Learning by Observation Using Qualitative Spatial Relations

Jay Young and Nick Hawes
Intelligent Robotics Lab
The University of Birmingham
United Kingdom

ABSTRACT

We present an approach to the problem of *learning by observation* in spatially-situated tasks, whereby an agent learns to imitate the behaviour of an observed expert with no interaction and limited observations. The form of knowledge representation used for these observations is crucial, and we apply Qualitative Spatial-Relational representations to compress continuous, metric state-spaces into symbolic states to maximise the generalisability of learned models and minimise knowledge engineering. Our system self-configures these representations of the world to discover configurations of features most relevant to the task, and thus build good predictive models. We then show how these models can be employed by situated agents to control their behaviour, closing the loop from observation to practical implementation. We evaluate our approach in the simulated RoboCup Soccer domain, and successfully demonstrate how a system using our approach closely mimics the behaviour of both synthetic (AI controlled) soccer players, and also human-controlled players, through observation.

Categories and Subject Descriptors

[**Machine Learning**]: Learning Settings—*Learning from demonstration*

General Terms

Design, Experimentation, Human Factors

Keywords

Learning by Observation;Qualitative Spatial Representations;RoboCup Soccer

1. INTRODUCTION

Engineering AI systems is a non-trivial task, which is why systems that can *learn* solutions to problems are desireable. In some cases, learning AI systems may be able to display better performance than if they were being programmed by a human developer. Learning-by-doing techniques (e.g. reinforcement learning) can be risky due to the need for trial-and-error which can be dangerous and expensive in certain domains (such as robotics). Alternatively, if an expert agent already exist for a target domain, then a system can learn to replicate a behaviour itself by observing the expert. This is known as *learning by observation* (LbO) [19]. When designing any learning system, we must consider how it will represent its environment. *Quantitative representations*, i.e. representations which use real numbers (to measure, for example, position, area, time etc.) are highly precise, but can be brittle when a system must generalise. Uncertain observations (due to noisy sensors or dynamic environments) exasperate this problem. Many LbO tasks are characterised by access to *limited* training data, so representations that offer good generalisation to scenarios which are previously unseen, but are related to the observed expert data, are essential.

In our work we approach the problem of LbO using *Qualitative Spatial Relations* (QSRs) to represent observations of the continuous, adversarial, spatially-situated domains in which we learn, predict and ultimately replicate the behaviour of entities. We use qualitative representations to discretise over the continuous variables which typically describe the behaviour of mobile entities, such as position, orientation and velocity, replacing them instead with symbolic representations, where these symbols represent ranges of possible continuous values. These techniques may also be *relational*, encoding information about relative properties that hold between entities at a given instant [10]. Such an encoding provides additional explanatory power in co-operative or adversarial domains where decision making is typically based on relationships between entities.

For example, consider a simple scenario from soccer – that of a player attempting to shoot a ball into a goal which is defended by a goalkeeper. When shooting at the goal, the *relative* angles between the player, goalkeeper and goal posts are important. The goalkeeper would like to minimise the angle between himself and a goalpost with respect to the other player, so as to make the shot more difficult for the opponent, whereas the player would prefer to widen the angle to make the shot easier. As there may be a range of angles that are satisfactory to either player, decision making in this task may be interpreted in a qualitative, relational context, with action selection being dependent on the relations that hold in the scene.

This paper makes the following contributions:

- A novel integration of qualitative spatial representations for learning by observation in RoboCup Soccer.
- Experimental comparisons between metric and qualitative representations across different task configurations, using a variety of learning mechanisms.
- An extension of this study to the problem of learning by observation of human behaviour.
- A study of a situated agent using behaviour models learned from observing both synthetic and human experts.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

2. LEARNING BY OBSERVATION

We adopt the LbO formalism of [19], of which we now give a brief overview. Let B_C be the *behaviour* of an expert agent C, and let E be an environment. A behaviour is the controller that the expert uses in order to determine which actions to execute. During operation, this behaviour is expressed as a Behaviour Trace BT representing the actions the expert performs over a period of operation. The behaviour trace of an expert is then:

$$BT_C = [(t_1, x_1), ..., (t_n, x_n)]$$

where t is a time index, and x is an action. The environment E over time is represented by an input trace, where y is a state description:

$$IT = [(t_1, y_1), ..., (t_n, y_n)]$$

The combination of a behaviour trace and an input trace produces a Learning Trace LT:

$$LT = [(t_1, x_1, y_1), ...(t_n, x_n, y_n)]$$

The LbO task is to learn a behaviour B which behaves the same way as the expert agents in the environment E, given the same inputs $LT_1, ..., LT_n$. Given limited training data, it is possible for an LbO agent to be presented with previously unseen states when acting. Given a non-deterministic expert, it is possible for the same controller to yield multiple BTs. In both cases we require that the LbO agent is able to perform appropriately.

