification Errors

This section studies the impact of classification errors on recognition performance. To recap, classification errors represent erroneous primitive feature observations and are caused by sensor noise and lower-level classification mechanisms. In this implementation those classification mechanisms are found within the image processing layer of the video surveillance framework (Chapter 7). When a particle in the particle filtering algorithm cannot explain an observation, low-level classification errors can be the cause, and the particle will assume that the detection is erroneous with a probability equal to the observation false-positive classification rate.

Hypothesis: Recognition performance should reduce gradually in the presence of increasing low-level classification errors.

When a particle assumes that an observation has been caused by a classification error that observation is ignored. This allows a subset of particles to filter classification errors and should facilitate their recognition of the correct underlying behaviour. Because this filtering of errors is an approximation achieved via sampling it is hypothesised that overall

behaviour recognition performance will gradually reduce as the level of input errors is increased. The null hypothesis is that performance will not reduce gradually.

Experimental Design:

To test this hypothesis the algorithm's performance was evaluated under five different conditions. For each condition a different, fixed level of classification errors were present in the observation stream, while all other algorithm parameters remained static. Five datasets were generated using Algorithm 7.2 on page 141, where each dataset used the same parameter values for the error model (Table 8.2), number of agents (10) and set of all behaviours (Table 8.1). However, each dataset used a different value for the error rate, where the first dataset used a value of 0, the second a value of 0.1, the third a value of 0.2 and so forth until the fifth dataset with an error rate of 0.4. Each dataset was comprised of 490 behaviour traces in which each behaviour was represented 70 times.

To isolate the error level as the only moving variable the inference algorithm was executed using the same parameters for all five datasets. That is: 220 particles per complex feature, and the primitive feature true-positive rates defined in Table 8.3. Under these conditions one would expect the inference algorithm to recognise behaviours with an F-Score of close to 1 with the dataset containing zero noise, and the lowest F-Score to be achieved with the 0.4 error level dataset.

Results:

Figure 8.11 shows support for the hypothesis. A recognition F-Score of 0.97 is achieved at zero classification errors, and drops by approximately 0.1 for every 10% increase in input classification error. It is encouraging that the accuracy reduces reasonably linearly although it would be difficult to estimate performance for higher levels of error due to the change in gradient between the 0.3 – 0.4 levels.

As anticipated, an F-Score close to 1 was achieved under zero noise. This is a reasonable performance, and it is hypothesised that perfect recognition was not achieved because the number of particles used was insufficient. This hypothesis will be considered further in the next section when the number of particles will be altered in isolation and tested against some of the same datasets as above.

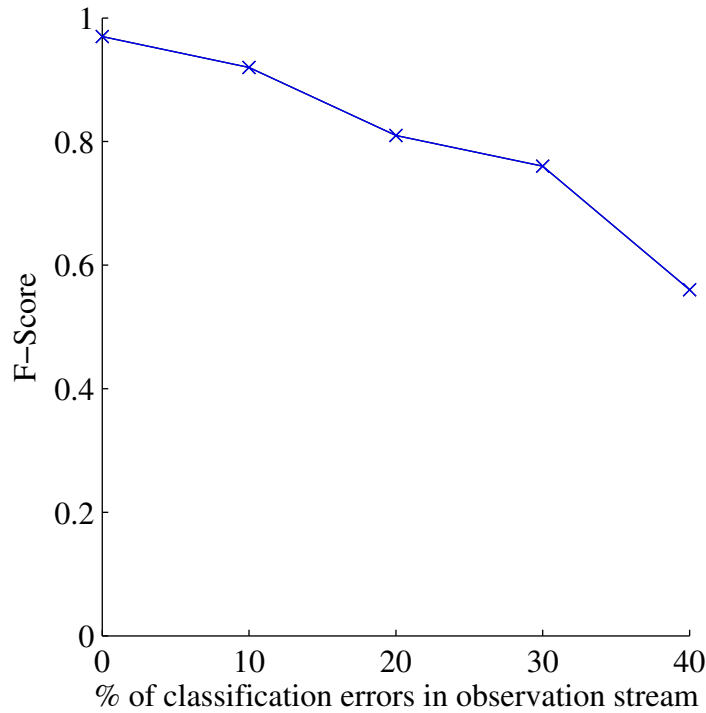


Figure 8.11: Effect of primitive feature classification errors on recognition F-Score [update to use levels]

8.4 Effects of Scaling

To help identify limitations of the approach the effects of scaling were analysed. There are primarily two factors that can be scaled: the number of particles in each filter, and the number of agents concurrently monitored. The number of particles in each filter is interesting because as this number increases, filter accuracy should improve at the cost of increased runtime. It is important to understand how effective the filters are under different conditions so that the accuracy/speed trade-off can be appropriately judged. Furthermore, in the previous section it was observed that an F-Score below 1 was obtained when no classification errors were present in the observations, and it was hypothesised that this was caused by inference with an insufficient number of particles. Section 8.4.1 will test this hypothesis by evaluating performance with the same dataset, but this time altering the number of particles.

The number of concurrently monitored agents is also important for identifying runtime limitations. Chapter 6 identified that the algorithm for identifying multi-agent behaviour is quadratic in the number of agents, a factor that will be analysed in detail in this section. The number of agents that inference can accommodate is a very important factor in identifying limitations with the approach.

It should be noted that this section does not analyse the number of behaviours being filtered. This is because behaviour confusion is dependent upon the similarity of behaviours. One cannot simply evaluate the approach with increasing numbers of behaviours without also controlling their similarity. The validation behaviours being used are meaningfully defined and exhibit non-uniform similarity. Although behaviour similarity could be controlled by defining a theoretical set of behaviours, such an experiment would still be of limited utility and thus such an experiment has not been performed.

8.4.1 Number of Particles

Hypothesis 1: Behaviour recognition performance will decrease as the number of particles is reduced

Particle filtering theory informs us that as the number of particles is increased, the associated probability density estimates approach the true values [9]. Given that behaviour recognition is based on these probability estimates it is hypothesised that recognition performance will decrease as the number of particles is reduced.

Hypothesis 2: Increasing the number of particles sufficiently will allow a recognition F-Score of 1 to be achieved when observing behaviours with zero input classification errors

If there are zero classification errors in the behaviour observations it is hypothesised that the algorithm will be able to achieve an F-Score of 1 if the number of particles is sufficiently high to model the state-space.

Experimental Design:

To test these hypotheses two of the datasets from Section 8.3 will be used:

- Dataset 1: Behaviour observation traces containing zero classification errors
- Dataset 2: Behaviour observation traces containing errors at a rate of 0.1.

These datasets will be used without modification. However, in contrast to Section 8.3 the experiment will alter the number of particles per complex feature. As before, Table 8.3 provides the primitive feature true-positive rates used during inference.

The experiment consists of two scenarios, one for each dataset. For both datasets it is anticipated that reducing the number of particles will decrease recognition performance and such a result will provide support for hypothesis 1. Section 8.3 showed that as the level of input classification errors was increased, behaviour recognition performance decreased. It is therefore expected that if the recognition performances for datasets 1 and 2 are compared for a given number of particles, the results with dataset two will always be below those achieved with dataset 1. Based on the results from the previous section it is also anticipated that a recognition F-Score ≥ 0.92 will be achieved when the number of particles per complex feature is higher than 220 and zero classification errors are observed. Such a result will provide support for hypothesis 2.

Results:

Figure 8.12 shows the effect of the number of particles on the behaviour recognition F-Score. The results support hypothesis 1 because as the number of particles reduces, so too does recognition performance. Furthermore, one can see that as expected, performance with dataset 2 is lower than dataset 1 due to the presence of classification errors in the input and is consistent with Section 8.3.

When there are no classification errors in the input observations the input can be considered ‘best-case’ conditions. It is encouraging to see that as the number of particles is raised from 10 – 100 recognition accuracy quickly improves to over 0.9. By 200 particles accuracy reaches 0.95, and then continues to converge towards 1 as N is further increased. An F-Score of 1 is achieved with 300 particles, supporting hypothesis 2. The second result in Figure 8.12 shows recognition accuracy with an input error level of 0.1. This error level is similar to that produced when applying the video event detectors from Chapter 7 to the video datasets. Again, the recognition accuracy increases sharply as the number of particles is increased, but plateaus at approximately 0.9 accuracy.

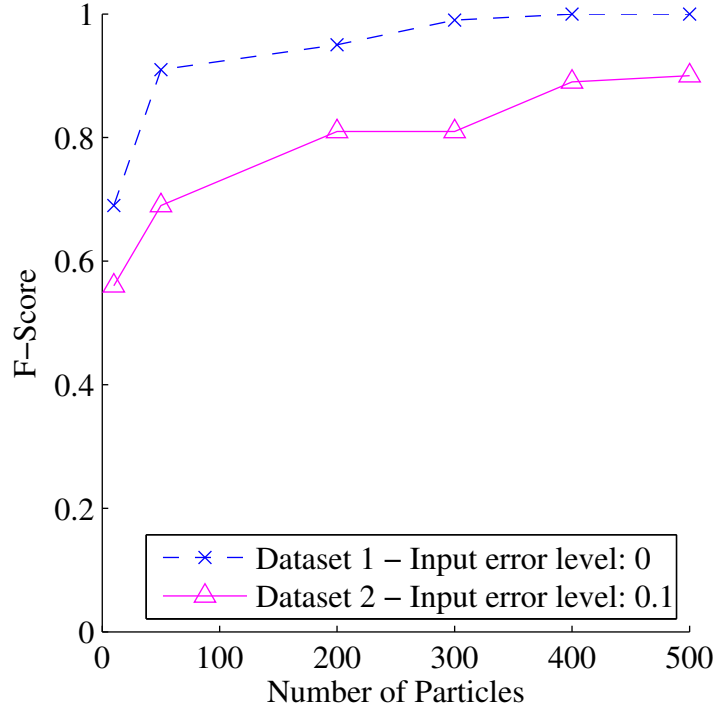


Figure 8.12: Effect of number of particles on F-Score

Hypothesis 3: For any given observation the runtime of the approach should be linear with the number of particles

If the number of particles is isolated as the only variable, the runtime of the algorithm should change linearly with the number of particles. As the number of particles used is increased, so too should the runtime.

Experimental Design:

To test this hypothesis requires that all variables are held static except the number of particles used. This is difficult to achieve because the number of agents in the inference process is dependent on the proportion of the observation sequence that has been observed. That is, if an observation sequence of length S contains six agents and is partially observed by time t (where $t < S$), but only three agents have been observed by t , then only three agents will be involved at the inference process. However, at the final observation ($t = S$) all six agents will have been observed and the inference process will therefore involve six agents. Because the number of particle filters directly correlates with the number of agents upon which inference is being performed, these two time-steps will involve different numbers of filters.

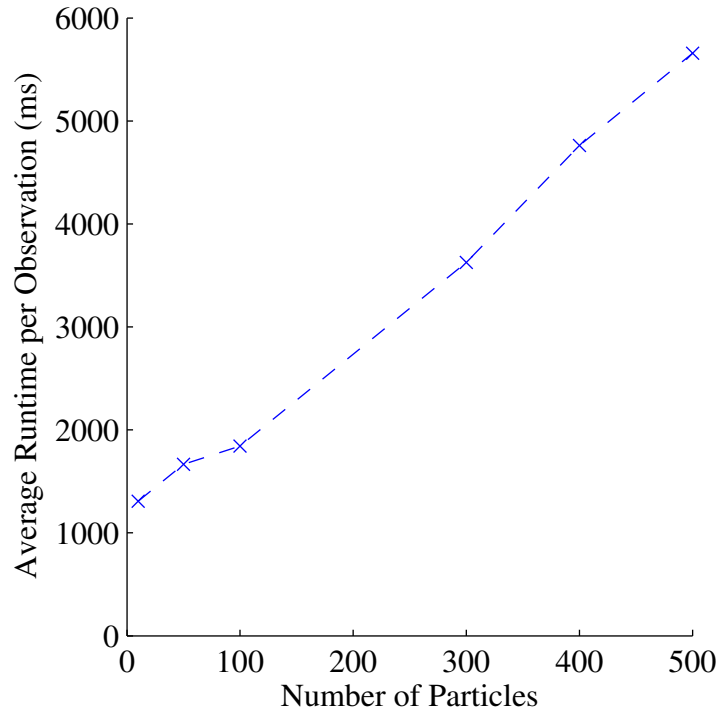


Figure 8.13: Effect of number of particles on runtime

To solve this problem a dataset containing six agents was generated, however, the mean runtime of the processing time for each sequence is used as an approximation of the runtime for each observation. It is acknowledged that this cannot be used as an accurate estimate of runtime for each observation, however, it can be used as a comparative value when altering the number of particles. By performing inference on the same dataset multiple times, but using different quantities of particles, one can compare the mean run-times whilst all other variables remain static.

To generate the dataset Algorithm 7.2 on page 141 was used with an error level of 0.1, the error model from Table 8.2, six agents and the set of all behaviours from Table 8.1. The resulting dataset contained 26 instances of each behaviour giving a total of 182 behaviour traces.

Inference was performed multiple times on the same dataset using Table 8.3 to provide the primitive feature true-positive rates. Only the number of particles per complex feature was altered per experiment. It is anticipated that as the number of particles is increased the mean runtime per observation will increase linearly, and will support hypothesis 3.

Results:

Figure 8.13 shows that the experiments provide support for hypothesis 3. This figure plots the mean filter runtime for each observation as the number of particles increases from 10 to 500, and shows that runtime increases approximately linearly. One can see that for every additional 100 particles the mean inference time increases by approximately 1000ms.

Cross-referencing Figures 8.13 and 8.12 shows that a range of 200 – 300 particles will deliver an F-Score ≥ 0.8 in around 3 seconds. These metrics are of course implementation dependent, but demonstrate a relatively effective method for determining the number of particles to use.

It should be noted that the X-axis in Figure 8.13 represents the quantity of particles used for each complex feature, rather than the total number of particles. Furthermore, recall from Chapter 6 that six agents require 21 hierarchical filters to detect all solo and multi-agent behaviours. It is the total runtime of all (21) hierarchical filters that is presented in Figure 8.13.

8.4.2 Number of Agents

This section analyses the growth rate of the number of goal filters required to identify multi-agent behaviour. Rather than proposing and evaluating a hypothesis, the purpose of this evaluation is to better understand the growth rates involved, and to visually represent this data.

Figure 8.14 shows the number of goal filters that are required to identify multi-agent behaviour as the number of agents increases. When agent behaviour is restricted to solo and paired collaborators the growth rate appears gradual, although it should be noted that it is still quadratic. 20 agents requires 210 goal filters, while 24 agents requires 300. When agents can perform solo behaviours, or collaborate in pairs or triplets, the figure shows that the growth rate quickly becomes exponential. Under these constraints 20 agents requires 1350 filters, and 24 agents requires 2324.

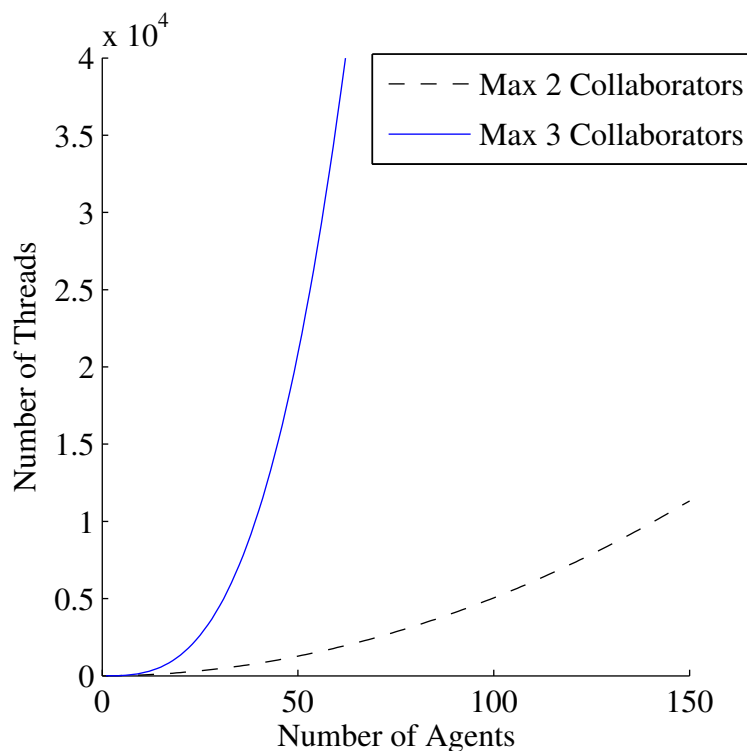


Figure 8.14: Thread growth rates

8.5 Summary

The experiments in this chapter focused upon rigorously testing different scenarios using simulated data. Under standard conditions (10% classification errors, 220 particles, single behaviours) the best performance was achieved by using prediction criteria to determine when a behaviour terminates and making predictions at that point only. An F-Score of 0.92 was achieved which is 9% better than when predicting behaviour at the end of a simulation. Furthermore, the experiments showed that event details (frame time, agent) could be accurately extracted from the filters once a behaviour had been detected allowing accurate behaviour summaries to be reported to operators.

Four types of behaviour variability were considered: presentation of unknown behaviours, ordering mutations on known behaviours, behaviour switching/concatenation and superfluous activities. When unknown behaviours were observed these sequences were generally ignored; that is, they were not misclassified as known behaviours. The average F-Score was 0.89, with the slight change in performance being largely due to dataset variability rather than the presence of unknown behaviour. Small ordering mutations interrupted contiguous sub-goals by 1 – 2 observations, which had negligible effect on recognition. Although some impact was observed on the posterior distributions these changes were not significant enough to effect prediction accuracy. However, concatenated and switched behaviour did effect performance. When a concatenated behaviour

was performed the F-Score dropped by 19% to 0.73, and dropped to 0.68 when behaviour switching occurred. A similar score (0.64) was also obtained when superfluous activities were observed such as placing and removing a luggage item multiple times.

