Learning a policy for collision avoidance by mining interactive multi-robot games

Jeffery Raphael[1], Eric Schneider[1], Simon Parsons[1] and Elizabeth Sklar[1]

Dept of Computer Science, University of Liverpool, UK
jeffery.raphael@liverpool.ac.uk, eric.schneider@liverpool.ac.uk,
s.d.parsons@liverpool.ac.uk, e.i.sklar@liverpool.ac.uk

Abstract. Humans are very good at abstract spatial reasoning, both in the physical world and in virtual settings. For example, game players immersed in virtual environments simulating mobile agents can quickly identify potential collisions and effect evasive action if given control over agents' movements. In a multi-robot system, avoiding collisions is a necessary behaviour. Traditional methods typically employ a fixed policy, such as robots negotiating for right-of-way when approaching a threshold separation distance. More sophisticated methods that account for robots' velocities and planned paths of motion can be more effective, but are computationally expensive, do not scale well, and are impractical for deployment on-board limited robot platforms. The work presented here takes advantage of humans' spatial reasoning skills, by collecting data from a simple 2D game where multiple robots move around in an enclosed arena and a human player prevents the robots from colliding with each other by issuing commands via a point-and-click interface. Behaviours for avoiding collisions are learned by mining data logged during games. These mined behaviours are then deployed on robots in a non-interactive environment and are compared with a fixed-range policy. Results show that mined behaviours provide a more effective and flexible alternative to the fixed policy.

1 Introduction

The notion of *behaviour mining* combines *data mining* and *behaviour modelling*. Here, we describe an approach to behaviour mining that involves building agent-based models of human behaviour by mining data collected from interactive applications, such as decision-making tools or games. The aim is for the acquired model to be used as a controller for an agent subsequently deployed in the application, either to perform better on its own, with other agents and/or with humans. This could be an agent that helps a novice human make decisions in a complex environment by suggesting actions that might be taken by more experienced human users; or it could be an agent that challenges a human in a competitive game environment by emulating moves of more creative or advanced human players.

Our approach to emulating human behaviour differs from traditional expert systems for several reasons. First, the human whose behaviour is being modelled

is not asked to explain their actions. Instead, data reflecting how the human behaves when interacting in the environment is mined for recognisable patterns. Second, the human being modelled is not necessarily an expert. Instead, a wide range of users' behaviours might be modelled, from novice to expert. There are many reasons for wanting to be able to model different types of users' behaviours. For example, in a learning environment, it would be helpful to be able to model behavioural characteristics commonly exhibited at different stages of the learning process. In a complex environment, it is sometimes hard to rank expertise, and instead, it may be desirable to characterise a range of strategies that exhibit different strengths.

There are many advantages to taking a data mining approach to behaviour modelling. Especially in real-time and complex domains, humans typically cannot explain how they perform a task or why they choose their actions. This is particularly true of people who are not experts or teachers. So, if the goal, as above, is to model not (only) experts but (also) a range of different levels and types of users, then a traditional knowledge-engineering approach is insufficient. Another advantage of a data mining approach is that behaviour can be captured "after the fact". As long as activity is being recorded, behavioural information lies in these logs. For interactive computer-based applications, these data are often referred to as the *clickstream*.

In the work presented here, we take advantage of humans' exceptional abilities with respect to spatial reasoning and decision making in dynamic, uncertain environments. For example, when crossing a crowded city street or rushing through a busy airport, humans can quickly judge whether they will need to modify their pace and/or heading in order to avoid crashing into other people. Further, these abilities, for many people, extend to video games, where human players control fast-moving vehicles and manage to avoid crashing them into other moving objects, such as racing games where players have to weave around cars driving at high speeds on a confined track, or spaceship games where players have to dodge asteroids or alien missiles.

We apply this observation to the multi-robot domain and explore two related questions: (1) Can a human player who has some control over a robot moving around in a video-game-like environment issue simple commands to prevent a robot from colliding with others? (2) If so, can we then learn a model of this human-directed collision avoidance behaviour by applying data mining and agent-based modelling techniques to the human's clickstream data, and thus learn an effective policy for collision avoidance? In this paper, we address both questions, but primarily focus on the second (because the first is true for most people). We begin in Section 2 by reviewing related work with respect to behaviour mining. Then, Section 3 describes our approach for mining users' clickstream data, learning a policy and applying that policy to a robot. Section 4 details a series of experiments that we have run to assess the potential of our approach and reports the results of those experiments, which are discussed in Section 5. Finally, we close with a summary and outline of future work.