In addition to this formalism, the work of [8] identifies a pipeline for the practical implementation of a Learning by Observation system. Our own system adopts this pipeline, and our version of it is shown in Figure 3. The pipeline given by Floyd features the following major steps.

1. Modeling Concerning the initial setup of the observing agent – determining how information about the environment and the entities within it will be provided.

2. Observation At this stage the agent observes an expert engaging in some task, the observations of which are provided in the format described in the modeling stage.

3. Pre-Processing Here, an agent may engage in information extraction to determine what parts of the observed input are relevant, and how to map those inputs to the desired outputs.

4. Deployment This phase involves learning, using the data from the previous phase to build a predictive model, which is then installed on a situated agent and used as a controller. In reference to the formalism, this is equivalent to constructing B, and some function $f(B)$ which provides the practical machinery for implementing the learned behaviour on a situated agent.

In this work, contributions are made to all steps of this framework in showing how such a system can be fully implemented. We are particularly interested in the form of *knowledge representation* available to a LbO system, how that representation can be exploited by machine learning, and the *mechanisms of deployment* to best make use of those models. In practice, these are not at all isolated steps, but are interconnected in various ways, which we will explore later. One main difference between our implementation of this pipeline and that used in the work of [8], is that our approach to *generalisation* is performed as part of step **1**, at the representation phase, rather than as a pre-processing step where learning traces are later examined and generalised.

3. ROBOCUP SOCCER

We apply LbO to the *keepaway* subproblem of RoboCup Soccer. The RoboCup 2D simulator is a widely-used multi-agent testbed which simulates games of soccer played by individual agents (players) in 2D. At any point in time, all players have a 2D pose (i.e. a 2D position and orientation) and the ball has a 2D position. Players have the ability to sense the ball and the other players, but this sensing is both spatially restricted and noisy. Given this limited sensing, on each time step each player must select an *action*: *kick*, *dash*, *turn*, *tackle* or *catch*, each of which may have continuous parameters (e.g. the power of a kick, or the degree of a turn). On each time step, the simulator provides access to the metric positions of all game elements (including static features such as the position of the goals), plus the actions the agents took. We refer to this data as the game's *state*.

3.1 RoboCup Keepaway

The keepaway subproblem takes place in a limited area and features two competing teams: one trying to maintain control of the ball, the other trying to gain control of the ball. Figure 1 shows a typical 3v2 keepaway scenario. The *keeper* team is situated around the borders of the keepaway area, with the opposing team in the centre. The game is split into slices called *episodes*, wherein a single episode continues so long as the keeper team controls the ball. As soon as control is lost to the other team (or the ball leaves the keepaway area), a new episode starts. In this work we make use of benchmark agents that play the keepaway task[1] [27, 29]. These agents follow hand-coded policies, and provide a baseline level of performance, so can be considered experts in the domain.

4. RELATED WORK

[19] identify terms in the literature such as *learning from demonstration, learning by imitation, programming by demonstration, or apprenticeship learning* as being synonymous with learning by observation. There is a wide range of related work under these various monikers, particularly in robotics [14]. The work of [7] uses behaviour traces of humans to train AI agents in tactical decision-making, and work by [8] has looked at learning by observation in the RoboCup Soccer domain, and [30]. [2] use traces of human behaviour to train agents to behave as more realistic opponents in a computer game.

A variety of techniques have been previously applied to behaviour learning in the RoboCup domain, including reinforcement learning approaches [31, 12, 24, 18] which consider learning tasks by doing, rather than observing experts. The work of [8] is the state-of-the-art in terms of learning by observation in the RoboCup domain, however Floyd evaluates *active* learning, whereby in addition to initial training data, an expert is available for the learning agent to request help from when confronted with significantly unfamiliar scenarios. This is discovered to provide improved performance over passive learning, however an LbO agent may not always have such a capability for active learning. In our work, we consider *entirely passive* learning whereby an agent is only given access to a limited amount of examples, and no communication with an expert is possible. In both cases however, one problem LbO systems face is that they must find ways to extract the most information from the data they have been given – be it because requesting help from an expert is expensive, or the available training data is finite or incomplete. Floyd[8] approaches this by building statistical models of environmental features to extract those which provide the most information about the expert agent's behaviour. We approach the problem

[1] http://github.com/tjpalmer/keepaway

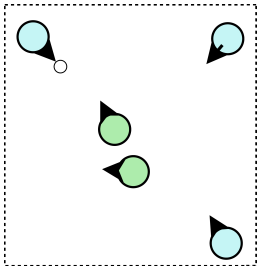

Figure 1: A 3vs2 keepaway Scenario

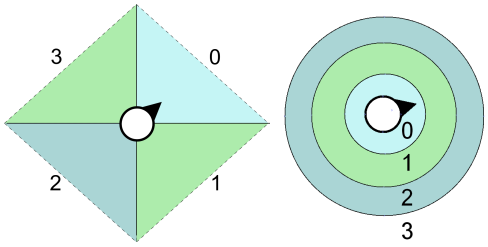

Figure 2: Left: A set of four angular zones egocentric with respect to a player, such that the 0^{th} zone is always aligned with the orientation of the player's body. Right: A homogeneous distance system with four relations.