To evaluate performance under more complex conditions the level of classification errors was also altered to range from 0 – 40%. Performance decrease is approximately linear, dropping by 0.1 for every 10% increase in primitive classification error. The approach also scales well with the number of particles, with sharp improvements in performance as the number of particles is raised from 10 – 100, and more gradual improvements thereafter. Optimal performance was achieved under zero classification error, while a maximum accuracy of 0.9 was achieved under 10% classification error using up to 500 particles. Aside from recognition performance, the number of particles also effects runtime with an increase of approximately 1000ms per observation for every 100 additional particles (with six concurrent agents). Where the approach scales less well is with regards to the number of agents, which is exponential. On an Intel Dual Core 2.4Ghz with 8GB RAM a limit of approximately 20 agents was encountered if real-time processing was to be achieved under standard conditions.

Algorithm 8.1 Generating a simulated scenario with random agent behaviours

```
1: Prototype:  $[S] = \text{GenerateUnkownBehaviourScenario}(ER, EM, N, AllB, AllC)$ 
2: Init:  $numberOfAgents = 0, scenarioIndex = 0, scenarios[] = \emptyset$ ,
3: Init:  $agents[] = \emptyset, observationSequence = \emptyset, S = \emptyset$ 
4: {Step 0: Add a random behaviour}
5:  $numFeaturesInBehaviour = \text{RandomInteger}(2, 8)$  {Range  $[2 : 8]$ }
6:  $beh = \emptyset, roles[] = \{1\}$ 
7: for  $f = 1 : numFeaturesInBehaviour$  do
8:    $role1 = roles[\text{RandomInteger}(1, roles.size())]$  {Get a random role}
9:    $feature = \text{RandomFeature}(AllComplex)$ 
10:  if  $\text{IsMultiAgentFeature}(feature)$  then
11:    if  $roles.Size() = 1$  then
12:       $roles.Append(2)$ 
13:       $role2 = 2$ 
14:    else
15:       $role2 = roles.Not(role1)$ 
16:    end if
17:     $beh.Append(feature, role1, role2)$ 
18:  else
19:     $beh.Append(feature, role1)$ 
20:  end if
21: end for
22:  $AllB.Append(beh)$  {Add to the set of all behaviours}
23: {Step 1: Generate random scenarios for  $N$  agents}
24: repeat
25:    $beh = \text{SelectRandomBehaviour}(AllB)$ 
26:    $numberOfAgents = numberOfAgents + numberOfAgentsRequired(beh)$ 
27:    $seq = \text{GenerateObservationTrace}(beh, agents)$ 
28:    $scenarios[scenarioIndex] = seq$ 
29:    $scenarioIndex = scenarioIndex + 1$ 
30: until  $numberOfAgents \geq N$ 
31: {Step 2: Merge the scenarios into a single observation sequence}
32: repeat
33:    $i = \text{RandomInteger}(scenarios.Size())$ 
34:    $currentSeq = scenarios[i]$ 
35:    $observation = currentSeq.RemoveFirstElement()$ 
36:    $observationSequence.Append(observation)$ 
37:   if  $currentSeq.Size()=0$  then
38:      $scenarios = \text{RemoveEmptyIndex}(scenarios, i)$ 
39:   end if
40: until  $scenarios.Size() = 0$ 
41: {Step 3: Apply the error model to the sequence at the appropriate error level}
42:  $S = observationSequence$ 
43: for  $i = 1 : observationSequence.Size()$  do
44:    $r = \text{RandomDouble}()$ 
45:   if  $r < ER$  then
46:      $classificationError = \text{GenerateRandomError}(agents, EM, AllPrims)$ 
47:      $S.Insert(classificationError, i)$ 
48:   end if
49: end for
50: return  $S$ 
```

Algorithm 8.2 Generating a dataset containing concatenated behaviours

```
1: Prototype:  $[S] = \text{GenerateConcatenatedScenario}(ER, EM, N, AllB)$ 
2: Init:  $F = \emptyset, S = \emptyset$ 
3: for  $a = 1 : AllB.Size()$  do
4:   for  $b = 1 : AllB.Size()$  do
5:      $scenarios[] = \emptyset$ 
6:      $observationSequence = \emptyset$ 
7:      $beh1 = AllB[a]$ 
8:      $beh2 = AllB[b]$ 
9:      $concatTrace = \text{GetConcatTrace}(beh1, beh2)$ 
10:     $scenarios.Add(concatTrace)$ 
11:     $stdTraces = \text{GenerateScenario}(0, EM, N, AllB)$ 
12:     $scenarios.AddAll(stdTraces)$ 
13:    {Next: merge the scenarios into a single observation sequence}
14:    repeat
15:       $i = \text{RandomInteger}(scenarios.Size())$ 
16:       $currentSeq = scenarios[i]$ 
17:       $observation = currentSeq.RemoveFirstElement()$ 
18:       $observationSequence.Append(observation)$ 
19:      if  $currentSeq.Size()=0$  then
20:         $scenarios = \text{RemoveEmptyIndex}(scenarios, i)$ 
21:      end if
22:    until  $scenarios.Size() = 0$ 
23:    {Next: Apply the error model to the sequence at the appropriate error level}
24:     $Temp = observationSequence$ 
25:    for  $i = 1 : observationSequence.Size()$  do
26:       $r = \text{RandomDouble}()$ 
27:      if  $r < ER$  then
28:         $classificationError = \text{GenerateRandomError}(EM, AllPrims)$ 
29:         $Temp.Insert(classificationError, i)$ 
30:      end if
31:    end for
32:     $S.Add(Temp)$ 
33:  end for
34: end for
35: return  $S$ 
```

Algorithm 8.3 Generating a dataset containing switched behaviours

```
1: Prototype:  $[S] = \text{GenerateSwitchedScenario}(ER, EM, N, AllB)$ 
2: Init:  $F = \emptyset, S = \emptyset$ 
3: for  $a = 1 : AllB.Size()$  do
4:   for  $b = 1 : AllB.Size()$  do
5:      $scenarios[] = \emptyset$ 
6:      $observationSequence = \emptyset$ 
7:      $beh1 = AllB[a]$ 
8:      $beh2 = AllB[b]$ 
9:     if  $beh1 \neq beh2 \wedge beh1 \neq PT1 \wedge beh2 \neq PT1$  then
10:       $concatTrace = \text{GetConcatTrace}(beh1, beh2)$ 
11:       $scenarios.Add(concatTrace)$ 
12:       $stdTraces = \text{GenerateScenario}(0, EM, N, AllB)$ 
13:       $scenarios.AddAll(stdTraces)$ 
14:      {Next: merge the scenarios into a single observation sequence}
15:      repeat
16:         $i = \text{RandomInteger}(scenarios.Size())$ 
17:         $currentSeq = scenarios[i]$ 
18:         $observation = currentSeq.RemoveFirstElement()$ 
19:         $observationSequence.Append(observation)$ 
20:        if  $currentSeq.Size()=0$  then
21:           $scenarios = \text{RemoveEmptyIndex}(scenarios, i)$ 
22:        end if
23:      until  $scenarios.Size() = 0$ 
24:      {Next: Apply the error model to the sequence at the appropriate error level}
25:       $Temp = observationSequence$ 
26:      for  $i = 1 : observationSequence.Size()$  do
27:         $r = \text{RandomDouble}()$ 
28:        if  $r < ER$  then
29:           $classificationError = \text{GenerateRandomError}(EM, AllPrims)$ 
30:           $Temp.Insert(classificationError, i)$ 
31:        end if
32:      end for
33:       $S.Add(Temp)$ 
34:    end if
35:  end for
36: end for
37: return  $S$ 
```

Algorithm 8.4 Generating scenarios with superfluous features

```
1: Prototype:
2:  $[S] = \text{GenerateScenarioWithSuperfluousFeatures}(ER, EM, N, AllB)$ 
3: Init:  $numberOfAgents = 0, scenarioIndex = 0, scenarios[] = \emptyset$ 
4: Init:  $agents[] = \emptyset, observationSequence = \emptyset, S = \emptyset$ 
5: {Step 1: Generate scenarios for  $N$  agents}
6: repeat
7:    $beh = \text{SelectRandomBehaviour}(AllB \setminus PT1)$ 
8:    $numberOfAgents = numberOfAgents + numberOfAgentsRequired(beh)$ 
9:    $seq = \text{GenerateObservationTrace}(beh, agents)$ 
10:   $numSurpPairs = \text{RandomInteger}(1, 2)$  {Insert one or two pairs}
11:  for loop=1:numSurpPairs do
12:    if  $\text{RandomBoolean}()$  then
13:       $seq.\text{InsertRandomly}(\{PlaceObject, RemoveObject\})$ 
14:    else
15:       $seq.\text{InsertRandomly}(\{FormGroup, SplitGroup\})$ 
16:    end if
17:  end for
18:   $scenarios[scenarioIndex] = seq$ 
19:   $scenarioIndex = scenarioIndex + 1$ 
20: until  $numberOfAgents \geq N$ 
21: {Step 2: Merge the scenarios into a single observation sequence}
22: repeat
23:   $i = \text{RandomInteger}(scenarios.Size())$ 
24:   $currentSeq = scenarios[i]$ 
25:   $observation = currentSeq.RemoveFirstElement()$ 
26:   $observationSequence.Append(observation)$ 
27:  if  $currentSeq.Size()=0$  then
28:     $scenarios = \text{RemoveEmptyIndex}(scenarios, i)$ 
29:  end if
30: until  $scenarios.Size() = 0$ 
31: {Step 3: Apply the error model to the sequence at the appropriate error level}
32:  $S = observationSequence$ 
33: for  $i = 1 : observationSequence.Size()$  do
34:   $r = \text{RandomDouble}()$ 
35:  if  $r < ER$  then
36:     $classificationError = \text{GenerateRandomError}(agents, EM)$ 
37:     $S.Insert(classificationError, i)$ 
38:  end if
39: end for
40: return  $S$ 
```

Chapter 9

Experimental Validation using Video

This chapter evaluates performance of the end-to-end video recognition framework described in Chapter 7. The experiments make use of two video datasets. The first is the publicly available PETS 2006 dataset [139], which was recorded by London Transport Police at a public transport hub. This thesis utilised the following videos in which the behaviours *PT1*, *PT2*, *AO1*, and *AO2* are all directly observable.

- S1-T1-C Camera 3
- S3-T7-A Camera 3
- S4-T5-A Camera 3
- S6-T3-H Camera 3

Additionally, the *WI* behaviour was synthesised by merging tracking data from two different videos (S4-T5-A and S6-T3-H), but unfortunately there are no instances of the *TH* or *HO* behaviours in the data. The PETS data makes use of three common types of luggage item: a briefcase, a wheeled cabin suitcase and a 25 litre rucksack.

A major disadvantage of the PETS data is that the size of the dataset is very limited. Aside from the lack of *TH* and *HO* behaviours, many of the other behaviours are only exhibited twice. This makes it challenging to not over-tune the video event detection modules. To overcome this data limitation a second dataset was recorded by the author in which all behaviours were demonstrated twenty-four times. Again, a selection of different size luggage items were used to increase the variability of the data and included: a wheeled

suitcase, two flight cases and a 25 litre rucksack. This dataset is termed the Heriot-Watt dataset, or HW for short.

9.1 The Video Based Inference Process

Details of the implementation framework have already been discussed in Chapter 7. However, to place this framework in the context of the experiments this section will describe the inputs and outputs of each process and will identify where errors were typically observed during video based inference.

Person and Object Tracking

The person tracker (see Section 7.1.1) operates directly on the input video and associates a unique identifier with each tracked person. When a new person is identified a time-stamp (frame number) based trace of their real-world x, y position begins, and is output against their unique identifier. If the agent is lost by the tracker the agent trace is terminated, and if they are subsequently re-identified, they will be associated with a new agent Id. The person's real-world coordinates are identified by using the PETS camera calibration data to project image coordinates to the ground plain (see [84]). The x,y origin for the PETS scene can be seen in Figure 9.1, while example output from the person tracker for several frames can be observed in Figure 9.2.

The static object detector also outputs tracking information in a similar format and consists of the following fields:

- X, Y position (Real World/Ground plain)
- Object Id
- Frame Number

Like the person tracker, an object trace commences when a static luggage item is first identified, and the output is terminated when the object is lost. Similarly, if an object is re-identified after it has been lost it will be tracked with a new object identifier. The

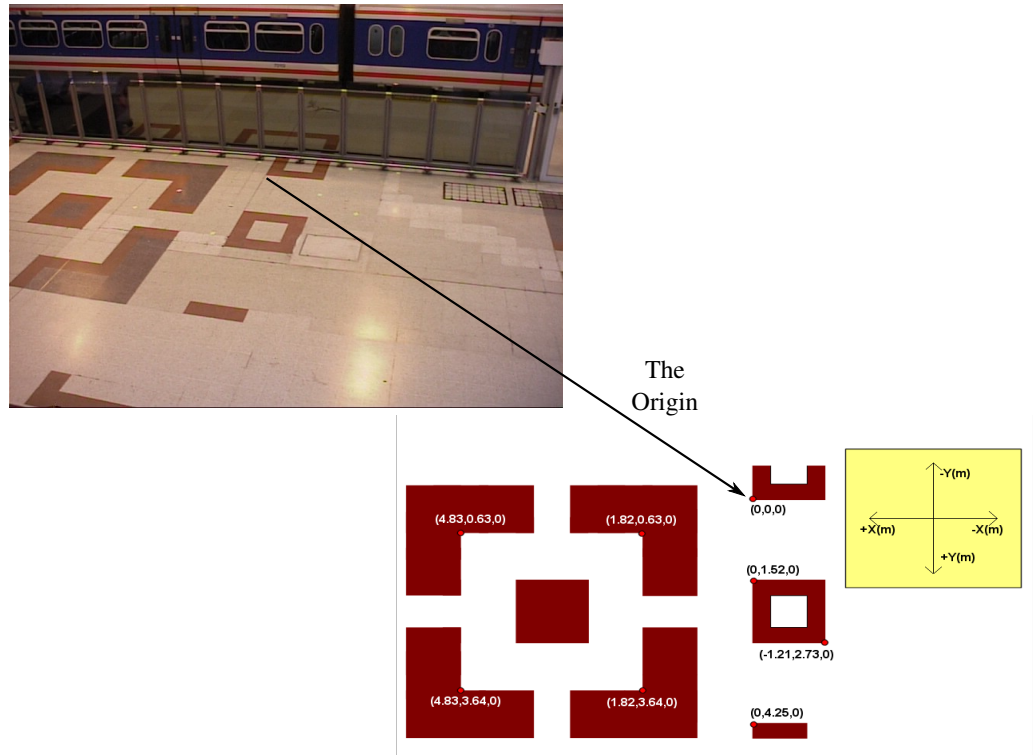


Figure 9.1: Camera calibration data for PETS scene S1-T1-C (Camera 3) showing the x,y, origin.

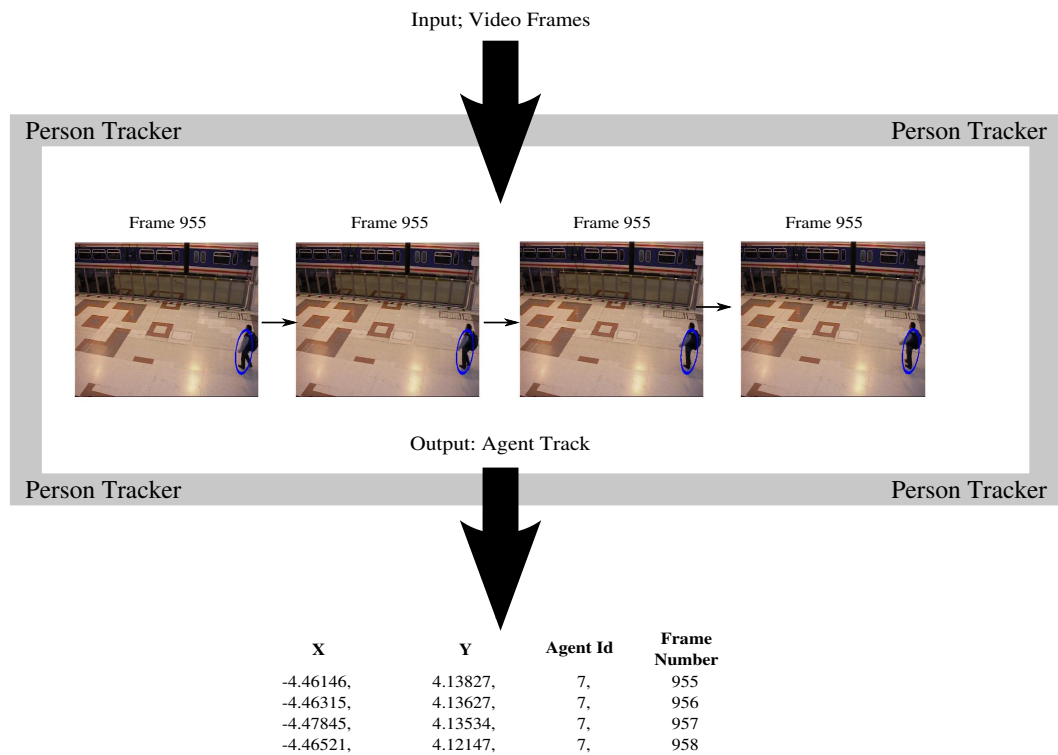


Figure 9.2: Example output from the person tracker for PETS scene S1-T1-C (Camera 3)

implications of having a single object with multiple object Ids will be discussed further in the next section.

Video Event Detection

Section 7.1.2 outlined six primitive features that can be detected from person and object tracking information. Each detected feature becomes an observation for the bag-of-features inference algorithm, and thus high-level inference is decoupled from the original sensors that provide agent and object observations. As an example, Figure 9.3 shows the chain of low-level inference steps leading to an *EnterAgent* observation: the person tracker identifies person 7 as a new track and starts outputting tracking information. This is consumed by the Agent Tracker, which detects an *EnterAgent* event at frame 995. In turn, the Agent Tracker outputs an *EnterAgent* primitive into the bag-of-features observation stream, upon which high-level inference is performed.

However, in addition to genuine tracking information, errors in the tracking data also cascade into the event detection modules. Tracking errors include agents and objects that are detected multiple times, and prematurely terminated tracks. These errors cause feature classification error and missed detections. Furthermore, the naivety of the feature detection modules means that certain conditions also lead to additional classification errors being generated. These are described in Table 9.1.

As a result of such errors the video processing stage of the inference framework outputs a stream of primitive feature observations containing classification errors, and ties in with the simulated classification errors discussed in the previous chapter. Positioned side-by-side in Figure 9.4 one can thus see how the simulated behaviour traces mimic the video processing framework.