2 Related Work

Mining the clickstream is not a new idea. Teitelman developed an automatic error correction facility in the 1970's [1], and Cypher created an agent that recognised repetitive tasks in the 1990's [2]. Subsequently, Maes used machine learning techniques to train agents at a variety of tasks (e.g., mail processing, news filtering, recommending entertainment) and acquire a representation of users' preferences [3]. Network security researchers have demonstrated the use of data mining techniques for modelling behaviours for intrusion detection [4, 5]. Yang et al. [6] applied data mining to modelling customers' switching behaviours in the telecommunications industry. The web is a popular environment for behaviour mining, and much recent work is centred on modelling patterns of searching and other web-based activity (e.g., [7–9]). In earlier related work, we learned behaviours from data collected while humans played video and educational games [10–12].

The idea of learning from humans within the context of robotics is also not new. A range of explicit teaching techniques, such as learning or programming from demonstration [13, 14] have been explored. These are generally considered supervised learning methods, acquiring a policy from a sequence of state/action pairs [13].

Chernova & Velosa [15] developed an interactive policy-learning algorithm where an agent is able to actively request demonstrations when needed. They utilised *Gaussian mixture models (GMM)* as a mapping function to derive a policy, where each GMM, with multiple Gaussian components, represents a single action. They were able to reduce the number of demonstrations required by the agent to learn a task, thus improving training time and minimising teacher effort. To test their algorithm, they trained a Sony Aibo robot to navigate a circular path. The Aibo used its infrared sensor, located in its head, to create a three-dimensional vector representation of its surroundings. They compared their algorithm to an agent that learned to navigate the path using a variant of reinforcement learning called Prioritized Sweeping [16] and found that the reinforcement learning method took much longer than the demonstration-based method.

Knox & Stone [17] developed the "Training an Agent Manually via Evaluation Reinforcement" (TAMER) framework, which provides a scheme for interactively training an agent using *shaping*. The machine learning instantiation of shaping is based on an animal training method where the trainer rewards successive precursory actions until some target behaviour is achieved. In shaping, the trainer observes the agent and provides simple feedback about each action the agent has taken. Knox & Stone evaluated their framework using two domains: the game of Tetris and the Mountain Car simulation. They employed 28 human trainers, 9 for Tetris and 19 for Mountain Car. They compared their Tetris TAMER agent to other Tetris agents developed in [18–21] and their Mountain Car agent to two different $Sarsa(\lambda)$ agents. They found that the Tetris TAMER agent learned an optimal policy much faster than the other agents. The agent, on average, outperformed both $Sarsa(\lambda)$ agents for the first few episodes.

Other approaches to robots learning from people include Katagami & Yamada [22], where human teachers provide examples that seed evolutionary learning; Lockerd & Breazeal [23], where a robot tries to identify a human's goal and make its own plan to achieve it; and Nicolescu & Matarić [24], where a robot observes while the human carries out actions in its domain and learns the outcomes of its own actions from these observations. Little of this work is concerned with multiple robots, however. There is a long history of multi-robot learning, for example [25–28], but these involve learning by trial and error, not learning from a human teacher.

3 Approach

Our behaviour mining approach is a multi-step process. This section describes our general approach. Details of our experimental environment and results are deferred to the next section.

3.1 Collecting clickstream data

First, we collect data from human subjects. For the experiments described here, a graphical user interface is provided to a human player, displaying a bird's-eye view of an environment in which a team of robots is deployed for performing exploration tasks. A screenshot of the interface is shown in Figure 1. The environment consists of six interconnected "rooms". Suspended above the arena is a grid of web cameras that track the robots in the arena, providing robots' positioning information to the team. At the start of a game, the human player and robot team are given a mission: the robot team must explore a set of *interest points* in the arena, and the human must prevent the robots from crashing into each other. The team uses a simple auction-based mechanism to distribute the interest points amongst the robots (we have documented this mechanism in other related work [29, 30]). Once the interest points are distributed, the robots start moving around the environment, exploring their assigned locations. During this *execution phase* of the game, it is the human's responsibility to ensure that the robots do not collide with each other. The human player can click on a robot to *pause* its movement, if she is worried that the robot might crash into a teammate. The player can then click on the same robot again, to *resume* its motion once the danger of colliding with a teammate has passed. The human can also pause/resume all the robots with the press of a button. Note that the robots have an internal map of their environment, so they know where the walls are and how to avoid them; the collision avoidance behaviour investigated here is focused on robots avoiding each other.