first with the application of qualitative-relational abstractions, and then use a learning approach that builds a similar statistical model, however now of *abstract spatial relations*. QSRs have been used to learn and recognise the activities of agents in videos [25] and [3]. These approaches typically form predictive models using sequences of relations over time. To model behaviour in complex, continuous domains, we make use of similar representations.

Relational information has often been included in behaviour learning tasks, highlighting its value. In [20], activity recognition is accomplished in a basketball domain using a state-space composed of relational symbols. In many cases relational information takes the form of variables encoding task-specific knowledge, which are often continuously-valued. [5] characterises player behaviour in RoboCup using relational sequence models using pre-defined relational symbols over the action space such as dribble, shoot and pass, abstracting over atomic actions. In [24] agents are provided continuous, relational variables such as the *"angle between the opponent goal and the ball"*, and the twelve additional variables added to the state space of [31] are almost entirely relational in nature, including such things as the *"Angle of the opponent in relation to the agent's body"*. In [15] similar state variables are used, such as *MyDistanceToSecondVisibleOpponent* and *MyAngleToFirstVisibleTeammate*. Much of the existing work employs what are essentially metric-relational representations. While such representations have the advantage of encoding relational information, which can be beneficial, they are still brittle due to being real-valued, and are often compiled on an unprincipled, ad-hoc basis. The representation employed by our work captures relational information by default, without encoding any task-specific data (such as the domain-specific relational symbols mentioned previously), but instead we explore the capabilities and characteristics of discrete *qualitative-relational* representations.

5. QUALITATIVE STATE GENERATION

Our LbO system operates on a sequence of qualitative states, where each state describes the the world at a discrete time step. The qualitative state is a vector of binary relations provided by several qualitative spatial-relational calculi, which we combine into a unified representation. Given the metric state of the world observed by an agent, the QSRs we use discretise it in a variety of ways. The Star Calculus (SC) provides binary relations for representing the *qualitative direction* between entities in space with respect to one-another [22]. SC employs angular zoning based on either the parameter m (yielding zones of size $360/m$, which we use), or through the specification of an arbitrary sequence of angles. We also employ the measure of *qualitative distance* proposed by [6] whereby the distance between two entities A and B is expressed in terms of the presence of B in one of a set of distinct, non-overlapping distance intervals relative to A, yielding doughnut-like spatial regions. An illustration is shown in Figure 2. The Qualitative Trajectory Calculus (QTC) provides qualitative information about the *relative motion* of entities [32], such as moving away/towards or faster/slower. We employ the QTC_B (Basic) variant of this calculi. As with the Star and qualitative distance calculi described above, QSR representations may have several parameters associated with them that determine the level of *granularity* of the discretisation they provide. Changing these parameters produces different binary relations in the resulting QSR state. When learning, different QSRs, or sets of them, may provide more predictive power than others, and this predictive power may also vary with action.Therefore we utilise a *generate-and-test* search [23] to find the most informative QSR representation for each action. This allows us to use QSRs tailored to each action, rather than the single monolithic representations used in previous work on similar QSR-based approaches [33]. We do this using a Monte-Carlo search (as opposed to an exhaustive grid search [4]) through the space of possible QSR representation parametrisations to find those that provide informative features. We evaluate candidates in the classifier-representation pair space with a ten-fold cross-validation, taking the Matthews Correlation Coefficient as a measure of performance [17]. We regularise this using the squared L_2 norm over the representation complexity – i.e. in our case the granularity of the calculi – which applies a penalty for complexity [21], meaning the search prefers *conservative* representations. We build a classifier-representation pair for each action, and combine them using a one-vs-all approach into our predictive model.

6. LEARNING ROBOCUP ACTIONS

Figure 3 shows a high-level overview of our system. Using the benchmark agents of Stone [27] we generated datasets for three common keepaway configurations – 3vs2, 4vs3 and 5vs4 – producing 1000 episodes per configuration. In the raw data extracted from the RoboCup simulator for this task, the behaviour traces contain the action chosen by each agent on each time step along with their continuous parameters (e.g. $kick(64)$ or $dash(99)$), as well as the metric state of the world. We discard the continuous parameters from the actions, leaving only the action class. This makes the classification/prediction problem easier, as we do not require a class for every observed parametrisation of each action. We will show later how the continuous parameters can be recovered for use by a situated agent. The metric observations are transformed into a set of QSR features using the calculi described previously. We use this to produce a model which predicts the action an agent will take given the state of the world it can observe, treating the learning problem as one of classification.