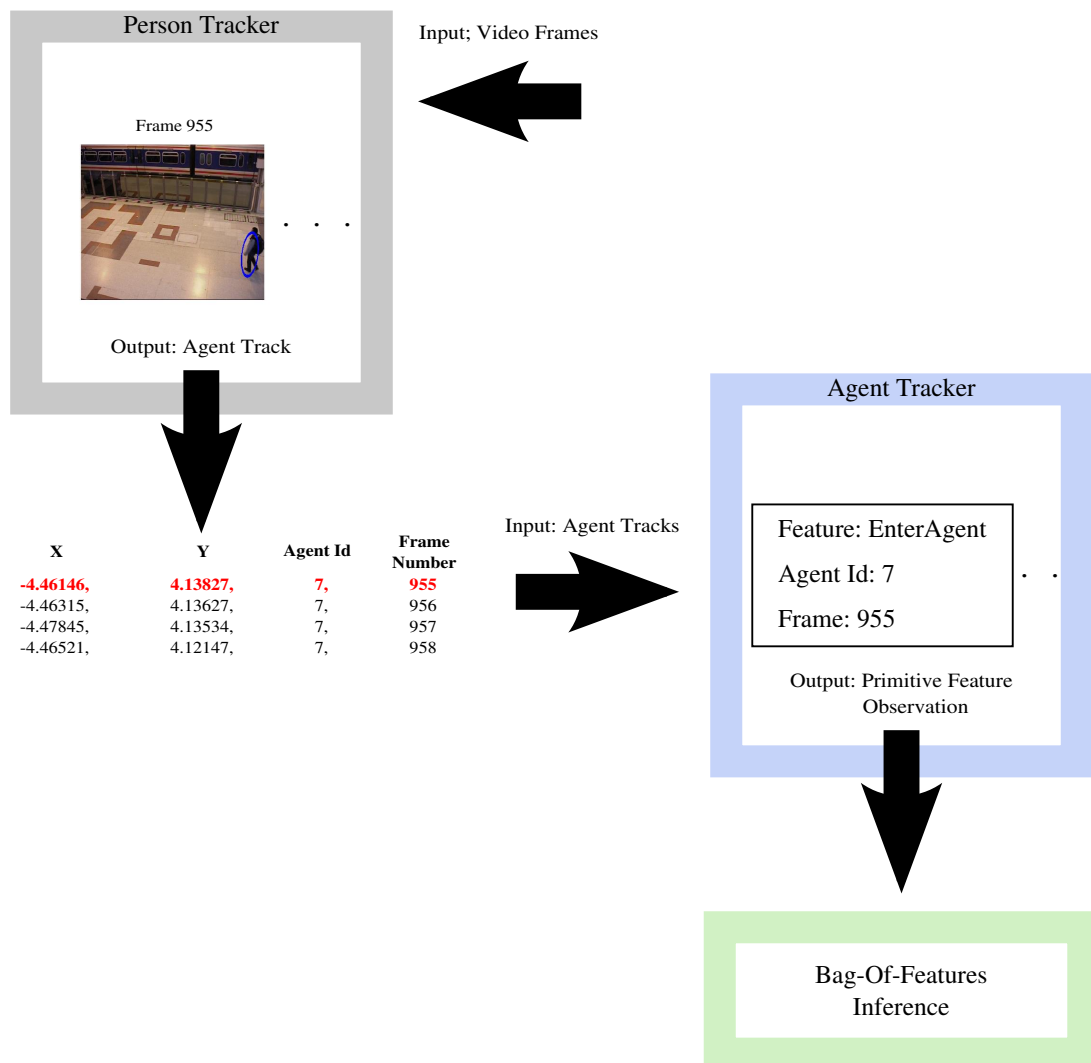


Figure 9.3: Person tracking information is provided to the Agent Tracker which detects and *EnterAgent* event. This event is output as a Primitive Feature and provided as input to the bag-of-features inference algorithm.

Error Type	Feature	Description
Classification Error	<i>EnterAgent</i> <i>ExitAgent</i>	If a tracked agent is lost without the agent leaving the scene (e.g. view is obstructed) this will cause the agent's track to terminate and an <i>ExitAgent</i> feature to be generated. If the agent returns to the field of view they will be associated with a new agent Id, causing a new <i>EnterAgent</i> feature to be generated. Occurrences of such an event thus cause two classification errors to be generated in the observation stream.
Classification Error	<i>EnterAgent</i> <i>ExitAgent</i>	Similar to above. A fast moving agent will sometimes cause an agent to be lost and re-found. This will cause the termination of the original track and the generation of a new track with a different agent Id. Erroneous <i>EnterAgent</i> and <i>ExitAgent</i> features result.
Classification Error	<i>PlaceObject</i> <i>RemoveObject</i>	When a dark shadow moves across an object this can cause the perceived size of the object to change. If the centroid of the new object is $> 0.3m$ away from its original a new object will be detected and an erroneous <i>PlaceObject</i> feature generated. Removal of the shadow or original object will then cause erroneous <i>RemoveObject</i> feature observations.
Classification Error	<i>FormGroup</i> <i>SplitGroup</i>	Agents travelling in close proximity can cause erroneous <i>FormGroup</i> features to be generated. Any subsequent increase in distance between such agents can then cause a <i>SplitGroup</i> detection.
Missed Classification	<i>EnterAgent</i> <i>ExitAgent</i>	Two agents travelling in very close proximity can cause a single agent track with a single agent Id. This means that only one <i>EnterAgent</i> feature will be generated. If both agents remain in very close proximity for the duration of their time in the scene, an <i>ExitAgent</i> will never be generated for the second agent when the agents leave the field of view.
Missed Classification	<i>PlaceObject</i> <i>RemoveObject</i>	In scenes with strong shadows it is not uncommon to observe that a static object is not recognised. This is because the person's shadow and object overlap and thus the size constraints imposed by object detection are not met. This prevents the generation of a <i>PlaceObject</i> feature, and any subsequent removal of that object will fail to generate a <i>RemoveObject</i> feature.

Table 9.1: Tracking errors are the normal cause of missed classifications, and are partially responsible for classification errors.

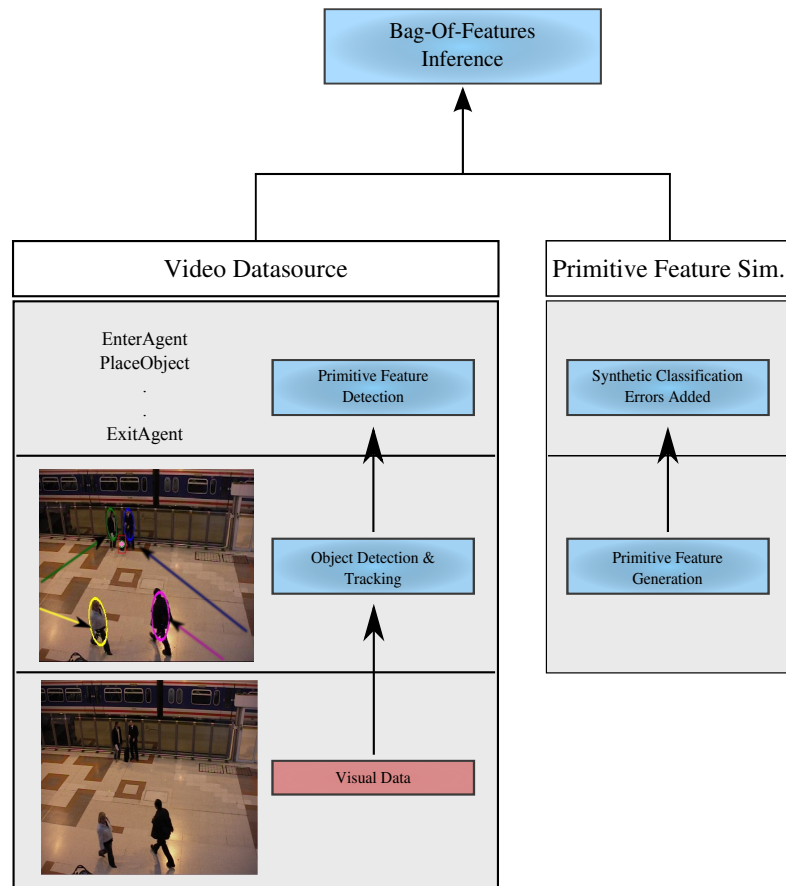


Figure 9.4: The video processing components of the framework provide a stream of primitive feature observations to the Bag-of-features inference algorithm. The primitive feature simulator also provides such a stream and incorporates synthetic classification errors to mimic video processing failures.

Module	Parameter	Value/Range
Person Tracker	Ellipse height	1.8m
	Ellipse width	0.4m
Object Tracker	Min width/height	0.3m
	Max width/height	1.0m
	Max distance from original centroid	0.3m
	Min static time	1 sec
Group Tracker	Min distance between members	2.5m
	Min grouping time	3 sec
	Bounce time	2 sec

Table 9.2: The parameters used with the tracker and event detection modules.

9.2 Event Recognition Performance

Hypothesis: The event detection modules can detect events from PETS and HW Video frames

To establish whether complex behaviours can be recognised from video it must be determined whether the primitive features can be detected from video. Support for the hypothesis would be gained by successfully recognising the primitive events from video taken in multiple environments. The null hypothesis is that the events cannot be recognised across both datasets.

Experimental Design

To test this hypothesis the experiment was designed to determine the precision and recall of the detectors when presented with the PETS and HW data. To ensure that the video processing modules were not over-tuned for each environment the modules were configured so that one set of parameters delivered reasonable recognition performance on both datasets. A single exception to this rule was made for the object detector, where the pixel change threshold was altered to overcome a change to the lighting configuration that occurred during HW data collection.

The trackers (person/object) and the event detectors used the fixed set of parameter values defined in Table 9.2.

To obtain performance information the videos from each dataset were presented to the person and object trackers, and their output was in turn presented to the event detection modules.

Results

Figure 9.5 shows the precision of each of the six detection modules: *EnterAgent*, *ExitAgent*, *FormGroup*, *SplitGroup*, *PlaceItem* and *RemoveItem*. One can see from the figure that the *FormGroup* event has a very low precision in the PETS data (0.33), and is slightly higher (0.62) in the HW data. This is because the PETS data only exhibits one example of the *FormGroup* event, although two false-positives were also detected in a particularly challenging scene (S1-T1-C). Frames from this scene are shown in Figure 9.6 where one can observe that four agents enter the scene in close proximity. This in itself causes problems for the person tracker, which changes the persons associated with the green and yellow ellipses, and fails to detect two agents at all.

The HW data contains eight true-positive *FormGroup* detections but five false positives are also generated during some of the *TH* behaviours. The *RemoveObject* event is also detected with particularly low precision in both datasets (HW: 0.65, PETS: 0.5), while the other events are detected with reasonable accuracy (≥ 0.74).

Figure 9.7 shows that the recall of the events is generally high. Again, the primary exception is the *FormGroup* event which has a recall of only 0.47 in the HW data. All other events have a recall of ≥ 0.73 on both datasets. Combining the precision and recall over both datasets leads to a mean event recognition F-Score of 0.78.

These results provide some support for the hypothesis, although it is clear that the event recognition performance is limited. The majority of events can be recognised with a precision ≥ 0.74 and a recall ≥ 0.73 , although some events are recognised with poorer performance. This is likely to be caused by the naivety of the event detection modules, with the method of group detection (proximity based) particularly unreliable. That said, there is clear support that the events can be recognised from both video datasets.

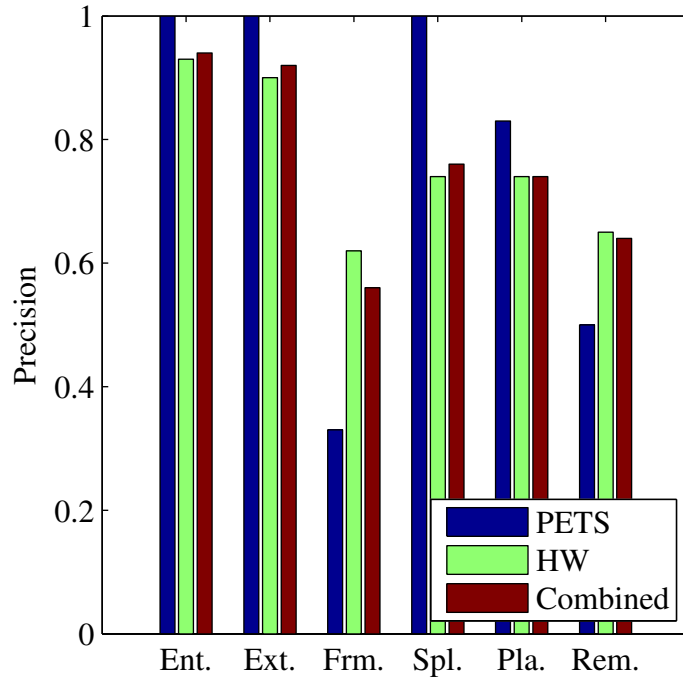


Figure 9.5: Video event detection precision



Figure 9.6: Frames from PETS scene S1-T1-C. A group of agents entering the scene together and travelling as a group causes tracking errors and false *FormGroup* detections.

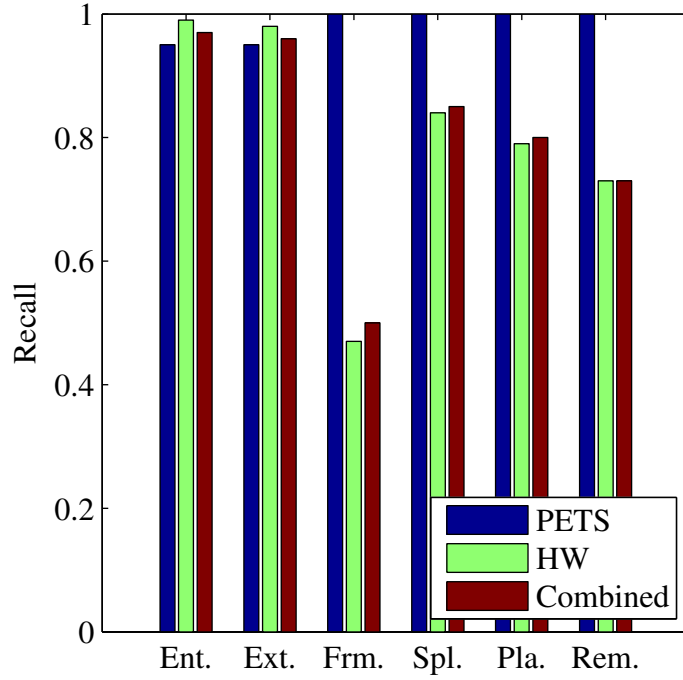


Figure 9.7: Video event detection recall

9.3 Behaviour Recognition Performance

Hypothesis: Complex behaviours can be recognised from video by integrating video processing and Bag-of-features inference.

By using the video processing framework to convert raw video data into primitive feature observations an observation stream can be provided for bag-of-features inference. If the sample behaviours can be successfully recognised by the inference algorithm then it can be concluded that bag-of-features inference can be performed on video based data. The null hypothesis is that they cannot be recognised from video-based data.

Experimental Design

To test this hypothesis experiments were performed with each video in the PETS and HW datasets. As highlighted at the beginning of the chapter, the PETS dataset consisted of scenarios S1-T1-C, S3-T7-A, S4-T5-A and S6-T3-H. Additionally, a synthetic PETS scenario was generated through a combination of scenarios S4-T5-A and S6-T3-H. This was achieved by truncating the tracker output from S4-T5-A at video frame 1000 and

Behaviour	HW	PETS
PT1	24	68
PT2	24	1
WI	24	2
AO1	24	2
AO2	24	2
TH	24	0
HO	24	0

Table 9.3: The number of instances of each complex behaviour in the datasets.

joining this with the tracker output for S6-T3-H commencing with frame 1370.

Table 189 shows the number of instances of each behaviour within the two datasets. Bag-of-features inference was performed with 220 particles per complex feature and the primitive feature true-positive rates defined in Table 8.3 on page 143. As in the previous section, the trackers and event detectors used the parameters from Table 185.

Results

Figure 9.8 shows behaviour recognition accuracy on the PETS data. One can clearly see that excellent results are achieved with a mean F-Score of 0.99. This is however an indicator that the dataset is of an insufficient size to obtain a realistic measure of performance. All behaviours are detected with a very high F-Score, with *PT1* being the only behaviour detected imperfectly with an F-Score of 0.96. These inaccuracies were generated as a result of the person tracker incorrectly detecting luggage items as people, with two examples of this shown in Figure 9.9.

When the framework is applied to the HW data the results obtained are much more varied. Figure 9.10 shows that the *PT1* behaviour is still recognised with the lowest F-Score and has dropped to 0.62. Again, tracking errors are generally the cause of these inaccuracies. The remaining behaviours are all detected with an F-Score ≥ 0.7 and have a mean F-Score of 0.77. These results provide support for the hypothesis.

Through analysing the scenarios in which incorrect behaviour classifications were made it was determined that tracking errors (specifically: multiple/missed person detections) and group detection errors were the primary reason for incorrect behaviour recognition. It is expected that better behaviour recognition performance could be achieved if these two sources of low-level classification errors were to be improved.