During a game, the system logs data continuously. Robots' positions are recorded multiple times per second. The waypoints that define robots' paths[1] to

[1] A robot devises a path to its next interest point using the classic A* path-planning algorithm [31].

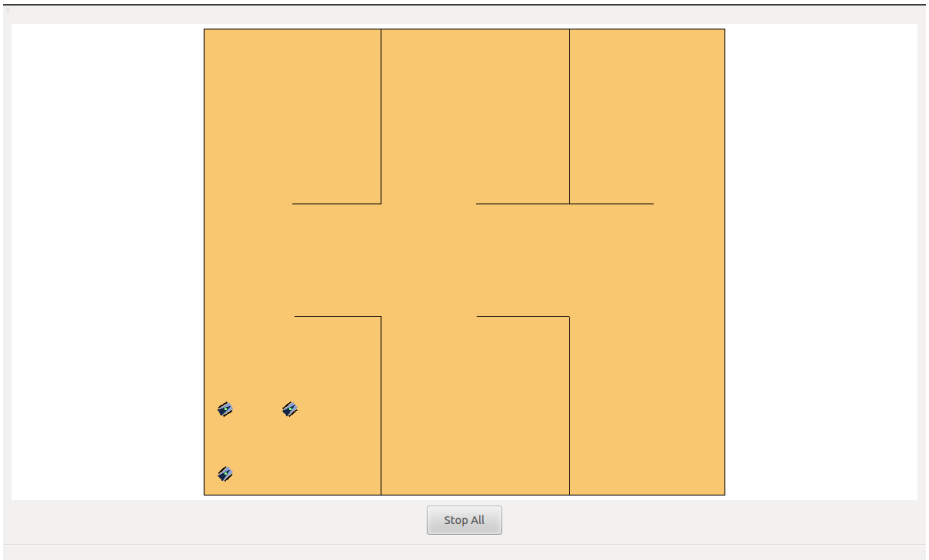


Fig. 1. Screenshot of user interface

their next interest points are recorded every time a new path is planned or an existing path is re-planned. The log file for any run can be used to reconstruct the state of the environment, at each timestep recorded. As described below, we mine these logs to learn the human's policy for pausing and resuming robot motion.

3.2 Mining the clickstream

Our overarching aim is to learn a small-footprint policy that can be placed on-board a low-cost robot platform that has limited memory and computing capacity. So it is important that the learned policy can be represented and executed with a conservative amount of memory and computation time. This means that many machine learning techniques would be unsuitable for this task. Neural networks, which are very good at generalising and identifying non-linear relationships, may require storing a large number of weights and/or performing a large number of computations. Similar barriers impede the use of GMMs, which have many applications in learning from human interaction [13]. Unfortunately, GMMs are another computationally expensive method of learning. GMMs rely on an iterative algorithm, called *Expectation-Maximization (EM)*, to maximise the logarithm of a likelihood function [32]. The EM algorithm itself is susceptible to singularities (unique conditions where the log-likelihood function approaches infinity) and may even converge to a non-global maximum.

We used WEKA[2] [33], an open source data mining tool, to develop our model. WEKA is capable of quickly processing a training set and developing a variety of classifiers. We chose to implement the learned policy in the form of a decision tree because decision trees are easy to implement and have the added bonus of being completely transparent. Our learned policy is implemented using WEKA's J48 decision tree (Quinlan's C4.5 decision tree algorithm [33]).

The training set used to develop our model is a collection of pair-wise configurations of robots and their corresponding labels extracted from the log data. Each example in the training set is represented by a seven-dimensional state vector. This state vector is defined as:

$$\langle range, \theta, V_x, V_y, H_x, H_y, \alpha \rangle$$

where *range* is the Euclidean distance (in cm) and θ is the angle (in radians) between two robots;

V_x, V_y and H_x, H_y are the velocity and heading (to the next waypoint) components of one of the robots in the pair.