Comparing the performance of classifiers is not as straightfor-

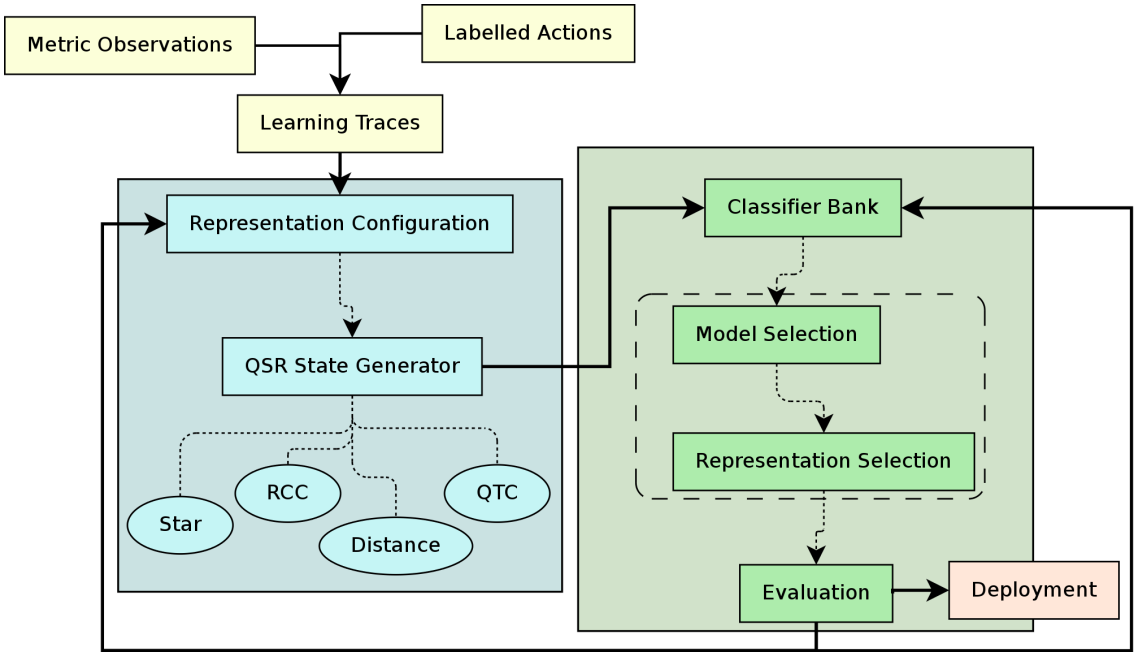

Figure 3: System Overview

ward as just looking at their prediction accuracy. The data features significant class imbalance – the balance of kick, dash and turn actions being 11%, 22% and 66% respectively. Figure 4 shows a comparison between three different learning algorithms – a Bayesian Network, the C4.5 decision-tree classifier and a Support-Vector Machine – using both the underlying metric and automatically-configured QSR representations, evaluated on ten-fold cross-validation. The implementations of the learning mechanisms were taken from the Weka toolkit [11], and standard hyper-parameter optimisation was performed. For the SVM we utilised a radial basis function and C of 1.0. For the C4.5 classifier a confidence factor of 0.27 was used.

The results show that we are able to predict these discrete actions with a high degree of accuracy (97% for the C4.5 classifier) when utilising the QSR-based representation scheme. Utilising the raw metric data provides generally worse results for all learning algorithms.

As described above, [8] also investigated learning by observation in RoboCup. In addition to their active learning approach, they presented a passive approach which is directly comparable with our work. [8] evaluated their system by predicting the behaviour of the *Krislet* team, a simple reactive agent system that chases the ball around the pitch, and kicks it towards the opponent's goal [16]. They also tried to predict the behaviour of the *CMUnited* team, a complex, champion system that employs layered learning, internal world models and multiple behavioural states [28]. This work uses the full-pitch, full-game scenario, as opposed to keepaway. We repeated their experiments with our QSR-based approach, a present directly-comparable results in Figure 5. In all cases we outperform their approach. We put this down to our system being able to represent *relations* between objects, as well as information about their relative motion. This allows for a richer representation of the world, which allows the C4.5 classifier to more closely model the way the expert programs were written. This is most clear in the results for the Krislet agent, which is a simple reactive program,