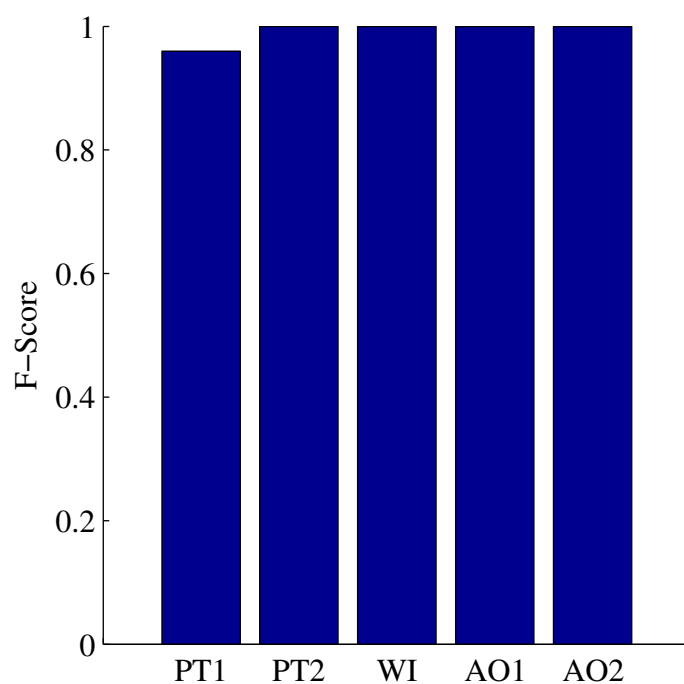


Figure 9.8: Behaviour recognition accuracy on the PETS data

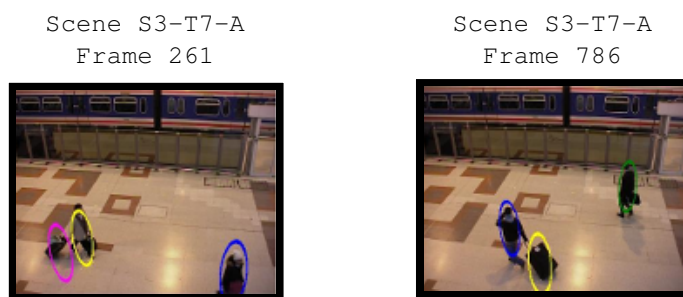


Figure 9.9: Two examples of the person tracker misclassifying luggage as people on the PETS dataset.

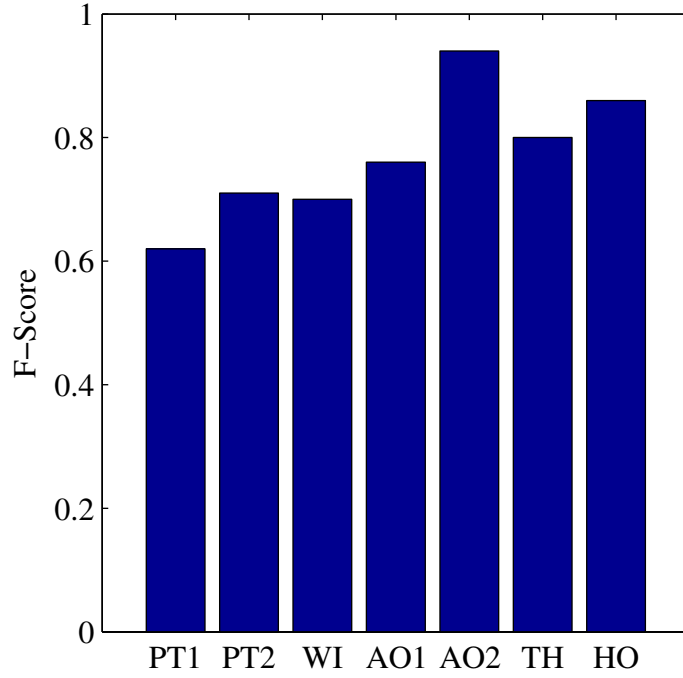


Figure 9.10: Behaviour recognition F-Scores on the HW data

9.4 Summary

This chapter focused on experiments using real video taken from the PETS 2006 dataset and a new (HW) dataset. To demonstrate the performance of the activity detectors, upon which high-level inference relies, the first set of experiments considered video event detection accuracy. Most of the events are detected with reasonable precision (≥ 0.74), although the *FormGroup* event is recognised with a precision of 0.33 in the PETS data and 0.62 in the more extensive HW data. The *RemoveObject* event also suffers from low precision, ranging from 0.5 – 0.65. These two events also suffer from the lowest recall, with *FormGroup* only achieving 0.47 on the HW data. A recall of 0.73 is achieved for *RemoveObject*, with all other events performing better.

While evaluating behaviour recognition performance against the PETS data our mean F-Score of 0.99 shows that this dataset is limited by size. On the more extensive HW data the mean F-Score drops to 0.77 with individual scores ranging from 0.62 to 0.94. Nevertheless, this performance is quite impressive given the relatively poor accuracies of the video event detection modules.

Chapter 10

Discussion

Nomenclature:

N	Number of particles	y_t	Observation at time t
B^k	The k th behaviour	$y_{1:T}$	Observation sequence (y_1, y_2, \dots, y_T)
$ B^k $	Number of features in B^k	Z_t	State at time t

This thesis proposed that probabilistic behaviour recognition could be achieved by decomposing behaviours into bags of features. It was argued that the primary advantage of this approach was that training corpora was no longer required, allowing probabilistic recognition to be performed in domains where training data is scarce. This chapter will discuss the support that the experimental results give to the approach, as well as evaluating the wider recognition framework in which it was set. This will include a discussion on its ability to meet real-world constraints such as operating in noisy environments generating low-level classification errors.

10.1 Feature Based Recognition

The experiments demonstrate encouraging support for the approach with in-sequence predictions showing very little behaviour confusion. Table 8.5 on page 150 showed that the mean false positive rate is only 0.34% on the simulated experiments. This rate increases

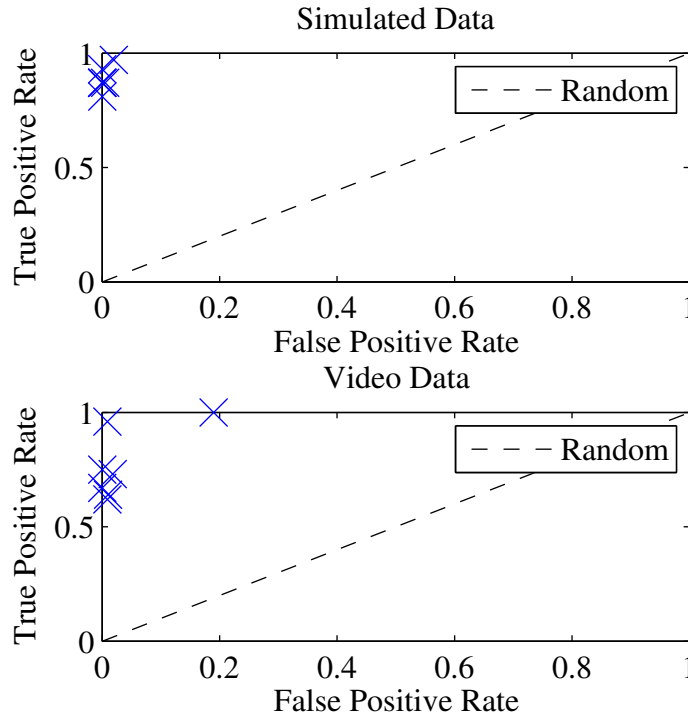


Figure 10.1: In-sequence prediction results plotted in ROC space. Optimal x,y coordinates: (0,1)

slightly to 3.4% in the combined video based experiments, but is none the less an excellent result. One can visualise these results by temporarily treating each behaviour as a binary classifier and plotting their performance in Receiver Operating Curve (ROC) space. This is shown in Figure 10.1, where one can see that all of the results are well above the ‘random’ decision boundary.

Section 8.1 showed that this accuracy is only achieved when applying prediction criteria that determines when a behaviour terminates. Chapter 7 introduced the prediction criteria used and the results show that enforcing these constraints improves the recognition F-Score by 0.09 over using the posterior filter densities alone. The high accuracy of behaviour prediction demonstrates that simulated annealing often identifies the correct assignments when a behaviour terminates. However, it was also observed that simulated annealing often identifies poor assignments both before and after behaviour termination. This means that a correct assignment made part-way through a simulation may not still be present by the end. It is for this reason that the prediction criteria proves very useful and significantly improves performance over post-simulation prediction.

It should be noted that changes to behaviour/collaborator assignments should be expected. Consider two agents who have partially performed two solo behaviours. If a multi-agent behaviour has a higher probability of having generated the set of their combined activities verses two individual behaviours, then it is correct that a collaborative assignment should

be made. As more observations arrive the probabilities for each behaviour will change in light of this further information, and again it should be expected that assignment changes will occur.

Because one cannot ever determine when a behaviour terminates using posterior filter density alone, one will always encounter the problem of changing assignments. The prediction criteria has demonstrated its ability in identifying behaviour termination and choosing the correct assignment, and thus this approach seems a suitable mechanism for achieving high-accuracy predictions. That said, this does imply that behaviours can only be predicted accurately when they are observed in their entirety.

These results are reasonable because a human can also only reliably predict the outcome of an event by using prior knowledge. For example, assume that an agent is observed entering the scene and placing luggage. A surveillance operator can anticipate that the agent will remove the luggage because of prior knowledge, but they will remain uncertain until such an action is observed, or the agent is observed abandoning the object. The bag-of-features approach is built upon the premiss that there is no prior knowledge, and thus one cannot expect the approach to make reliable early predictions.

There are also arguments against using (some types of) prior knowledge. Indeed in this scenario one might suggest that **not** using prior knowledge might be advantageous, because that prior knowledge may be incorrect. For example, a surveillance officer may choose to focus on one person rather than another because of racial prejudices, which are essentially (incorrect) priors. A report by the Missouri Attorney General has found that racial profiling by police officers actually lowers the rate of successful stop-and-search procedures, and can distract officers from using more effective policing techniques [4]. One cannot of course say that all priors are harmful because many are not, but there is none the less an argument that a human's priors may be ineffectual, and thus a prior-free approach could potentially outperform a human operator.

10.1.1 Earlier Prediction

To obtain earlier detections with the approach one must change the prediction criteria so that a smaller number of features have to be observed. A natural choice is $|B^k| - 1$: one less than the number of features defining each behaviour B^k . Figure 10.2 shows the affect that this has on prediction accuracy, and compares performance against the original (all features) results.

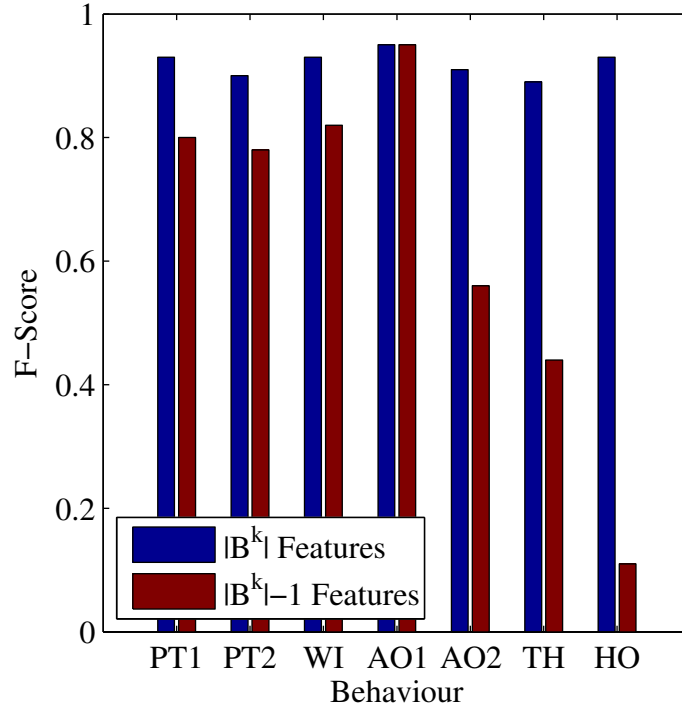


Figure 10.2: F-Scores for each behaviour B as the number of features required for prediction is altered. When predictions are made with one outstanding feature expected there is a 0.28 reduction in mean F-Score from 0.92 to 0.64.

It should be clear from the figure that altering the prediction criteria has a negative effect on accuracy. The mean F-Score reduces by 0.28 from 0.92 to 0.64, with the *HO* behaviour the most badly affected. Further analysis shows that *HO* has substantial confusion with *AO2* (48%) and *TH* (25%). Because the features in *AO2* are a subset of *HO* this confusion is to be expected if collaboration is not correctly detected. Furthermore, *HO* and *TH* differ by only one feature, with *HO* containing an additional instance of *ExitAgent*. The approach's minimal explanation feature is likely to be a part of this confusion.

One can conclude from the experiments that bag-of-features inference performs very well when presented with complete behaviours under moderate (10%) classification errors. To achieve low false-positive rates and high precision and recall the prediction criteria are essential for identifying when behaviours terminate. If one adjusts the prediction criteria to make earlier predictions, performance is damaged due to the lack of prior knowledge. Assuming that increased errors are an acceptable cost for early prediction there is no reason why the approach cannot be configured to operate in this manner, refining predictions as further observations are made.

10.1.2 Heuristic Particle Weights

Recall from Chapter 4 that the inference algorithm made use of a weighting function. This function was defined by Equation 4.9, although it was identified that the equation's structure was too complex to estimate in the absence of training data. It was then proposed that if a heuristic weight encapsulated the underlying principles of particle weighting then convergence should still have been achievable. Equation 5.9 was the heuristic suggested and was used throughout the experiments.

Aside from measuring prediction accuracy the results can also be used to determine the efficacy of the heuristic weight. The theoretical underpinning of particle filtering tells us that as the number of particles in a filter (N) increases, the estimated density of $P(Z_T|y_{1:S})$ should converge to the true value [9]. One would therefore expect that behaviour predictions should also improve until some maxima is reached. With this premiss in mind Figure 8.12 on page 169 can be used to analyse how well the heuristic weight performs by showing recognition accuracy as the number of particles is varied.

It is encouraging to see from the figure that recognition accuracy quickly improves to over 0.9 as the number of particles is raised from 10 – 100. By 200 particles accuracy reaches 0.95, and then continues to converge towards 1 as N is further increased. The fact that recognition accuracy approaches 1 as $N \rightarrow \infty$ shows that the heuristic works well in this domain and provides some support that the heuristic encapsulates some of the complex dependencies of Equation 4.9.

The figure also shows the effect of the number of particles under 10% classification errors, although optimal performance is not obtained by 500 particles. Unfortunately, it is unclear from the results whether optimal performance would ever be achieved under these conditions. Above 500 particles the run-time efficiency of the implementation prevents further data collection, although the gradient of the graph indicates that large improvements in accuracy would be unlikely without very large quantities of particles.

10.1.3 Behaviour Confusion

Table 8.5 on page 150 showed that behaviour confusion was minimal with no significant errors. One attribute of interest is that *HO* is often misclassified as *AO2*. To recap, the *HO* behaviour involves two agents. The first enters the scene, places luggage and leaves.

This sub-behaviour is identical to *AO2*. The second agent then enters the scene, removes the luggage and leaves the scene. This second sub-behaviour is very similar to *PT2*, with the only difference being that in *PT2*, the (same) agent places the luggage first. Further investigation showed that some combinations of erroneous features would cause the joint probability of both agents performing *HO* to drop below the probability of one agent performing *AO2* and the other performing *PT2*. This leads to the *HO* confusion observed. Because the *PT2* behaviour is not observed in its entirety a classification error is not made for this agent, but the recall is effected instead.

10.2 Real-World Application

One might argue that because the evaluation was performed within a visual surveillance domain the approach should be compared against a human baseline. However, there are a number of reasons why such a comparison is both infeasible and unnecessary. The wide spread deployment of Closed Circuit Television has led many to question the effectiveness of existing installations and has led to a growing base of literature. Howard *et al.* have identified that there are many factors affecting operator performance and state that visual overload is just one major difficulty [61]. They found that operator performance was effected by monitoring multiple targets, having to multi-task, and by the number of feeds that must be monitored. Other studies have found that sheer boredom and poor motivation are also major problems [129].

For instance, Tickner and Poulton compared operator detection performance when varying the numbers of monitors an operator must observe. They reported a drop in detection accuracy of 19% when moving from nine to sixteen monitors [140]. This is consistent with further research by the UK Police Scientific and Development Branch, which reported that observers viewing one, four, six and nine monitors achieved respective detection accuracies of 85%, 74%, 58% and 53% on a simple observation task [142]. This is just one variable that would have to be controlled in any human performance comparison and would require large video datasets to provide meaningful results.

Aside from visual overload the literature also suggests that human operators are fundamentally flawed. Studies by Norris and Armstrong found that 40% of people targeted by surveillance officers were targeted for no obvious reason and were chosen primarily for belonging to a particular subculture or group [110, 111]. One can argue that an automated visual surveillance system is beneficial not only for combating operators that are distracted from identifying real threats [61], but also because such systems are non-

discriminatory.

Furthermore, a study by Smith revealed the shocking drawback of requiring officers to choose which feeds to monitor [129]. They quote an operator from their study:

“I can’t tell you how many things we’ve missed when we have not been watching the other screens. Break-ins, assaults and car thefts have been going on whilst we’ve been operating the other cameras” ([129] page 385)

The inefficiencies of human operators are widely acknowledged in the literature and one can conclude that any automated system would be beneficial if it could reliably monitor concurrent feeds. In validating the effectiveness of an automated visual surveillance system one can still ask whether the system can outperform an operator, but the number of operational variables makes such a comparison of limited use. The more prevalent question is how well the system can detect behaviours of interest, which would allow the flagging of interesting behaviour when an operator is distracted or overloaded.