The last dimension, α, is a flag indicating whether the human clicked on the robot in this state or not. For each time step, two state vectors are produced for each unique robot pair. As described earlier, the effect of a click is to stop a robot if it is moving, or restart a robot if it is stopped.

As is often the case, the data must be pre-processed prior to training in order to prevent skewed learning. Before the log data is processed by WEKA, we balance the data set, because there are so few *pause* commands in relation to the other data that is collected. We create a training set by filtering the majority of exemplars where the human chose to do nothing (i.e., did not click on a robot). For every recorded click, two non-click exemplars are chosen from the log. These other exemplars are selected at regular intervals in the log file, to obtain a training set that is not only balanced but also evenly distributed throughout the duration of the training game. It should also be noted that because the aim is to learn collision avoidance, all the exemplars represent states where collisions did not occur.

3.3 Constructing the behaviour model

Embedding the decision tree into our multi-robot system entails simply placing code that follows the rules in the tree within our existing robot controller. A decision tree, such as the ones shown in Figure 3, can be transcribed into a sequence of nested `if-then` statements that drive the robot's collision avoidance behaviour.

4 Experimental Results

This section describes our experiments and results. First, we collected data from human subjects to seed the behaviour mining process. We asked each human

[2] `http://www.cs.waikato.ac.nz/ml/weka`, Version 3.6.7

subject to play 5 games. In each game, three robots were deployed to explore a total of 8 interest points. The configuration of the 8 interest points was different in each game, but the robots always started in the same positions (see Figure 2).

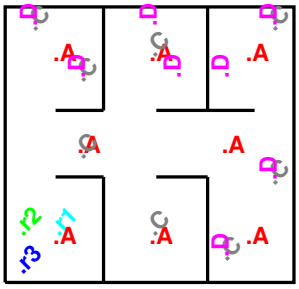


Fig. 2. Game configurations: r*i*'s indicate robots' starting positions; labels indicate positions of interest for three different game configurations, A, C and D (some overlap).

Then, we ran the J48 decision tree algorithm to mine the data logged during humans' games (see Section 3.2). Table 1 shows the results of the training on data from two human players, as well as stratified 10-fold cross validation of the learned policy and the corresponding confusion matrix.

	human player #1 (number of instances = 186)		human player #2 (number of instances = 364)	
	training	cross-validation	training	cross-validation
correct classifications	82.80%	72.58%	79.95%	71.43%
Kappa statistic	0.66	0.46	0.60	0.43
root mean squared error	0.34	0.44	0.38	0.45

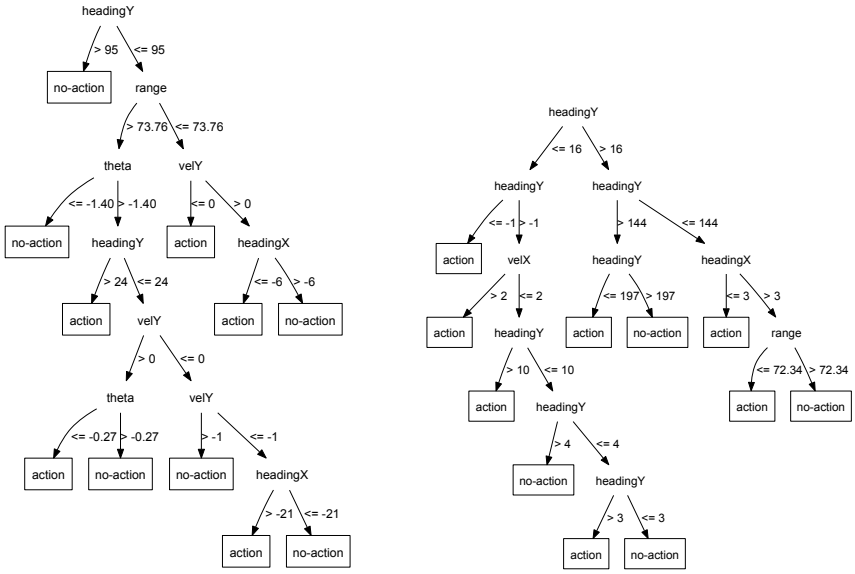
confusion matrix (cross-validation):

classified as →	*no action*	*action*	*no action*	*action*
no action	76	17	111	71
action	34	59	33	149