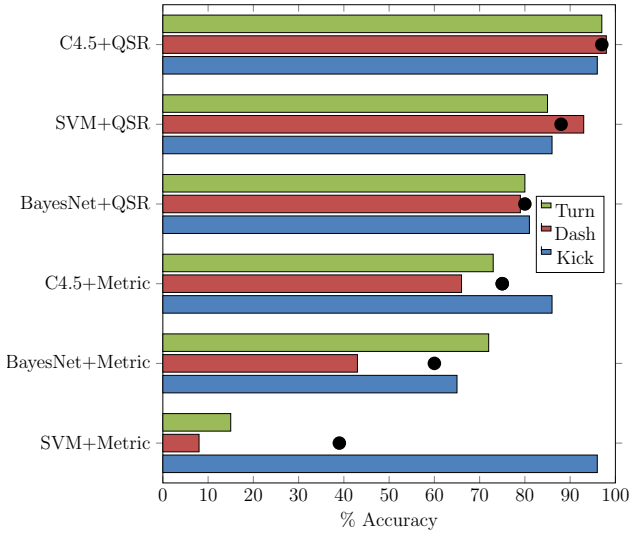

Figure 4: Per-class performance of classifiers in the 3vs2 scenario with synthetic agents. Black dots indicate the average performance of a given classifier.

	Expert	Overall	Kick	Dash	Turn
Our Work	Krislet	**94%**	**96%**	**94%**	**92%**
Floyd [8]	Krislet	61%	27%	71%	83%
Our Work	CMUnited	**64%**	57%	**70%**	**66%**
Floyd [8]	CMUnited	50%	**58%**	55%	50%

Figure 5: Comparison between our work and the work of Floyd, predicting discrete action labels.

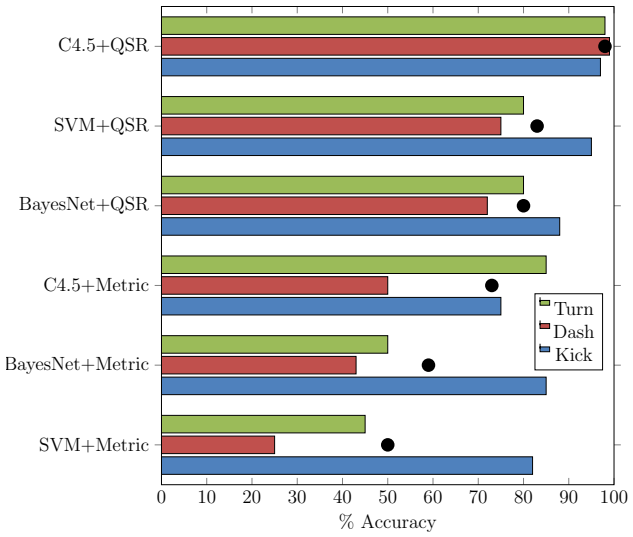

Figure 6: Performance of each learning mechanism on predicting the behaviour of human players in the 3vs2 scenario.

written as a decision tree based on spatial information gathered by the agent. In contrast, our system, as Floyd's, struggles to predict the behaviour of CMUnited, which utilises a learning architecture as well as an internal world model with temporal components. We are as yet unable to capture these expert-internal changes purely through state observation. This highlights a limit on the efficacy of our approach.

6.1 Learning models of human behaviour

In addition to working with benchmark software agents, we also gathered a dataset from human operators controlling a team of RoboCup players in realtime. Using a standard game controller, human operators were able to move players around the game environment and interact with it from the birds-eye view provided by the RoboCup Soccer Simulator visualiser. This is in contrast to the noisy and limited field of view provided to synthetic agents.

We collected 200 episode traces of the 3vs2 game configuration, wherein three human players each controlled a member of the keeper team, and the taker team was controlled by the benchmark agents used in the previous evaluation. This allowed us to create models of human behaviour using the same technique described previously. The predictive results are shown in Figure 6, evaluated using ten-fold cross-validation. Our QSR approach again outperforms the metric representation in general, and C4.5 shows an advantage over other classifiers on the data. This demonstrates that QSRs can encode human behaviour as well as that of artificial agents, and that a decision tree can provide a strong predictive model for that behaviour.

7. PLAYING KEEPAWAY

So far we have mainly discussed classification results and the building of behaviour models, but we now take the important step of showing how these models can be *deployed* within situated agents to perform the behaviours they have learned from observation of both synthetic and human experts. We have developed the first agent to use QSR-based learned behaviour models in the RoboCup soccer domain. Deploying a learning by observation model poses several novel problems to which we have provided solutions. Model deployment is accomplished through a player agent which provides its perception of the environment as input to a pre-trained classifier, and, on each simulator step, uses the resulting classification to perform action selection. To achieve this, we must not only consider the (discrete) output of the classifier, but also the parametrisation of RoboCup actions, which we had previously not considered. To do this, we utilise a rough predictive model of the physics of RoboCup Soccer, provided by the parameters of the simulator describing such things as the ball weight, maximum speed, movement decay, wind resistance and so on. This allows us to simulate the result of actions, producing a distribution over potential future states. For example, we can produce a prediction of the position of the ball n frames after executing a kick action with a specified parameter. We then match the *qualitative* versions of these predicted states with the QSR states observed in the training data – the states subsequent to the one in which the expert executed the predicted action. The closeness of this match provides an objective function to explore the space of possible action parameters, over which we again employ a Monte-Carlo search. We generate candidate parameters, predict their result over a finite time horizon (in our case 10 future frames), to find those that result in qualitative state sequences most similar to those observed in the training data. In our experiments this procedure yields an error rate of 24% when compared with the actual parameters chosen by the expert agent. This may seem high, however in practice miscalculated parameters may not greatly influence behaviour. For instance there may be little practical difference between a turn of 30° and one of 31°, but this is still counted as a miscalculation.