10.2.1 Explaining Behaviour

One of the arguments for behaviour recognition over anomaly detection is that reasoning can be explained to an operator while anomaly detection can only identify portions of video that are ‘interesting’. Although anomaly approaches are frequent in previous research [145, 8, 144, 12, 98, 97, 74], Dee and Velastin state in their summary paper on the field that:

“It could also prove necessary to produce easily comprehensible output, as any system designed to integrate within an existing CCTV control room environment will need to be accepted and understood by the operatives. As a human computer interaction problem, this has not received a great deal of attention to date;” “Any system performing (for example) behaviour summarisation will need to be able to produce results in a form that makes sense to those unfamiliar with computer vision.” [36]

Section 8.1.2 of the results not only presented the high accuracy and recall of the approach but also highlighted that behaviour explanations were optimal. An example explanation from the PETS data can be seen in Figure 10.3 from which one can say with

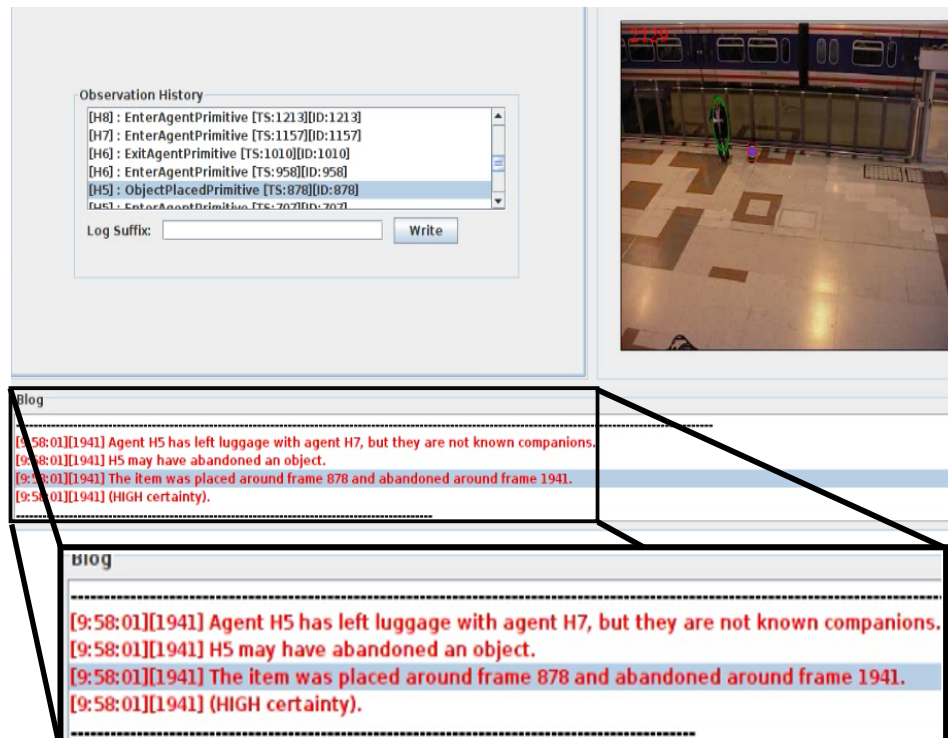


Figure 10.3: Example explanation from the PETS dataset (Scenario 4). The text reads: “Agent H5 has left luggage with agent H7, but they are not known companions. H5 may have abandoned an object. The item was placed around frame 878 and abandoned around frame 1941. (High certainty)”

some certainty that the detailed description is aligned with Dee and Velastin’s vision of comprehensible output.

A further argument for the approach was that the need to operate in noisy environments calls for probabilistic methods to be used. Indeed, Dee and Velastin advocated probabilistic approaches by stating that reasoning about uncertainty is paramount in successfully achieving high level behaviour recognition [36]. In surveillance applications low-level video processing errors generate uncertainty about what has really been observed. However, this problem is not unique to video surveillance, with many applications relying upon signal processing techniques to provide low-level observations. Indeed, early behaviour recognition techniques were often based upon first-order-logic [68], but their inability to reason about uncertainty has largely led to their demise in modern behaviour recognition research.

To demonstrate how the approach performs in noisy environments Figure 8.11 on page 166 showed recognition accuracy with simulated classification errors with a fixed number of particles. The results showed that accuracy dropped linearly as primitive classification errors increased and it is hypothesised that this reduction in performance is partly due to the

significant overlap of the validation behaviours. In several instances it is possible to convert one behaviour to another by adding only one or two primitive features. An analysis of the simulation data showed that various conversions did indeed occur. For example, the *PT1* behaviour can be converted into *AO2* via an erroneous *PlaceObject* feature. Furthermore, additional classification errors compounded the problem of *HO* \rightarrow *AO2* confusion. Nevertheless, the approach's ability to correctly recognise such behaviour is evident with an F-Score of 0.76 still achieved when observing behaviour containing 30% classification errors.

Having observed performance under simulated classification errors it is surprising that the video-based experiments deliver poorer performance than expected. In the simulations with 10% classification errors the average F-Score was 0.92, while the combined results from both video datasets gave an F-Score of 0.81. Given that the video event detectors generate classification errors at approximately 11% and have a similar distribution to the simulated data one would expect the two F-Scores to be closer. One of the reasons for this discrepancy is likely to be the error model used in the simulated experiments. Although the error model inserted erroneous observations it failed to take into consideration missed observations. This is tantamount to assuming that the event modules had a recall of 1, while in reality the modules achieved a mean recall of 0.8.

In competing approaches the temporal ordering of activities has helped to identify 'missing' observations. For example, Geib and Goldman considered behaviour recognition for detecting computer network intrusions (an Intrusion Detection System) [51]. Given a library of known computer attacks their approach attempted to recognise hacker behaviour in what is commonly referred to as hostile behaviour recognition. That is, an agent's actions are observed without the agent's knowledge or interaction and are potentially disguised. The authors argue that it should be expected that some actions will not be detected, and make use of the temporal constraints between actions to identify when this occurs.

Because bag-of-features recognition only imposes a weak temporal ordering it is more difficult to detect when an action has been missed. This is especially true when one considers the recall distribution in Figure 9.7 on page 188. Notice that the recall rates for placing and removing objects are correlated: one cannot detect that an object has been removed if it has never been detected as placed. However, there is still the potential that a filter could acknowledge that an object (for example) has been removed without acknowledging that it has been placed. Some kind of loose dependence could be encoded in the approach to overcome such errors, although identifying the benefits of such an alteration falls outside the time constraints of this research.

Approach	PETS Precision	PETS Recall	PETS F-Score
ObjectPlaced Module	0.83	1.0	0.91
Grabner <i>et al.</i> [55]	0.94	0.97	0.95
Smith <i>et al.</i> [130]	0.96	0.86	0.91

Table 10.1: Comparison of luggage detection (*PlaceObject*) accuracy

10.2.2 Event Detection Comparison

Comparing the accuracy of the event detection modules with competing approaches is somewhat difficult due to a lack of published results. However, one of the modules that can be compared is the *ObjectPlaced* module, which has more frequently been a component of other research efforts.

Table 10.1 compares the module’s accuracy on the PETS dataset against two other approaches using the PETS data. Smith *et al.* use an approach that is actually very similar to the one described in Chapter 7, with the key difference being that they prevent duplicate object detections by filtering any object candidates (small foreground pixel blobs) that lie on-top of other objects [130]. It is highly likely that a similar filtering step would have improved the precision of the *ObjectPlaced* module, where duplication errors were the primary reason for false-positive detections. However, one should also observe that the recall of Smith’s approach is only 0.86, suggesting that their filtering may also be removing true-positive detections. If one calculates the appropriate F-Scores it becomes apparent that Smith’s approach and the *ObjectPlaced* module are actually comparable as they have the same score.

Grabner *et al.* use an entirely different approach to detection based upon on-line AdaBoost [55]. Their approach performs well in comparison to the others, although it should also be highlighted that their results are based upon only two of the PETS scenarios. Furthermore, these scenarios differ from those used during this research and by Smith *et al.*, making it difficult to relate the results fairly.

Another research area that has attracted some publications is group formation and splitting. Marques *et al.* specifically looked at tracking groups of 2 – 4 pedestrians, identifying when merging and splitting occurred [93]. Their approach is based upon Bayesian Networks and makes use of several probabilities: a merging probability, splitting probability, occlusion probability and new track probability. They reported that their approach was able to correctly solve most situations, but did not publish any metrics against which the *FormGroup* and *SplitGroup* modules can be compared. Similarly, Bose *et al.* also track groups of pedestrians, although their work focused more upon tracking agents through

Approach	F-Score
Bag-of-features	1.0
Auvinet et al. [10]	1.0
Guler and Farrow [56]	0.8
Krahnstoever et al. [73]	0.86
Martinez-del-Rincón et al. [94]	1.0
Li et al. [80]	1.0
Lv et al. [91]	1.0
Smith et al. [130]	0.86

Table 10.2: Comparison of the Abandon Object (AO2) detection accuracy with competing techniques

group formation and splitting rather than detecting the events themselves [23]. In their approach two agents passing each other are considered as forming and splitting a group, which differs from the definitions used in this research.

10.2.3 Behaviour Detection Comparison

The high-level nature of the behaviours used within this research makes it difficult for results to be compared with competing approaches. Research in complex video event detection is disjointed and has few standard behaviours/datasets and ranges from aeroplane arrival procedures [46] to kitchen activities [76]. However, there is one behaviour; the detection of abandoned objects, that has received considerable attention. Unlike this research competing approaches normally use the distance of objects to their owners to identify abandonment [10, 56, 73, 94, 80, 91, 130], and thus no high-level understanding of the video is actually occurring. However, that is not to say that these approaches cannot be compared to the bag-of-features technique. Indeed, there are several papers from the PETS 2006 workshop with which results can be directly compared for the abandoned object (AO2) behaviour.

Table 10.2 compares video recognition performance for the PETS data with several competing approaches. To ensure fairness F-Scores have been calculated from the results for videos 1,3,4 and 6, which were the same videos used in this research. The table shows that the bag-of-features approach is comparable with four of the competing techniques, all achieving optimal recognition. Furthermore, it performs considerably better than three of the competing approaches, offering 0.2 and 0.14 improvements.

Although the automated video surveillance community is extremely active there are relatively few other results with which performance can be compared. Hakeem and Shah

have published research using normalised cuts to identify clusters of correlated activities. Their datasets include surveillance scenarios and a theft behaviour is mentioned, but there are no further details other than the mean results [57]. They report an F-Score of 0.87 for their surveillance scenarios, which is marginally higher than the F-Score achieved by bag-of-features inference (0.83). However, it is extremely unlikely that the behaviours being considered are the same in both pieces of research, and indeed it is unclear what behaviours were being recognised by Hakeem and Shah. Furthermore, because their work focused on behaviour recognition they justified their use of manual object identification and labelling. This removes some of the video processing tasks performed within this research and has the potential to significantly reduce the level of low-level classification errors under which high-level recognition must take place. In this research no manual object identification or labelling has been performed and thus there is an additional noise element under which our approach has had to perform.

Symbolic networks are another competing approach for complex-event recognition and thus some comparison of performance would be useful. Symbolic networks perform constraint satisfaction and are constructed from nodes representing video activities, and edges denoting spatio-temporal rules. Recent work by Fusier *et al.* has shown that complex behaviours can be recognised utilising spatio-temporal rules such as *before* and *during* [46]. In some ways spatio-temporal rules are more robust than bags-of-features because they can define event relationships. The bag-of-features approaches cannot represent relationships and is thus more restrictive of the types of behaviour that can be represented.

Although constraint satisfaction can represent complex relationships their primary limitation is that they cannot reason about uncertainty. Fusier *et al.* use real-world data for their evaluation and provide end-to-end recognition in a similar vein to this research. However, although they report that several complex behaviours are identified including two multi-agent scenarios, they provide no metrics regarding precision or recall. Furthermore, their validation environment considers aircraft arrival procedures and is thus relatively controlled in the number of actors concurrently present in a scene. This means that no performance comparison can be drawn with Fusier *et al.*, although such a comparison would prove interesting future work.

10.2.4 Runtime Scaling

Figure 8.13 on page 170 showed that the number of particles in a filter has a large impact on inference time. Recall that this experiment was performed with a constant number of six agents. At the far left of the graph it can be seen that even with only 100 particles

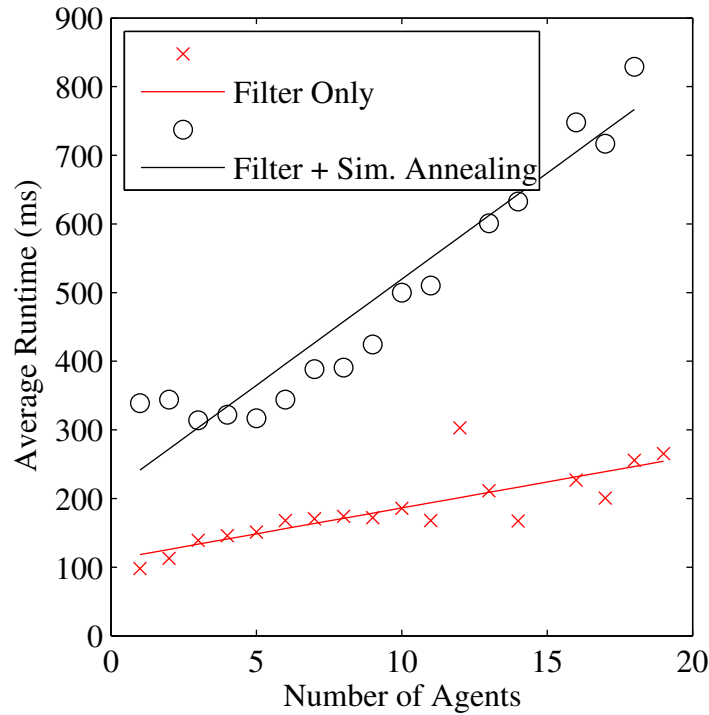


Figure 10.4: Runtime vs. number of agents with fixed 220 particles

runtime is approaching 2 seconds per observation. Video surveillance frame rates tend to range from 7 to 24 frames per second, which would require a processing speed of approximately 42 – 142 ms to achieve real-time processing. However, although the filters are operating slower than video frame-rate, the filters process video events, not video frames. Video event rates are data dependent and are primarily determined by scene occupancy, although a typical scene from the PETS dataset (scene 6) produced a rate of 1 event every 3.5 seconds. With such a rate up to 300 particles could be used to maintain real-time event processing performance.

Although the figure shows that real-time processing can be achieved, what may be unclear is that the number of agents changes the number of active filters, and thus also affects the over-all runtime. Furthermore, as the number of agents increases so too does the complexity of the simulated annealing algorithm, further increasing the run-time. To try and demonstrate this, Figure 10.4 shows both filter runtime and combined filter/annealing runtime with 220 particles. Because each filter executes independently the run-time data is affected by operating system scheduling, making it difficult to obtain precise information. However, the general trend of the data can be identified and is plotted in the figure.

One can see from the data that both run-times appear to increase linearly. As the number of agents increases more filters are required which gradually increases the filtering run-time. The combined filtering/simulated annealing run-time increases more sharply as

the simulated annealing algorithm requires longer to identify good multi-agent behaviour assignments. It is apparent from the figure that both the number of particles and the number of agents have a large impact on runtime. In some ways the number of agents is a more concerning variable because the number of particles remains fixed and can be tuned to the environment. However, the number of agents is likely to be determined by the environment in which the approach is deployed. If real-time processing is required it is clear that the number of agents must remain capped or else some other approach is required to reduce the run-time. Some ideas for progressing in this direction in the future will be discussed in Chapter 11.

10.2.5 Complexity of Human Behaviour

One of the challenges in observing human behaviour is that behaviours can frequently change. The results in Section 8.2.2 showed that the relaxed temporal model allowed small changes in component order without affecting recognition accuracy. This is an important attribute of the approach because it shows that different instantiations of the same behaviour can be recognised via a single representation. The experimental data used for Figure 8.8 on page 159 demonstrated a single representation of *AO1* recognising two different orderings.

These results show that modelling strict temporal order is not necessarily beneficial, even though it may appear logical. For instance, one might argue that if two activities are clearly dependent (e.g. placing an object and removing it), why remove that information? However, this thesis proposes that such strong dependencies do not need to be modelled in many cases and actually reduces generality. The results in Section 8.1.2 support this proposition by the very fact that different variations of the same behaviour can still be recognised via a single definition.

Laxton *et al.* adopted the alternative approach of partially fixing temporal order in their model, where each sub-goal activity was totally ordered while individual sub-goals were unordered [76]. The limitation of their work remains as described above; they require multiple sub-goal representations in order to match different orderings. Their validation domain utilised very strict ordering constraints (cooking) and thus their approach performed well, but in a scenario with more variable orderings the bag-of-features approach might demonstrate that its more generic behaviour descriptors are beneficial.

Aside from changes to activity ordering there are also a number of behavioural com-

plexities that arise when the goals of an agent change. This can lead to the pursuit of one goal followed by another (concatenation), goal abandonment, and interleaved activities from multiple goals. These aspects of recognition apply to many domains, although it is primarily plan recognition researchers that have considered them in previous work (e.g. [100, 27, 63, 48]). Despite their relevance to automated visual surveillance, to the author's knowledge there are no prior examples of these concepts being applied within video surveillance applications.

While not the focus of this research these concepts have nevertheless remained prominent and have been considered during algorithm design. Correspondingly, the particle reset mechanism was designed to facilitate the recognition of concatenated and switched goals, while concurrent and interleaved goals remained outside the scope of this research. Recall from Chapter 4 that the particle reset mechanism is activated for particles that cannot explain an observation. If that particle has observed all features from its current behaviour hypothesis it is re-initialised according to the prior distribution which should aid the recognition of a new goal being commenced (concatenation). In contrast, if a particle has not observed all features the target behaviour is re-initialised and the set of consistent observed features is maintained. This mechanism should allow goal abandonment and switching to be detected.

Concatenated Behaviour

Focusing initially on goal concatenation, the results showed a 19% drop in mean recognising performance (F-Score: 0.73) for a concatenated (second) behaviour. This drop in performance seemed to be caused by the fact that not all particles represent the same state and thus only some of the particles were reset when a behaviour completed. Resultingly, a subset of the particles attempt to explain all observations, reducing overall filter accuracy. Having observed this phenomenon it was hypothesised that an alternative approach would be to reset all particles when a behaviour prediction occurred. It has already been observed that standard behaviour prediction achieves an F-Score of 0.92 and thus this should provide a reliable way of triggering particle reset for an entire filter.

To test this hypothesis an additional experiment was performed which highlighted a number of further issues. Firstly, because the *HO* behaviour contains the *AO2* behaviour, this ordinarily causes an *AO2* prediction to be generated, followed by a retraction and an *HO* prediction as further observations are made. However, when particles are reset at prediction this prevents the full *HO* behaviour from being recognised. Furthermore, even if one removes the *HO* behaviour from the experiment, the results show that recall falls by 14%

for the other behaviours causing an overall F-Score of 0.6 (precision remained comparable). Analysis showed that the multi-agent behaviours are responsible for this decrease and is caused by the initial (first) behaviours of each agent terminating at different points. This prevents the joint set of features from their second behaviours from being correctly identified.

If one removes all collaborative behaviours from the test an F-Score of 0.73 is obtained, although this only represents a 6% improvement if one isolates the solo behaviours from the original results. Correspondingly, it can be concluded that resetting all particles at behaviour prediction is no more effective than the original approach, which also delivered an F-Score of 0.73.