Table 1. Results of mining humans' games

The next step was to test the ability of the mined policy to guide a robot to avoid collisions. As outlined in Section 3.3, we coded the mined policy in our robot controller software. Figure 3 contains the decision trees that were learned from mining the logs of games with two human players. The training examples gathered seem to indicate that both players considered a robot's heading in the

positive y direction (i.e., up), represented by the variable H_y in the state vector, as the top-level criterion for determining whether to click on a robot or not. It is interesting to observe that player #1 initially branches on $headingY = 95$, while player #2 branches on a much lower value, $headingY = 16$. Subsequently, player #2's behaviour is almost exclusively defined by further refinements of $headingY$; whereas player #1's behaviour considers other aspects of the robot state vector, including range, direction and velocity.



(a) human player #1 (b) human player #2

Fig. 3. Decision trees mined from games with human players.

We then deployed one robot using the mined policy (from human player #1) in a non-interactive version of the multi-robot game described in Section 3.1 in order to assess the learned behaviour. Figure 4 illustrates traces of some representative games. The robots start in the lower left corner of the environment, indicated by three square boxes. The interest points for each configuration are indicated with red X's. The top row shows the games played by the human. The bottom row shows non-interactive simulations where one robot uses the learned policy for collision avoidance (indicated by the cyan-coloured lines).

These runs demonstrated that the mined policy can avoid collisions, but they do not indicate how good the policy is. For example, how does the collision avoidance in the non-interactive games compare to the human-controlled colli-

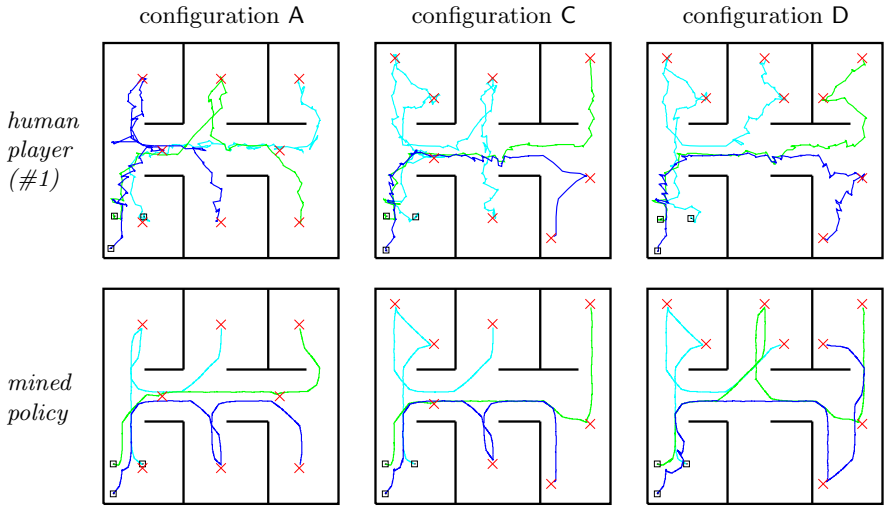


Fig. 4. Performance of policy mined from human player #1. The trajectories for the human games (on the top) are more irregular because they show paths of robots moving in a physical environment, where noise in the motion and position estimates introduce fluctuation. In contrast, the mined policy (on the bottom) was tested in a noiseless simulation environment.

sion avoidance? Table 2 illustrates two metrics computed with respect to human player #1. The first metric, number of pauses, counts the number of times that the human player or the robot following the mined policy paused to avoid a collision. The second metric, total pause time, sums the amount of time that the paused robot in each condition remained motionless until resuming movement, either because the human clicked on it a second time or because the mined policy sensed the environment was safe for the robot to move again. On average, the human initiated pausing less frequently than the mined policy. However, the amount of time that the robot remained motionless in the human-controlled game was significantly longer than in the non-interactive simulation. The biggest indicator of this is the average pause time, which is only 2.27 seconds for the mined policy as compared to 12.70 seconds for the human-controlled robot.

These results appear promising, however, one pressing question remains: how effective is this mined behaviour in comparison with the fixed policy that was in place in the multi-robot game prior to the experiments described above? The next section addresses this question.