While software agents performed actions with real-valued parameters, the human operators could only choose from a discretised action set with fixed parameters, e.g. a weak, medium or strong kick, all taking their direction from the way the player currently faced. This allowed us to forego parameter recovery with the models based on human data, instead we concatenated action classes with their fixed parameters to create distinct classes in the data.

LbO Agents	Benchmark Agents	Avg. Episode Length
0	3	131
1	2	127
2	1	124
3	0	121

Figure 7: Effects of replacement of pre-programmed benchmark agents in the keeper team with agents using behaviour models generated by observing those benchmark agents (C4.5+QSR).

8. EXPERIMENTS AND RESULTS

Our study of model deployment is restricted to the 3vs2 keepaway scenario, and only the C4.5 classifier, due to space constraints. We tested our system by replacing an increasing number of the benchmark keeper team's players with players controlled by our system. This allows us to evaluate the performance of the LbO agents, not only when teamed with the original hand-coded agents, but also when teamed other LbO agents. As the goal of the keeper team is to maximise episode duration (by keeping the ball), we evaluate our system using the metric of average episode duration over 200 episodes.

Figure 7 presents the results of the systematic replacement of benchmark agents by agents deploying LbO behaviour models. Average episode length decreases consistently as LbO agents are swapped in, showing that these agents cannot completely recreate the observed expert behaviour. This is largely due to misclassifications,

System	Rep.	Avg. Episode Length
Human Controller	-	156 ± 14
Benchmark Agents	-	131 ± 18
LbO (Human)	QSR	134 ± 17
LbO (Benchmark)	QSR	121 ± 12
LbO (Human)	Metric	76 ± 32
LbO (Benchmark)	Metric	65 ± 37
Random	-	60 ± 20

Figure 8: Average episode duration in simulator steps showing variation in performance after learning from both human and benchmark agents. C4.5 is used for both metric and QSR-based approaches.

and errors in their ability to select parameters of actions. Some misclassifications are more severe than others – missing a common action, like a turn or dash on one frame may not have a major impact, however missing a rare, important action such as a kick may be unrecoverable. The same issue arises when selecting parameters for actions. We find that these kinds of errors have the side-effect of landing agents in states that they had not previously observed before. This leads to a feedback loop: the agent will make a mistake, enter into states it is not familiar with, perform poorly, and performance will degrade. This highlights a limitation of our approach so far, and may indicate a good point where reinforcement learning approaches could be integrated into the process. Future work will look at how our approach can be used to jump-start an RL system, potentially using an algorithm such as SARSA [13].

Figure 8 presents a comparison between the QSR and metric representations when all 3 keeper agents are replaced, and also the difference in performance of models learnt by observing synthetic and human experts. Again the QSR representation outperforms the metric, and there is a larger drop in performance on the human data than synthetic in terms of average episode length in simulator steps. The larger deficit on human data can be explained by the fact that the human experts were actually naïve users who had not played RoboCup previously. They therefore had to learn whilst playing and being observed. This is visible in the data, where human episode duration starts low but increases after approximately 20 episodes. The models learnt from humans include these initial episodes, and thus may include errors introduced by the expert – for instance, if the humans pressed a button in error. Human controllers required a handful of trials before they were comfortable and competent with the simulator and control mechanism. This is not the case for the benchmark agents which perform consistently throughout. Discarding the initial 20 episodes from the human data produces average episode lengths of 139 and 96 for QSR and metric representations respectively (an improvement of 5 and 10). This is an interesting result, but discarding this data may not be the best option – we suspect it could be harnessed to provide an agent with a more thorough understanding of the task domain, by observing directly the process of an expert attempting to accomplish some goal.