Goal Abandonment/Switching

Turning now to the recognition performance of switched behaviours, the results show that a performance drop is observed with a mean F-Score of 0.68. Again, it is not possible to directly contrast this result with previous research, although it may be discussed in relation to similar research by Geib and Goldman [48]. Rather than detecting switched behaviour their research focused exclusively upon goal abandonment, where an agent pursuing several goals ceased performing activities for one of them. Their approach evaluated the likelihood that a set of observations didn't support a goal and used a threshold termed the *probability abandonment threshold* (PAT). The PAT could be adjusted to reduce the number of false-positive 'abandoned' classifications, and achieved an optimal accuracy of 75%. Geib and Goldman remark that the length of the evaluation behaviours was one of the causes of this limited performance. Increasing the PAT should in theory have allowed the number of false positives to be reduced to zero, but as the PAT was increased the limited length of the test sequences meant that the threshold could not always be reached before the sequence terminated.

It is speculated that both the limited length, and large overlap of the behaviours, also relate to the poor recognition performance in this research. If only a relatively few activities are observed after a behaviour has switched it is difficult for the filter to correctly converge to the new behaviour, while longer sequences might have allowed this to occur.

A partial comparison can be made with Chai and Yang's work on complex behaviour recognition [27]. Their research specifically looks at recognising three types of complex behaviour: concatenation, concurrency (activities supporting multiple goals), and inter-

Approach	Single goal performance	Multi-goal performance
<i>Bag-of-features</i>	96.4%	88%
Chai and Yang [27]	94.6%	91.4%

Table 10.3: Comparing precision with Chai and Yang’s MG-Recogniser

leaving. Although interleaved goals and goal abandonment/switching are not identical, they do produce a similar effect: the switching of one behaviour to another. There are thus two similarities between our research and [27], with concurrent goals remaining completely out of scope of this research.

Chai and Yang’s work introduces the concept of behaviour suspension. That is, when a behaviour ceases to be recognised with a high probability it enters a suspended mode that may or may not be resumed at a later point. A time limiting threshold is applied such that behaviours that are not resumed after some period are classified as terminated/abandoned, while other behaviours continue to be suspended/resumed until termination is observed. They partition their experiments into two scenarios: single-goal, and multi-goal (incorporating all three types defined above), and have published the recognition accuracies shown in Table 10.3.

These results can be compared to the bag-of-features approach by merging the results for concatenation and switching to generate comparative multi-goal results. One can see from the table that the two approaches offer very similar performances, with Chai and Yang achieving a 3.4% improvement on multi-goal behaviour but are out-performed on solo-goal behaviours by 1.8%. It can be concluded that our results are aligned with related work in the field for detecting multi-goal behaviours.

However, it should also be emphasised that while results are comparable, Chai and Yang’s approach relied on training data and thus could not be applied in data-scarce domains. With this consideration in mind, the bag-of-features approach has performed very well, given that comparable performance has been achieved without with use of model training.

Superfluous Activities

A further complexity of human behaviour was also demonstrated in Figure 8.9 on page 161, which showed that superfluous activities also harm recognition performance and cause the average F-Score to drop to 0.64. One important thing to note about superfluous activities is that they consisted of pairs of repeatable features such as placing and removing

an object. Section 4.2 discussed the algorithms used to process repeatable feature subsets, and it should be highlighted that the technique was integrated with algorithms for dealing with classification errors. It is possible that these algorithms are insufficient to allow repeatable subsets, with classification error filtering altering particle densities so that the ‘repeating features’ hypotheses have insufficient weight. Indeed, a final experiment disabled repeating subsets from the algorithm, meaning all superfluous features were considered classification errors. The results showed that accuracy dropped by only 3%, suggesting these algorithmic elements were having limited effect.

10.3 Generic Application

Considering the wider context of the results several aspects should be discussed. Firstly, it is apparent from Figure 10.2 on page 195 that the approach is poorly effected by incomplete behaviour traces. It appears that this is at least in part because of the overlap between the behaviours, meaning that they can often only be distinguished from one another once all behaviour features have been observed. Having a prediction criteria that requires the observance of all features delivers very good results, but limits the approach because partial behaviours cannot be recognised. If there were less overlap between the behaviours it is possible that the prediction criteria could be relaxed and more accurate earlier/partial behaviour recognition could be achieved. Due to time constraints exploring this hypothesis must be left for future work.

None the less, these results do have implications for the wider application of the approach. For instance, digital assistants are a growing area in which behaviour recognition technology has a place. Microsoft’s Clippy was an early example of a recommender system in which an animated paper-clip attempted to recognise simple tasks so that they could be automatically performed on behalf of the user. Although Clippy is generally considered as a user interface failure, digital assistants are once again becoming popular, with Siri from Apple being a current example [5]. The integration of digital assistants with behaviour recognition techniques may again re-surface, however, a key factor of recommender systems is that the user’s behaviour is detected before their task is complete. In this respect the bag-of-features approach is clearly inappropriate unless the behaviours to be detected are quite distinct.

Although the validation has focused on indoor video surveillance this area is representative of a number of other domains. For instance, behavioural awareness is a key factor in the successful deployment of autonomous vehicles within multi-agent environments.

Autonomous vehicles are becoming increasingly common within the underwater [13], ground [77], and air domains [58] for both civilian and military applications. Providing these systems with behavioural awareness requires sensory processing and inference techniques and is not dissimilar from the video surveillance field. Video retrieval is another vibrant area of research in which bag-of-features inference may have a place [131]. If one can extract distinct events from video then videos containing particular behaviours could be automatically identified. This could be extremely useful for large archives of video data and allow filtering of complex behaviours. Inversely, the approach could also be used to automatically annotate video as it is archived.

10.4 Summary

The beginning of the chapter showed that reliable behaviour predictions can be made by using a prediction criteria that identifies when behaviours terminate. Because the approach does not model any prior information it does poorly at recognising behaviours before all features have been observed, although in-part this is likely to be due to the large overlap of behaviours.

In evaluating the approach there is evidence to support the use of our heuristic weight. The results showed us that as the number of particles increases so too does recognition performance, and tends to optimal as the number of particles approaches infinity.

One of the key aims of this research was that behaviour detections should be explainable and this has been clearly demonstrated. Explanation templates were associated with each behaviour with ‘slots’ marked for pertinent information. Upon making a prediction these templates were consulted and combined with particle filter details to provide complete behaviour descriptions. This was demonstrated with an example detection, which identified the agents involved in a behaviour and identified relevant video frames for key activities.

The approach has been compared to competing techniques using the PETS 2006 data. Optimal performance was achieved in this respect, which is comparable with several other approaches and exceeds three. Comparable results were also achieved with regards to multi-goal recognition, where agents concatenate or change their behaviour.

Finally, it has been shown that the use of combinatorial search to identify multi-goal behaviour has led to scaling issues with regards to the maximum number of agents, reach-

ing a limit of approximately twenty within this implementation. Within these bounds real-time recognition can be performed to a high degree of accuracy, but runtime is significantly affected after this point. The next chapter will consider several approaches by which this limitation might be removed in future work, in addition to suggesting several other extensions to this research.

Chapter 11

Conclusions and Future Work

Nomenclature:

$f^{\mathcal{R}1}$	Filter for agent 1
C	Set of currently achieved features
T	Set of target features
B^s	The set of solo behaviours
B_b^s	The b th behaviour in B^s

11.1 Conclusions

The goal of this research was to develop probabilistic inference techniques for high-level behaviour recognition in data scarce domains. It was the author's thesis that behaviours could be recognised without model training by using a feature based approach. To this end a generic framework has been developed and demonstrated within a surveillance application. The results have shown that good recognition performance can be achieved without requiring training data or experts to define model probabilities. The findings of this research have already been discussed in detail in the previous chapter, however, to summarise:

1. As a probabilistic inference technique the framework has been successful in recognising complex human behaviours without parameter estimation. This has been achieved by building on Rao-Blackwellised Particle Filters, combining them with a novel behaviour representation that removed the need for learnt transition probabilities. The experimental results show that the approach is robust to primitive classification errors, one of our key aims, giving a mean recognition F-Score of 0.92 when evaluating the approach using seven noisy behaviours. Further validation was provided by means of the video-processing test-harness that employed both publicly available and newly gathered video data of the same seven behaviours. A mean F-Score of 0.81 was achieved on the combined video datasets. These results support the thesis that complex behaviour can be recognised using a feature based approach.
2. A further objective of this research was the maintenance of low false-positive prediction rates. This is another area in which the results have demonstrated great success with a mean false-positive rate of only 3.4% on the combined video data, and 0.31% on the simulated data. Furthermore, these rates have not been achieved at the expense of recall, which still remains at 0.77 and 0.88 for video and simulated data respectively.
3. With respect to the goal of real-time inference, the evaluation has shown that the framework can achieve real-time inference, although under the current implementation this is only possible when restricting the number of concurrently monitored agents to twenty. This limitation is a result of multi-agent behaviour detection, which currently relies on heuristic combinatorial search.
4. A further objective of this research was to develop approaches that could explain detected behaviour in order to facilitate their integration into real-world environments. Our video surveillance test-harness has shown that the approach performs well in this respect, allowing key information such as activity detection times to be extracted from the particle filters and presented in an explanatory manner.
5. At the very beginning of this thesis it was highlighted that the ability to recognise behaviour is dependent upon several skills: the ability to sense the environment, the ability to infer action purpose and the ability to apply knowledge. This research has focused on the last of these skills, and is thus highly dependent on other systems to provide low-level sensing and interpretation. It was for this reason that one of the aims of the thesis was to understand the relationship between low-level errors and high-level reasoning. The evaluation has shown that poor recall in particular has an adverse effect on complex behaviour recognition, where the framework lacks the ability to infer that activities have been ‘missed’ by the detectors. High-level performance does however adapt well, with a linear drop in performance as low-level precision reduces. As one would expect, good low-level detection accuracy is key

to high-level inference, although the results indicate that good (≥ 0.8) performance can still be achieved with relatively high low-level false-positives (20%).

6. Abandoned object detection is one area where the results can be compared with competing approaches. Bag-of-features inference offers comparable performance on this task (compared with others using the PETS data), although it should be noted that unlike many competing approaches, bag-of-features inference can recognise many different behaviours and is thus a more generic approach.
7. Inference was demonstrated in the data-scarce domain of visual surveillance, and it has been shown that our novel framework can be used to provide reliable behaviour recognition without parameter learning. In prior work automated visual surveillance is one domain in which high-level models have been unable to utilise probabilistic techniques. This has primarily led to two competing approaches; anomaly detection and constraint satisfaction. Anomaly detection is focused on flagging abnormal behaviours without comprehending their meaning. This is quite different from the bag-of-features approach which reasons about the semantic components of known behaviours. However, it is recognised that each approach tackles different but highly related problems. In reality it is likely that a combination of these approaches would be most useful in real-world scenarios, harnessing the benefits of each methodology.

Semantic constraint satisfaction uses complex spatio-temporal relationships to define behaviours but are unable to reason about uncertainty. This is important not only because low-level signal processing algorithms generate classification errors, but because variability makes human behaviour inherently uncertain. The mechanisms for dealing with uncertainty are not traditionally available with semantic rules and thus one would expect bag-of-features inference to out-perform them on most occasions. A lack of published results means that a direct comparison of the approaches cannot currently be drawn, although this would provide interesting future work. However, one can summarise the benefits of our framework over anomaly detection and semantic rules by stating that bag-of-features inference can recognise and explain detections of specific behaviours in uncertain, noisy domains, while competing approach cannot.

8. The approach has been successful at recognising multi-agent behaviours in addition to single agent behaviours. This is in great contrast to the majority of behaviour recognition research where there has been very little exploration of multi-agent behaviour recognition. In prior research multi-agent behaviours have generally been restricted to military manoeuvres and formations (e.g. [134, 136]) for which spatial pattern recognition techniques have been employed. Within this thesis multi-agent behaviours have been recognised by considering the semantic relationships between the activities of different agents.

9. One of the ‘extended’ aims of this research was to recognise interleaved, switched and concatenated behaviour. In this respect the framework delivered performance that is comparable with other work in the field with a mean F-Score of 0.71 when detecting multi-goal behaviours. Success was also demonstrated in recognising different activity orderings of the behaviours, making the approach more robust than fixing temporal order.
10. As a whole, this research has noted that high-level behaviour recognition research is particularly effected by a lack of realistic, publicly available data, with many datasets lacking the richness to expose complex behaviour. Additionally, the use of video data means that video processing techniques have to be reimplemented and this distracts research from the overall goal. Future work is likely to yield greater advances if it is undertaken within a wider research setting in which specialists from different fields contribute directly towards a common goal.

11.2 Future Work

The time-constraints of this research mean that many questions remain open for future work. There are also several theoretical challenges yet to be solved, such as how the approach can be adapted to scale better with the number of agents, and how repeatable behaviour might be modelled. To this end this section will identify several directions in which future work might proceed.

11.2.1 Collaborator Detection

As it stands collaborator detection is fundamentally reliant upon combinatorial optimisation. This causes two distinct problems. Firstly, a particle filter hierarchy is required for each set of potential collaborators which causes exponential growth as the number of agents increases. The results showed that under the experimental conditions real-time performance could only be achieved for up to approximately twenty agents. The second problem with combinatorial optimisation is that the simulated annealing algorithm finds good behaviour/collaborator assignments at each observation. Because the algorithm considers all agents as potential collaborators the runtime increases with the number of agents.

If one makes the assumption that broad temporal behaviours are out of scope for multi-

agent behaviour detection (e.g. two agents who enter the scene one hour apart), then logically, not all agents need to be considered as potential multi-agent behaviour participants (collaborators). One could thus use temporal rules to inform collaborator detection and reduce the scope of the combinatorial problem. However, this is still fundamentally limiting and perhaps better methods can be derived by considering the cognitive processes of a human observer. For instance, a human would probably consider agents as non-collaborators until there is evidence to the contrary. Evidence might be in the form of direct interaction (e.g. talking, shaking hands) or inferred (e.g. dual manipulation of same object), and could also be applied to the inference algorithm to trigger new filter hierarchies. This approach would reduce the filter growth rates shown in Chapter 8, but is reliant upon the presence of suitable triggers being available.

Another approach for reducing potential collaborators would be to only consider agents concurrently present in the scene. If agent A leaves the scene before agent B arrives the two agents would be considered independent, although the return of A would then initiate a potential collaboration. Again, the approach has the ability to significantly reduce the filter growth rates, but is restrictive of the behaviours that can be modelled. For instance, the HO behaviour could not be detected using this strategy.

One final approach for limiting this complexity is to use high-probability predictions to reduce the space of potential collaborators. Suppose that there are three agents being observed, agents 1, 2 and 3, leading to solo filters $\{f^{\mathcal{R}1}, f^{\mathcal{R}2}, f^{\mathcal{R}3}\}$ and multi-agent filters $\{f^{\mathcal{R}1,2}, f^{\mathcal{R}1,3}, f^{\mathcal{R}2,3}\}$. Suppose that at time t a prediction is made that agent 1 is performing solo behaviour B_b^s , and that all features for B_b^s have been observed with high probability. This prediction could be used as a heuristic to eliminate unlikely filters such as $\{f^{\mathcal{R}1,2}, f^{\mathcal{R}1,3}\}$, and further, can be used to prevent the generation of further multi-agent filters as additional agents arrive. Thus, if agent 4 is observed after t , only filters $\{f^{\mathcal{R}4}, f^{\mathcal{R}2,4}, f^{\mathcal{R}3,4}\}$ would be generated, but not $f^{\mathcal{R}1,4}$. It should be clear that such an approach is heuristic and is not guaranteed to produce optimal results. The largest danger is that B_b^s is incorrect, and thus the removal of other multi-agent filters could prevent the correct detection of a multi-agent behaviour involving the agent after time t .

11.2.2 Repetitive Behaviour

One of the assumptions of our approach is that behaviours do not contain repetitive components such as repeated sub-goals. This simplifying assumption was made so that one did not have to consider multiple occurrences of the same feature in the ‘target’ and ‘currently achieved’ feature sets. In some ways this assumption is overly restrictive because

the real issue is with regards to components that can be repeated an infinite or unknown number of times. This is because uniform probability is assigned to elements in $C \setminus T$, which is undefined when an element of $C \setminus T$ can be repeated an infinite number of times.

There are a number of methods by which the assumption might be relaxed. Components that have a fixed repetition count are the easiest to accommodate, where each component could simply be identified uniquely (e.g. *FeatureA1*, *FeatureA2*). Some progress could also be made for behaviours where components are repeated an unknown, but limited number of times, with several instances of the behaviour being defined with a different number of component repetitions. This approach grows with complexity as the number of potential repetitions increases and is exponential with the number of repetitious features.

An infinitely repetitious feature is the most challenging to incorporate. If the distribution of repetitions could be approximated then one could incorporate this distribution into the model. For instance, if the probability of *FeatureA* occurring twice is 0.75, and the probability of it occurring once is 0.25, if $T = \{\textit{FeatureA}, \textit{FeatureB}\}$ and $C = \emptyset$, then $P(\textit{FeatureA}|T, C) = 0.66 * 0.75 + 0.5 * 0.25 = 0.625$ and $P(\textit{FeatureB}|T, C) = 0.33 * 0.75 + 0.5 * 0.25 = 0.375$. This distribution is also more scalable than the previous approach for unknown but limited repetition and could also be applied, in addition to solving the simplest case of a known number of repeated instances. In all cases the ability to distinguish multiple instances in C is still required, although this is a trivial task. This last approach is preferred by the author, but does require the distribution of occurrences to be reasonably approximated.

11.2.3 Active Vision

This research has specifically focused on high-level inference, and as such the integration between high-level inference and low-level vision algorithms has been limited. However, significant performance gains might be achieved through a feedback loop that focuses video-processing resources in accordance with expected future activities. For instance, in a crowded surveillance application one might detect that a bag is being abandoned, and thus focus more sensors/processing time on tracking the offending bag owner. In more complex scenarios one might even predict future actions, and swap/prioritise different sensing algorithms that are optimised for the detection of these actions.