5 Analysis

Our autonomous multi-robot system has been using a manually engineered *fixed range* policy for collision avoidance, which works as follows. If, whilst moving, two

	human player	*mined policy*
number of pauses	5.00 (1.22)	16.87 (6.84)
total pause time (sec)	63.52 (27.68)	38.36 (25.41)
average pause time	12.70	**2.27**

Table 2. Metrics comparing performance of mined policy with that of human player. Human player statistics are averaged over 5 games recorded by one representative player. Mined policy performance is averaged over 15 games. Standard deviation values in parentheses.

robots detect that they are within a fixed range of each other, then both robots stop. They exchange messages indicating the distance that each has remaining to travel from their current location to their current interest point. The robot with the least path alteration cost[3] is given right-of-way and resumes its motion. The other robot waits until the first robot has moved beyond the fixed threshold distance, and then the second robot resumes its motion. This is a very simple policy, and it works satisfactorily most of the time. In prior work, we have tested this policy successfully both in simulation and on physical robots [34]. However, this policy is not flexible, nor is it particularly efficient. For example, Figure 5 illustrates two cases where there is potential for collision. In case I, the robot on the left is heading toward the waypoint on the right, and the robot on the right is heading toward the waypoint on the left. Clearly, the robots' projected paths intersect, but depending on the relative timing of the robots' entries into the potential collision zone (marked by a circle in the centre of the diagram), there may or may not be a collision. In case II, even though both projected trajectories intersect with the potential collision zone, the robots are less likely to need to initiate a collision avoidance policy because their routes do not cross each other.

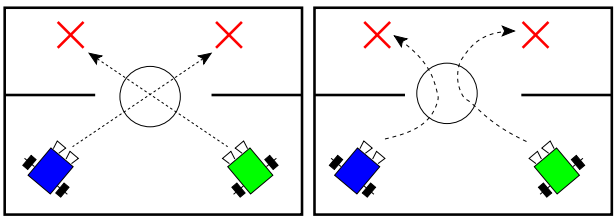


Fig. 5. Example paths that intersect (I) and do not (II).

[3] The path alteration cost is calculated as the distance each robot must travel to go around the other robot, while still heading to its next waypoint.

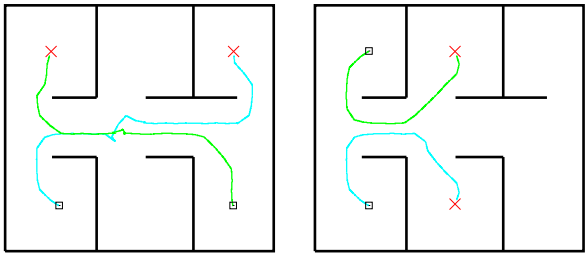
There is related work that explores modulating robots' velocities to avoid collisions, e.g., [35, 36]. In contrast, our goal is to provide a range of policies for members of a heterogeneous robot team, where all robots do not use the same policy for collision avoidance and they do not make assumptions about how the other robots' policies operate. This is in contrast to [35], for example, because there the assumption is that all the robots follow the same policy of slowing down to avoid meeting. Other work, such as [37], considers heterogenous strategies for agent behaviour models in simulated crowds, which is a large application area for multiagent simulation. Future work will include comparison with other collision avoidance techniques that have been successfully demonstrated in simulation environments.

We compared the performance of the policy mined from human player #1 with our fixed range policy in two test cases similar to those shown in Figure 5. We ran each case with two conditions: (a) both robots used the policy mined from human player #1; and (b) one robot used the fixed range policy and the other robot uses the mined policy. Figure 6 illustrates sample trajectories from some successful runs for these conditions. Table 3 contains metrics for the test cases, as in Table 2. This time, the metrics are less clear—especially since the standard deviations are extremely high. Ignoring standard deviation, the condition with the shortest average pause time is when the paths intersect and both robots employ the mined policy. However, when the paths come close to each other but do not intersect, the shortest average pause time occurs when both robots employ the fixed policy. These results warrant further investigation.