Our results show that we are able to represent and model a richer variety of behaviours than the existing work by employing qualitative, relational representations, and configuring them automatically. However, a weakness of our approach in the deployment phase is that our agents will also mimic the *mistakes* of an expert. This is because, bestowed with no *a priori* task information, learner agents cannot tell beneficial behaviours from detrimental ones. If an expert agent is observed losing control of the ball, a learner agent, later put in the same scenario, will select actions that lead to the same detrimental result. So far our system is most capable of learning reactive behaviours – such as the benchmark keepaway agents and the Krislet agent. But this also extends to the way human players approached the task, which we have shown the ability to capture and replicate. Our system re-configured its representations of the world in order to find those that gave it better predictive power over the behaviour it had observed. For instance, as mentioned previously, the Krislet agent simply kicks the ball towards a goal when it is nearby. Looking at our model, we see that the representation associated with the classifier for the kick action shows a small number of finely-grained ringed distance intervals. For its configuration of the Star calculus, it favoured relatively wide discretisations of the view cone of around 20°. This means that the representation-classifier pair discovered the underlying control rule of the agent – when the ball is within reach, in front of the player, kick it. The QSR relations used in this are the presence of the ball within the most central of the Star relations (directly in front of the player), and within one of the closest distance rings. Without these symbols being present, the model switches to another classifier. If the ball is not close to the Krislet agent, it will turn and dash towards it. This rule is captured by other relations, such as a further-off distance ring that encodes that the player is far from the ball, or a Star relation that shows it is not facing the ball. While Krislet is a very simple agent, it serves to illustrate how our system harnesses QSRs to build predictive models, and the same principles apply for the models of more complex agents such as human players and the benchmark keepaway players.

In our results, we observed that the C4.5 algorithm proved to be extremely fast in terms of training time, and well-suited to the QSR representation which takes the form of a boolean vector. We will next look in to how this speed can be harnessed to build agents that learn by observation on-line, rather than relying on a separate training phase. Other learning mechanisms often benefit from some pre-processing step – feature selection, clustering etc. – which may be slow, or result in a deformation of the state space. This causes problems later on if, for instance, a feature thought previously to be irrelevant suddenly becomes relevant (perhaps in an edge case) – this then requires tackling the problem of how (or if) this change is detected, and how any pre-processing steps should be reapplied to take the new worth of the feature into account. Using C4.5, this can be solved through a re-training approach due to the speed of the algorithm. Our experience is that this would not be possible using the other learners we evaluated, either due to their relatively high training times.

The work of [26] discusses the problem of ad-hoc teamwork, whereby an agent must co-operate with teammates it has never seen before, cannot communicate with, and who all might follow different policies, potentially in a task it has no experience of itself. Here we expect LbO systems like ours to be key to allowing an agent to meaningfully contribute to a team effort by imitating the behaviour of its teammates. In our own evaluation, we stick to imitating a single set of behaviours in a limited domain, however there is no reason why, in more complex scenarios, a LbO system would not be able to fulfil a multitude of team roles, perhaps even utilising a behaviour model that is the union of many different observed behaviours. For instance, in the 2014 Computer Poker Competition [1], the winning agent used a model trained on observations of a range of previous competition winners. Here is where we expect the capacity for a LbO system to surpass the performance of the experts it observes lies. In some scenarios though, doing only as well as the expert may be the desired outcome – for instance, if the expert is only available for a short time, or the availability of traces of the expert's behaviour are limited (as in our system). The main

performance hit from our LbO approach is that it does not treat the expert as an oracle – other LbO approaches, such as that of [9] gather cases where the agent was uncertain of what to do to be presented to the expert later in order to improve performance. This procedure can be repeated an unlimited number of times. Here, we do not have such a luxury, as we assume the expert is inaccessible, meaning our agent typically fails in scenarios where it has no reference point in the underlying expert trace. Regardless, we demonstrate that our work still beats the state-of-the-art in LbO in our chosen domain.

9. CONCLUSION

We presented an approach to the problem of *learning by observation* in spatially situated tasks, specifically considering how environmental observations should be represented, comparing both quantitative and qualitative representations, aiming to maximise the generalisability of learned models and minimise knowledge engineering. Our system achieved this by self-configuring its own qualitative-relational representation of the world, to discover configurations of spatial-relational features most relevant to the task. We showed that using this approach, it is possible to produce behaviour models of both synthetic, pre-programmed agents, as well as human-controlled agents, with a high degree of accuracy. We then showed how these models can be deployed within situated agents to control their behaviour, closing the loop from observation to practical implementation in a real-world system. Our work was evaluated in the domain of RoboCup Soccer keepaway – a complex, multi-agent, co-operative and adversarial domain – and shown to outperform current state-of-the-art approaches, which rely on extensive knowledge engineering and/or quantitative representations. Our agent is the first LbO agent based on QSRs to be deployed in the RoboCup domain.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623 and the EPSRC grant EP/K014293/1.

REFERENCES

[1] AAAI. Computer Poker Competition. *AAAI Workshop on Computer Poker and Imperfect Information*, 2014.

[2] C. Bauckhage, C. Thurau, and G. Sagerer. Learning human-like opponent behavior for interactive computer games. *Pattern Recognition*, pages 148–155, 2003.