11.2.4 Ontology Integration

Another direction in which this research could be extended would be to use ontologies to provide a taxonomy framework upon which behaviours could be generalised. For instance, suppose that two behaviours are to be detected : ‘abandoned object’ and ‘abandoned vehicle’. In essence these two behaviours are very similar, with the most apparent difference being that one tends to consider an object as being reasonably small and a vehicle as being reasonably large. However, in the context of these behaviours the dimensions of an object are not necessarily as important as the objects’ attributes, such as their ability to ‘conceal’. Both a vehicle and suitcase could be explicitly defined as having such a property, although it may also be possible to infer such an attribute using languages such as OWL: the Web Ontology Language [95]. More generic behaviours could then be defined by referring to ontological concepts rather than specific real-world objects. An ontology could then be integrated into the framework between the video processing layers and the reasoning layers to allow inferred events to be provided as observations.

11.2.5 Comparing Approaches

A disparity between the datasets of published literature means that a direct comparison for many of the results has not been possible. However, it is expected that the probabilistic nature of bag-of-features inference means that they will out-perform semantic rule-based approaches (constraint satisfaction) such as [46], and this is one area in which future work could provide compelling evidence.

11.2.6 Behaviour Similarity

The experiments identified a number of results which pertain to the similarity of behaviours. It was hypothesised that both the similarity and length particularly impacted the recognition of switched behaviours, where one would logically expect that less similarity should aid successful identification. Additionally, fewer similarities should also facilitate more accurate early prediction and reduce false-positive predictions. A more theoretical analysis of these factors using controlled behaviours could help to identify these relationships more clearly and better define the behaviours with which the approach works best.

11.2.7 Video Event Detectors

Validating the approach called for the implementation of several low-level video processing algorithms. Because the implementation time for these algorithms was strictly limited by the overall goal of high-level recognition there are several areas where the video processing could be improved. The simple event detection modules in particular are naive in their approach, with group detection entirely based upon the proximity of individually tracked agents. Papadourakis and Argyros argue that groups of agents should be identified and tracked as part of the tracking process itself, rather than separating tracks and then trying to recognise groups [113]. It is anticipated that integrating better group tracking algorithms into the framework would significantly improve video recognition performance.

Furthermore, this research utilised an experimental person tracker upon which large improvements could be made. The tracker was observed to perform particularly badly at short range where track precision reduced. This has implications for object association where ownership is determined via the distance of objects to people. This in turn affected the recognition performance of the *WI* and *AO1* behaviours, which rely on accurate object associations. Although object tracking is a very active subject within video processing research there are relatively few trackers available to the wider research community and this has a significant impact on high-level video recognition research.

11.2.8 Standardised Dataset

Significant benefit could be obtained from the generation of a large, publicly available dataset for high-level behaviour research. The Performance Evaluation of Tracking and Surveillance (PETS) workshops have made great strides in this area for lower-level tracking researchers and have released numerous datasets (e.g. [139, 45, 44]). However, this research has shown that the limited presence of complex behaviours within [139] limits its use for complex recognition research, and was similarly observed with [44]. On the other hand, [45] is extremely complex at the tracking level and is thus too challenging for higher level recognition without the availability of good tracking information.

Aside from the PETS workshops, other datasets in a similar vein also exist, with the BEHAVE [1] and CAVIAR [2] datasets being two other examples of publicly available data useful for complex behaviour recognition research. For this project these datasets still lacked a sufficient level of complexity, focusing more on interacting behaviours such as fighting.

One can thus conclude that the availability of a high-level dataset, especially one that already includes tracking information, would be of substantial benefit to high-level behaviour recognition research.

11.3 Closing Remarks

Bag-of-features inference has both been inspired by and built upon different techniques from a large number of disciplines. Yet as an approach for behaviour recognition, it is strikingly different from related work. Bag-of-features inference has been born out of the need for high-level inference in real and challenging environments, where prior research has had limited success. Its key benefit is that training data is not required, yet it is able to perform probabilistic inference in challenging domains. This thesis has shown that many of the limitations of prior work can be overcome by adopting our novel approach, which has demonstrated very promising results on both real and synthetic data. Although significant work still remains, these results have brought us one step closer to the autonomous systems that have captured the human imagination for the past forty years, and deployable systems that can recognise complex human behaviour in real-world environments may soon become a reality.

Appendix A

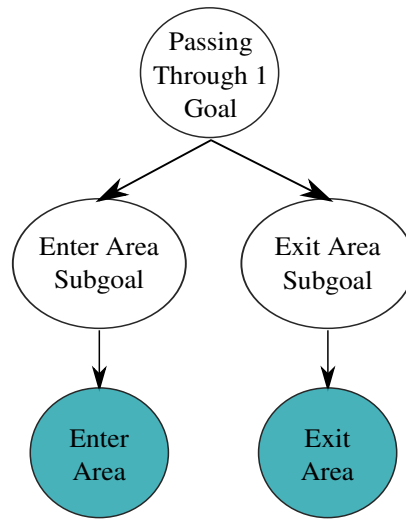


Figure A.1: Structure of the Passing Through 1 behaviour

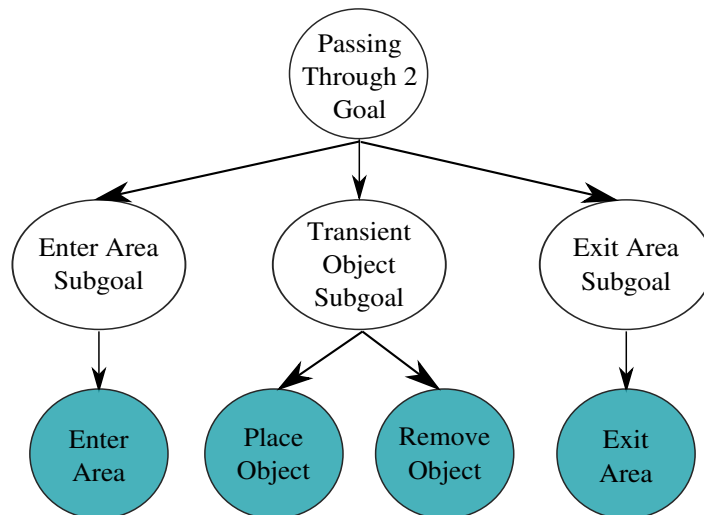


Figure A.2: Structure of the Passing Through 2 behaviour

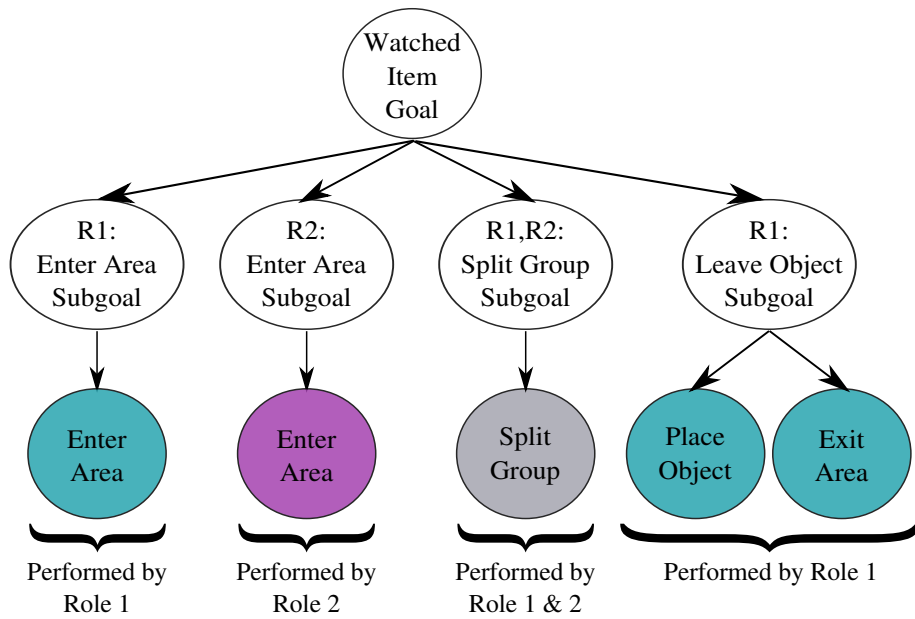


Figure A.3: Structure of the Watched Item behaviour

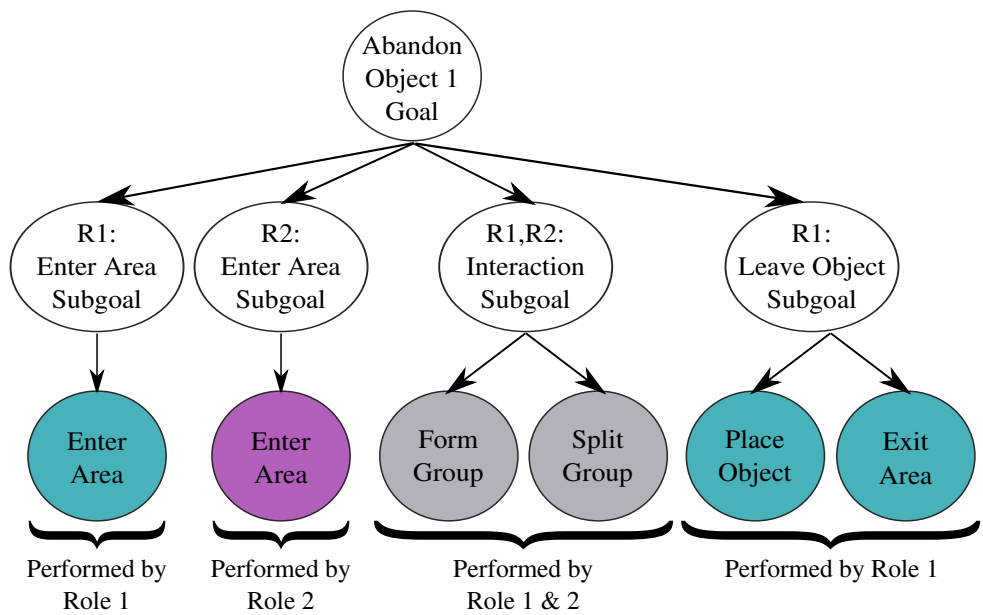


Figure A.4: Structure of the Abandon Object 1 behaviour

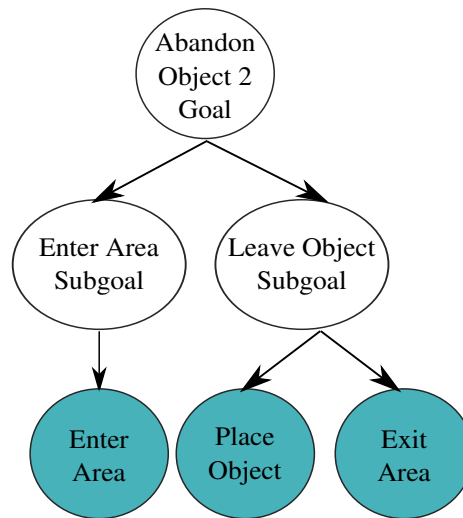


Figure A.5: Structure of the Abandon Object 2 behaviour

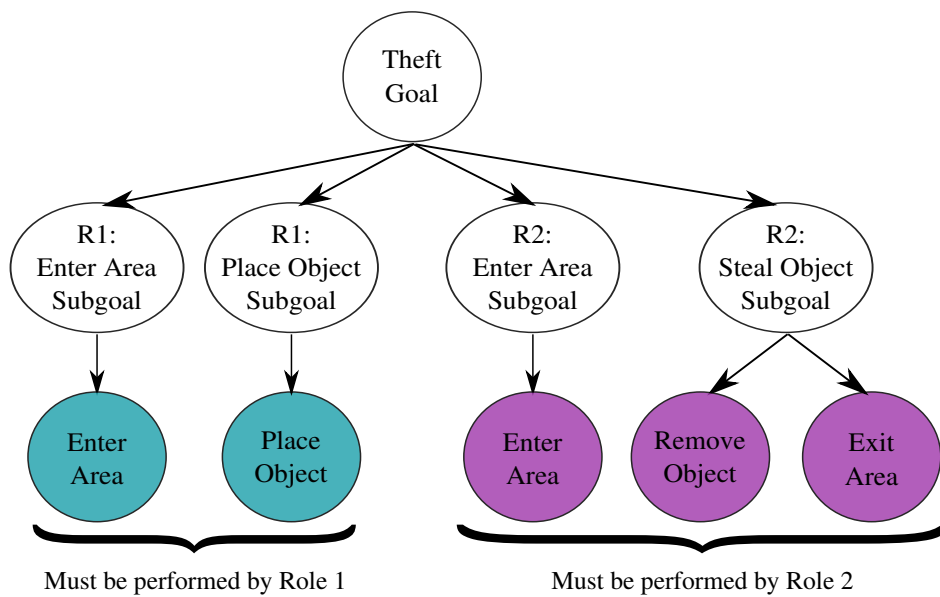


Figure A.6: Structure of the Theft behaviour

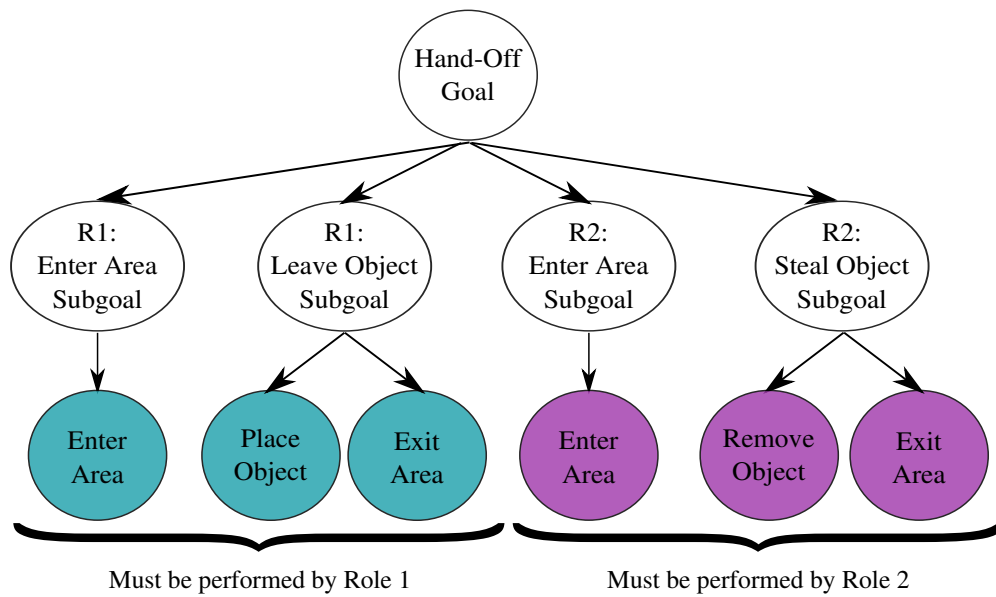


Figure A.7: Structure of the Hand-Off behaviour

Bibliography

- [1] Behave: Computer-assisted prescreening of video streams for unusual activities. URL <http://homepages.inf.ed.ac.uk/rbf/BEHAVE/>. Edinburgh University Informatics Department (2007)
- [2] Caviar: Context aware vision using image-based active recognition. URL <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>. Edinburgh University Informatics Department (2002)
- [3] Collins english dictionary: Complete and unabridged. URL <http://www.collinslanguage.com/>. HarperCollins Publishers (2003)
- [4] Executive summary on 2005 missouri vehicle stops. URL <http://ago.mo.gov/racialprofiling/2005/racialprofiling2005.htm>. Missouri Attorney General (2005)
- [5] Siri. URL <http://www.apple.com/iphone/features/siri.html>. Apple (2012)
- [6] U.k. ministry of defence: Defence technology plan. URL http://www.science.mod.uk/strategy/dtplan/technologies_default.aspx. UK Ministry of Defence (2011)
- [7] J. Aguilera, H. Wildenauer, M. Kampel, M. Borg, D. Thirde, and J. Ferryman. Evaluation of motion segmentation quality for aircraft activity surveillance. In 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 293 – 300 (2005)
- [8] P. Antonakaki, D. Kosmopoulos, and S. Perantonis. Detecting abnormal human behaviour using multiple cameras. *Signal Processing*, 89(9), 1723–1738 (2009)
- [9] M. Arulampalam, S. Maskell, N. Gordon, T. Clapp, D. Sci, T. Organ, and S. Adelaide. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2), 174–188 (2002)

- [10] E. Auvinet, E. Grossmann, C. Rougier, M. Dahmane, and J. Meunier. Left-luggage detection using homographies and simple heuristics. In Proceedings of the 9th IEEE International Workshop on Performance Evaluation in Tracking and Surveillance, 51–58 (2006)
- [11] D. Ayers and R. Chellappa. Scenario recognition from video using a hierarchy of dynamic belief networks. In Proceedings of the 15th International Conference on Pattern Recognition, volume 1, 835–838 (2000)
- [12] A. Basharat, A. Gritai, and M. Shah. Learning object motion patterns for anomaly detection and improved object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–8 (2008)
- [13] R. Baxter, J. Cartwright, J. Clay, O. Clert, B. Davis, J. Lopez, F. Maurelli, Y. Petillot, P. Patron, and N. Valeyrie. Nessie v turbo : a new hover and power slide capable torpedo shaped auv for survey, inspection and intervention. In Proceedings of The Association for Unmanned Vehicle Systems International Conference (2010)
- [14] S. Beauchemin and J. Barron. The computation of optical flow. ACM Computing Surveys (CSUR), 27(3), 433–466 (1995)
- [15] M. Beetz, N. v. Hoyningen-Huene, J. Bandouch, B. Kirchlechner, S. Gedikli, and A. Maldonado. Camera-based observation of football games for analyzing multi-agent activities. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, 42–49 (2006)
- [16] N. Bird, O. Masoud, N. Papanikolopoulos, and A. Isaacs. Detection of loitering individuals in public transportation areas. IEEE Transactions on Intelligent Transportation Systems, 6(2), 167–177 (2005)
- [17] M. Black and A. Jepson. A probabilistic framework for matching temporal trajectories:condensation-based recognition of gestures and expressions. In Proceedings of the European Conference on Computer Vision, volume 1, 909–924 (1998)
- [18] N. Blaylock and J. Allen. Hierarchical instantiated goal recognition. In Proceedings of the AAAI Workshop on Modeling Others from Observations, 8–15 (2006)
- [19] N. Blaylock and J. F. Allen. Fast hierarchical goal schema recognition. In Proceedings of the National Conference on Artificial Intelligence, volume 21, 796–801 (2006)
- [20] N. J. Blaylock. Towards Tractable Agent-based Dialogue. Ph.D. thesis, University of Rochester (2005). P. 133-136