[3] A. Behera, D. Hogg, and A. Cohn. Egocentric activity monitoring and recovery. *Computer Vision ACCV 2012*, pages 7–9, 2013.

[4] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 2012.

[5] G. Bombini, N. D. Mauro, S. Ferilli, and F. Esposito. Classifying Agent Behaviour through Relational Sequential Patterns. *Proceedings of the 4th KES international conference on Agent and multi-agent systems*, pages 273–282, 2010.

[6] E. Clementini. Qualitative representation of positional information. *Artificial Intelligence*, 95:317–356, 1997.

[7] H. K. G. Fernlund, A. J. Gonzalez, M. Georgiopoulos, and R. F. DeMara. Learning tactical human behavior through observation of human performance. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics*, 36(1):128–40, Mar. 2006.

[8] M. Floyd. *A General-Purpose Framework for Learning by Observation*. PhD thesis, Carleton University, 2013.

[9] M. Floyd and B. Esfandiari. Building Learning by Observation Agents Using jLOAF. *Workshop on Case-Based Reasoning for Computer Games: 19th international conference on Case-Based Reasoning*, (Figure 1):37–41, 2011.

[10] L. Frommberger. *Qualitative Spatial Abstraction in Reinforcement Learning*. PhD thesis, University of Bremen, 2010.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[12] T. Jung and D. Polani. Learning RoboCup-Keepaway with Kernels. *JMLR: Workshop and Conference Proceedings*, 1:33–57, 2012.

[13] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *arXiv preprint cs/9605103*, 4:237–285, 1996.

[14] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, Dec. 2011.

[15] S. Konur, A. Ferrein, and G. Lakemeyer. Learning decision trees for action selection in soccer agents. *ECAI-04 Workshop on Agents in dynamic and real-time environments*, 2004.

[16] K. Langner. The Krislet Java Client - http://www.engsoc.org/~kevlam/, 1999.

[17] B. W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et biophysica acta*, 405(2):442–451, 1975.

[18] M. Molineaux, D. W. Aha, and G. Sukthankar. Beating the Defense : Using Plan Recognition to Inform Learning Agents. *Proceedings of Florida Artifical Intelligence Research Society*, 2008.

[19] J. Montana and A. Gonzalez. Towards a unified framework for learning from observation. *IJCAI Workshop on Agents Learning Interactively from Human Teachers*, 2011.

[20] V. I. Morariu and L. S. Davis. Multi-agent event recognition in structured scenarios. *Cvpr 2011*, (1):3289–3296, June 2011.

[21] L. Moseley. *Introduction to Machine Learning*, volume 1. Pitman Publishing, London, 1988.

[22] J. Renz and D. Mitra. Qualitative direction calculi with arbitrary granularity. *Lecture notes in computer science*, 3157:65–77, 2004.

[23] D. Sculley. Results from a semi-supervised feature learning competition. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning,*, 2011.

[24] K. Shimada, Y. Takahashi, and M. Asada. Efficient Behavior Learning by Utilizing Estimated State Value of Self and Teammates. *Robot Soccer World Cup XIII*, pages 1–11, 2010.

[25] M. Sridhar and A. Cohn. Unsupervised learning of event classes from video. *AAAI*, pages 1631–1638, 2010.

[26] P. Stone, G. Kaminka, and S. Kraus. Ad hoc autonomous agent teams: Collaboration without pre-coordination. *AAAI'10*, 2010.

[27] P. Stone, G. Kuhlmann, M. Taylor, and Y. Liu. Keepaway soccer: From machine learning testbed to benchmark. *RoboCup 2005: Robot Soccer ...*, 2006.

[28] P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. *RoboCup-99: Robot soccer world cup III*, (Section 2), 2000.

[29] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement Learning for RoboCup-Soccer Keepaway. *Adaptive Behavior*, pages 1–50, 2005.

[30] K. Sullivan and S. Luke. Real-Time Training of Team Soccer Behaviors. *RoboCup 2012: Robot Soccer World Cup XVI*, 2013.

[31] D. C. D. L. Vieira, P. J. L. Adeodato, and P. M. Gon. Improving Reinforcement Learning Algorithms by the Use of Data Mining Techniques for Feature and Action Selection. *IEEE International Conference on Systems Man and Cybernetics*, pages 1863–1870, 2010.

[32] N. Weghe, A. Cohn, G. Tre, and P. Maeyer. A qualitative trajectory calculus as a basis for representing moving objects in geographical information systems. *Control and Cybernetics*, 35(1), 2006.

[33] J. Young and N. Hawes. Predicting Situated Behaviour Using Sequences Of Abstract Spatial Relations. *In Proceedings of the AAAI 2013 Fall Symposium How Should Intelligence be Abstracted in AI Research: MDPs, Symbolic Representations, Artificial Neural Networks, or _____?*, 2013.