- [21] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 993–1022 (2003)
- [22] O. Boiman and M. Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74(1), 17–31 (2007)
- [23] B. Bose, X. Wang, and E. Grimson. Multi-class object tracking algorithm that handles fragmentation and grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–8 (2007)
- [24] H. H. Bui and S. Venkatesh. Policy recognition in the abstract hidden markov model. *Journal of Artificial Intelligence Research*, 17(1), 451–499 (2002)
- [25] J. Candamo, M. Shreve, D. Goldgof, D. Sapper, and R. Kasturi. Understanding transit scenes: A survey on human behavior-recognition algorithms. *IEEE Transactions on Intelligent Transportation Systems*, 11(1), 206–224 (2010)
- [26] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings on Radar, Sonar and Navigation*, 146(1), 2–7 (1999)
- [27] X. Chai and Q. Yang. Multiple-goal recognition from low-level signals. In *Proceedings of the National Conference on Artificial Intelligence*, 1, 3–8 (2005)
- [28] Y.-L. Chen, B.-F. Wu, H.-Y. Huang, and C.-J. Fan. A real-time vision system for nighttime vehicle detection and traffic surveillance. *IEEE Transactions on Industrial Electronics*, 58(5), 2030–2044 (2011)
- [29] L. Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2), 211–218 (1991)
- [30] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4), 271–288 (1998)
- [31] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proceedings of the Workshop on Statistical Learning in Computer Vision*, volume 1, 1–22 (2004)
- [32] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10), 1337–1342 (2003)
- [33] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational intelligence*, 5(2), 142–150 (1990)
- [34] H. Dee and D. Hogg. Detecting inexplicable behaviour. In *Proceedings of the British Machine Vision Conference*, 477–486 (2004)

- [35] H. Dee and D. Hogg. On the feasibility of using a cognitive model to filter surveillance data. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*, 34–39 (2005)
- [36] H. Dee and S. Velastin. How close are we to solving the problem of automated visual surveillance? *Machine Vision and Applications*, 19(5-6), 329–343 (2008)
- [37] S. M. I. Dong Zhang; Gatica-Perez, D.; Bengio. Modeling individual and group actions in meetings with layered hmms. *IEEE Transactions on Multimedia*, 8(3), 509–520 (2006)
- [38] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 176–183 (2000)
- [39] A. Doucet, S. J. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3), 197–208 (2000)
- [40] A. Doucet and A. Johansen. *A tutorial on particle filtering and smoothing: Fifteen years later*. Oxford University Press (2009)
- [41] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proceedings of the 6th European Conference on Computer Vision*, volume 2, 751–767 (2000)
- [42] M. Endsley. Toward a theory of situation awareness in dynamic systems: Situation awareness. *Human factors*, 37(1), 32–64 (1995)
- [43] T.-J. Fan, G. Medioni, and R. Nevatia. Recognizing 3-d objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11), 1140–1157 (1989)
- [44] J. Ferryman and A. Shahrokni. An overview of the pets 2009 challenge. In *Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 25–30. IEEE (2009)
- [45] J. Ferryman and D. Tweed. An overview of the pets 2007 dataset. In *Proceeding of the Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 49–53 (2007)
- [46] F. Fusier, V. Valentin, F. Brémond, M. Thonnat, M. Borg, D. Thirde, and J. Ferryman. Video understanding for complex activity recognition. *Machine Vision and Applications*, 18, 167–188 (2007)
- [47] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1), 82–98 (1999)

- [48] C. Geib and R. Goldman. Recognizing plan/goal abandonment. In Proceedings of the International Joint Conference on Artificial Intelligence, volume 18, 1515–1517 (2003)
- [49] C. Geib, J. Maraist, and R. Goldman. A new probabilistic plan recognition algorithm based on string rewriting. In Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, 91–98 (2008)
- [50] C. W. Geib. Assessing the complexity of plan recognition. In Proceedings of The National Conference on Artificial Intelligence, 507–512 (2004)
- [51] C. W. Geib and R. P. Goldman. Plan recognition in intrusion detection systems. In DARPA Information Survivability Conference and Exposition (DISCEX II) (2001)
- [52] C. W. Geib and M. Steedman. On natural language processing and plan recognition. In Proceedings of the International Joint Conference on AI, 1612–1617 (2007)
- [53] M. Ghallab. On chronicles: Representation, on-line recognition and learning. In Proceedings of 5th International Conference on Principles of Knowledge Representation and Reasoning, 597–607 (1996)
- [54] N. J. Gordon, D. J. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. IEE Proceedings of Radar and Signal Processing, 140(2), 107–113 (1993)
- [55] H. Grabner, P. M. Roth, M. Grabner, and H. Bischof. Autonomous learning of a robust background model for change detection. In Proceedings of 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, 39–46 (2006)
- [56] S. Guler and M. Farrow. Abandoned object detection in crowded places. In Proceedings of the 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, 18–23 (2006)
- [57] A. Hakeem and M. Shah. Learning, detection and representation of multi-agent events in videos. Artificial Intelligence, 171(8-9), 586–605 (2007)
- [58] F. Heintz, P. Rudol, and P. Doherty. From images to traffic behavior - a uav tracking and monitoring application. In Proceedings of the 10th International Conference on Information Fusion, 1–8 (2007)
- [59] T. Hofmann. Probabilistic latent semantic indexing. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 50–57 (1999)

- [60] T. Horprasert, D. Harwood, and L. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In Proceedings of the IEEE International Conference on Computer Vision, volume 99, 256–261 (1999)
- [61] C. J. Howard, T. Troscianko, I. D. Gilchrist, A. Behera, and D. C. Hogg. Searching for threat: factors determining performance during CCTV monitoring. Maastricht, the Netherlands: Shaker Publishing (2008)
- [62] D. Hu, X. Zhang, J. Yin, V. Zheng, and Q. Yang. Abnormal activity recognition based on hdp-hmm models. In Proceedings of the 21st International Joint Conference on Artificial Intelligence, 1715–1720 (2009)
- [63] D. H. Hu and Q. Yang. Cigar: Concurrent and interleaving goal and activity recognition. In Proceedings of the 23rd national conference on Artificial intelligence, 1363–1368 (2008)
- [64] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 34(3), 334–352 (2004)
- [65] S. Intille and A. Bobick. A framework for recognizing multi-agent action from visual evidence. In Proceedings of the sixteenth national conference on Artificial intelligence, 518–525 (1999)
- [66] P. Jarvis, T. Lunt, and K. Myers. Identifying terrorist activity with ai plan recognition technology. AI MAGAZINE, 26(3), 73 (2005)
- [67] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. International Journal of Computer Vision, 1, 321–331 (1988)
- [68] H. Kautz and J. Allen. Generalized plan recognition. In Proceedings of the Fifth National Conference on Artificial Intelligence, 32–38 (1986)
- [69] H. A. Kautz. A Formal Theory of Plan Recognition and its Implementation, 69–125. Morgan Kaufmann (1991)
- [70] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. Science, 220(4598), 671 (1983)
- [71] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. Journal of Computational and Graphical Statistics, 5(1), 1–25 (1996)
- [72] T. Ko. A survey on behavior analysis in video surveillance for homeland security applications. In Proceedings of the 37th IEEE Applied Imagery Pattern Recognition Workshop, 1–8 (2009)

- [73] N. Krahnstoever, P. Tu, T. Sebastian, A. Perera, and R. Collins. Multi-view detection and tracking of travelers and luggage in mass transit environments. In Proceedings of the Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, 67–74 (2006)
- [74] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1446–1453 (2009)
- [75] G. Lavee, E. Rivlin, and M. Rudzsky. Understanding video events: a survey of methods for automatic interpretation of semantic occurrences in video. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 39(5), 489–504 (2009)
- [76] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–8 (2007)
- [77] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, *et al.* A perception-driven autonomous urban vehicle. In M. Buehler, K. Iagnemma, and S. Singh (editors), The DARPA Urban Challenge, volume 56 of *Springer Tracts in Advanced Robotics*, 163–230. Springer Publishing (2009)
- [78] N. Lesh. Scalable and Adaptive Goal Recognition. Ph.D. thesis, University of Washington (1998)
- [79] N. Lesh and O. Etzioni. Scaling up goal recognition. In Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, 244–255 (1996)
- [80] L. Li, R. Luo, R. Ma, W. Huang, and K. Leman. Evaluation of an ivs system for abandoned object detection on pets 2006 datasets. In Proceedings of the 9th IEEE International Workshop on Performance Evaluation in Tracking and Surveillance, 91–98 (2006)
- [81] Y.-B. Li, T.-X. Jiang, Z.-H. Qiao, and H.-J. Qian. General methods and development actuality of gait recognition. In Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition, volume 3, 1333–1340 (2007)
- [82] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. The International Journal of Robotics Research, 26(1), 119–134 (2007)
- [83] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. Artificial Intelligence, 171(5-6), 311–331 (2007)

- [84] W. Limprasert. People detection and tracking with a static camera. Technical report, School of Mathematical and Computer Sciences, Heriot-Watt University (2010)
- [85] D. F. Lin Liao and H. Kautz. Location-based activity recognition using relational markov networks. In Proceedings of the Nineteenth International Conference on Artificial Intelligence, 773–778 (2005)
- [86] A. Lipton, H. Fujiyoshi, and R. Patil. Moving target classification and tracking from real-time video. In Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision, 8–14 (1998)
- [87] J. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. Journal of the American statistical association, 93(443), 1032–1044 (1998)
- [88] Z. Liu and C. Liu. A hybrid color and frequency features method for face recognition. IEEE Transactions on Image Processing, 17(10), 1975–1980 (2008)
- [89] D. Lowe. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, volume 2, 1150–1157 (1999)
- [90] D. G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, volume 2, 1150–1157 (1999)
- [91] F. Lv, X. Song, B. Wu, V. Kumar, and S. R. Nevatia. Left luggage detection using bayesian inference. In Proceedings of the Ninth IEEE Workshop on Performance Evaluation of Tracking and Surveillance, 83–90 (2006)
- [92] L. Manevitz and M. Yousef. One-class svms for document classification. The Journal of Machine Learning Research, 2, 139–154 (2002)
- [93] J. S. Marques, P. M. Jorge, A. J. Abrantes, and J. M. Lemos. Tracking groups of pedestrians in video sequences. In Conference on Computer Vision and Pattern Recognition, volume 9, 101–108 (2003)
- [94] J. Martínez-del Rincón, J. Herrero-Jaraba, J. Gómez, and C. Orrite-Uruñuela. Automatic left luggage detection and tracking using multi-camera ukf. In Proceedings of the 9th IEEE International Workshop on Performance Evaluation in Tracking and Surveillance, 59–66 (2006)
- [95] D. McGuinness, F. Van Harmelen, *et al.* Owl web ontology language overview. URL <http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>. (2009)
- [96] S. J. Mckenna, S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld. Tracking groups of people. Computer Vision and Image Understanding, 80, 42–56 (2000)

- [97] A. Mecocci and M. Pannozzo. A completely autonomous system that learns anomalous movements in advanced videosurveillance applications. In Proceedings of the IEEE International Conference on Image Processing, volume 2, 586–589 (2005)
- [98] A. Mecocci, M. Pannozzo, and A. Fumarola. Automatic detection of anomalous behavioural events for advanced real-time video surveillance. In Proceedings of the IEEE International Symposium on Computational Intelligence for Measurement Systems and Applications, 187–192 (2003)
- [99] M. Mitchell. An introduction to genetic algorithms. ISBN-10: 0262631857. The MIT press (1998)
- [100] J. Modayil, T. Bai, and H. Kautz. Improving the recognition of interleaved activities. In Proceedings of the 10th international conference on Ubiquitous computing, 40–43 (2008)
- [101] K. P. Murphy. Dynamic Bayesian networks: representation, inference and learning. Ph.D. thesis, UC Berkeley, Computer Science Division (2002)
- [102] J. A. Nate Blaylock. Corpus-based, statistical goal recognition. In Proceedings of the 18th international joint conference on Artificial intelligence (2003)
- [103] J. A. Nate Blaylock. Statistical goal parameter recognition. In Proceedings of the International Conference on Automated Planning and Scheduling, 297–305 (2004)
- [104] J. A. Nate Blaylock. Recognizing instantiated goals using statistical methods. In Proceedings of the Workshop on Modeling Others from Observations, 79–86 (2005)
- [105] J. A. Nate Blaylock. Statistical goal parameter recognition. In Proceedings of the 14th International Conference on Automated Planning and Scheduling, 297–305 (2005)
- [106] R. Neil M and R. Ian D. Automatic reasoning about causal events in surveillance video. EURASIP Journal on Image and Video Processing, 2011 (2011)
- [107] N. T. Nguyen, H. H. Bui, S. Venkatesh, and G. West. Recognising and monitoring high-level behaviours in complex spatial environments. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 620–625 (2003)
- [108] N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. In Computer Vision and Pattern Recognition, volume 2, 955–960 (2005)

- [109] J. 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)
- [110] C. Norris and G. Armstrong. *The unforgiving eye: Cctv surveillance in public space*. Technical report, Hull University (1997)
- [111] C. Norris and G. Armstrong. *The maximum surveillance society: The rise of CCTV*. ISBN-10: 1859732267. Berg Publishers (1999)
- [112] N. M. Oliver, A. Garg, and E. Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 1(2), 163–180 (2002)
- [113] V. Papadourakis and A. Argyros. Multiple objects tracking in the presence of long-term occlusions. *Computer Vision and Image Understanding*, 114(7), 835–846 (2010)
- [114] N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3), 266–280 (2000)
- [115] A. Patron, E. Sommerlade, and I. Reid. Action recognition using shared motion parts. In *Proceedings of the 8th International Workshop on Visual Surveillance* (2008)
- [116] M. Piccardi. Background subtraction techniques: a review. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 4, 3099–3104 (2004)
- [117] D. Poole. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1), 81–129 (1993)
- [118] D. V. Pynadath. *Probabilistic Grammars for Plan Recognition*. Ph.D. thesis, University of Michigan (1999)
- [119] D. V. Pynadath and M. P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 507–514 (2000)
- [120] C. M. R. Goldman, C.W. Geib. A new model of plan recognition. In *Proceedings of the fifteenth conference on Uncertainty in Artificial Intelligence*, 245–254 (1999)
- [121] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, 257–286 (1989)

- [122] R. N. Ramprasad Polana and A. Nelson. Low level recognition of human motion (or how to get your man without finding his body parts). In Proceedings of the IEEE Computer Society Workshop on Motion of Non-Rigid and Articulated Objects, 77–82 (1994)
- [123] N. Ravi, N. D, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence, volume 3, 1541–1546 (2005)
- [124] N. Robertson, I. Reid, and M. Brady. Automatic human behaviour recognition and explanation for cctv video surveillance. *Security Journal*, 21(3), 173–188 (2008)
- [125] A. Sadilek and H. Kautz. Recognizing multi-agent activities from gps data. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)
- [126] S. Sarkar, P. Phillips, Z. Liu, I. Vega, P. Grother, and K. Bowyer. The humanid gait challenge problem: data sets, performance, and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2), 162–177 (2005)
- [127] G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2), 461–464 (1978)
- [128] S. Skiena. The algorithm design manual. ISBN-10: 1848000693. Springer (1998). Chapter 5
- [129] G. Smith. Behind the screens: Examining constructions of deviance and informal practices among cctv control room operators in the uk. *Surveillance & Society*, 2(2/3), 376–395 (2002)
- [130] K. Smith, P. Quelhas, and D. Gatica-Perez. Detecting abandoned luggage items in a public space. In Proceedings of the 9th IEEE International Workshop on Performance Evaluation in Tracking and Surveillance, 75–82 (2006)
- [131] C. G. M. Snoek and M. Worring. Concept-based video retrieval. *Foundations and Trends in Information Retrieval*, 4(2), 215–322 (2009)
- [132] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2, 2246–2252 (1999)
- [133] G. Sukthankar, M. Mandel, K. Sycara, and J. Hodgins. Modeling physical variability for synthetic MOUT agents. In Proceedings of the Conference on Behavior Representation in Modeling and Simulation (2004)
- [134] G. Sukthankar and K. Sycara. Robust recognition of physical team behaviors using spatio-temporal models. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, 638–645 (2006)

- [135] G. Sukthankar and K. Sycara. Policy recognition for multi-player tactical scenarios. In Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, 1–8 (2007)
- [136] G. Sukthankar and K. Sycara. Hypothesis pruning and ranking for large plan recognition problems. In Proceedings of the 23rd national conference on Artificial intelligence, volume 2, 998–1003 (2008)
- [137] M. Swain and D. Ballard. Color indexing. International journal of computer vision, 7(1), 11–32 (1991)
- [138] J. Tao and Y.-P. Tan. Color appearance-based approach to robust tracking and recognition of multiple people. In Proceedings of the Fourth International Conference on Information, Communications and Signal Processing, volume 1, 95–99 (2003)
- [139] D. Thirde, L. Li, and J. Ferryman. An overview of the pets 2006 dataset. In Proceedings of the International Workshop on Performance Evaluation of Tracking and Surveillance, 47–50 (2006)
- [140] B. Tickner and E. Poulton. Monitoring up to 16 synthetic television pictures showing a great deal of movement. Ergonomics, 16(4), 381–401 (1973)
- [141] H. Tu, J. Allanach, S. Singh, K. Pattipati, and P. Willett. Information integration via hierarchical and hybrid bayesian networks. IEEE Transactions on Systems, Man and Cybernetics, Part A, 36(1), 19–33 (2006)
- [142] E. Wallace, D. Diffley, E. Baines, and J. Aldridge. Ergonomic design considerations for public area cctv safety and security applications. In Proceedings of the 13th Triennial Congress of the International Ergonomics Association (1997)
- [143] X. Wang, X. Ma, and W. Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(3), 539–555 (2009)
- [144] T. Xiang and S. Gong. Video behavior profiling for anomaly detection. IEEE transactions on pattern analysis and machine intelligence, 893–908 (2007)
- [145] T. Xiang and S. Gong. Video behavior profiling for anomaly detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(5), 893–908 (2008)
- [146] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(11), 1531–1536 (2004)
- [147] J. Yin, X. Chai, and Q. Yang. High-level goal recognition in a wireless lan. In Proceedings of the National Conference on Artificial Intelligence, 578–584 (2004)