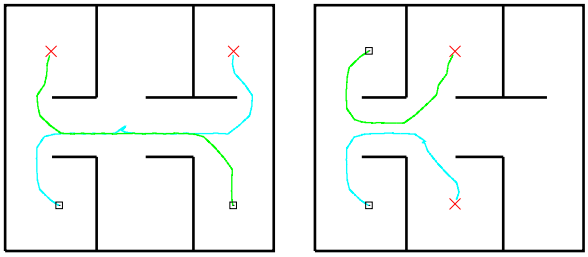
	intersecting paths	*close paths*
number of pauses		
(a) both mined policy	13.00 (10.40)	13.33 (16.65)
(b) one of each policy	7.00 (0.00)	15.33 (10.69)
total pause time (sec)		
(a) both mined	20.37 (7.87)	27.37 (33.82)
(b) one of each policy	26.67 (0.00)	43.03 (25.55)
average pause time		
(a) both mined	**1.57**	2.05
(b) one of each policy	3.81	2.81

Table 3. Comparison of metrics on test cases

Finally, we look at the range values for both mined policies. The fixed-range policy uses a value of 35cm to determine whether to invoke a collision avoidance action or not. Analysis of the training data reveals a lower range value for both human players (see Table 4). Figure 7 plots the range values from the training data, in the order in which they are presented to the trainees. The red circles indicate where actions were issued. The horizontal blue lines indicate the minimum



(a) both mined policy



(b) one of each policy (cyan-colored trajectory is that of mined policy)

Fig. 6. Sample trajectories of test cases

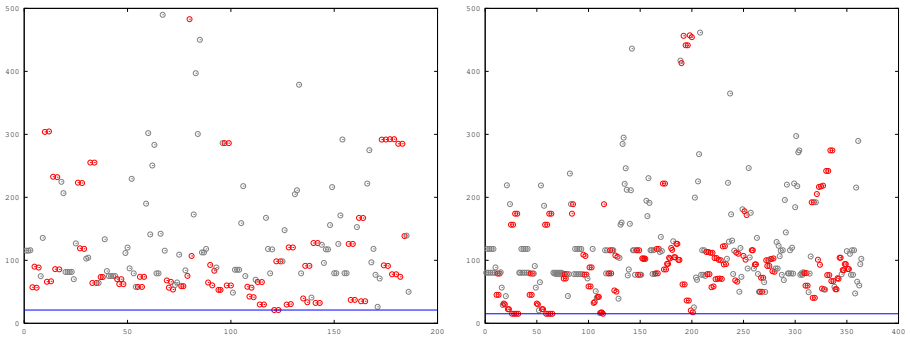
action range values that are listed in Table 4. It is easy to see the differences in policies, even from just looking at this range data.

policy	*minimum action range (cm)*
human player #1	21.10
human player #2	15.13

Table 4. Minimum action range values. Compare to the fixed-range policy which activates collision avoidance at a threshold of 35cm.

6 Summary

We have presented a method for behaviour mining, applied to collision avoidance in multi-robot systems. This methodology has several advantages. First, it is a general technique for constructing behaviour models from data and can be applied in a variety of domains. Second, it does not require a human teacher to explain their actions, engineer training examples or spend a long time demonstrating behaviours for a learner. Third, the resulting policy does not require a



(a) human player #1 (b) human player #2

Fig. 7. Range and action values in training data. Red circles indicate where actions where issued. Horizontal blue lines indicate minimum action range values (see Table 4). There are some outliers, which indicates that range alone is not the only factor being considered by the human trainers—as borne out by the decisions coded in Figure 3.

lot of memory to store or computational power to execute, which makes it appropriate for deployment on low-cost, small compute-footprint robot platforms. Finally, in the context applied here, the results demonstrate behaviours that are more flexible and efficient than the alternative fixed policy evaluated.

Follow-on work will extend this method to a wider user study, gathering data from many human subjects and mining policies from a broader population. We are also experimenting with a 3D version of the interactive game, using Blender[4] to provide a more immersive environment for the human players.

As well, additional experiments will be conducted in which the mined policy is evaluated on physical robots. Finally, a more comprehensive study of collision avoidance in multi-robot systems is in process, to compare techniques other than the fixed-range policy assessed here, using the metrics discussed herein.

Acknowledgments

This work was partially funded by the National Science Foundation (NSF) under grant#IIS-1116843, by a University of Liverpool Research Fellowship and by a Fulbright-King's College London Scholar Award.

[4] `http://www.blender.org`

References

1. Teitelman, W.: A display oriented programmer's assistant. International Journal of Man-Machine Studies **11** (1979) 157–187
2. Cypher, A.: Eager: Programming repetitive tasks by example. In: Proceedings of the ACM Conference on Human Factors in Computing Systems. (1991)
3. Maes, P.: Agents that reduce work and information overload. Communications of the ACM **37**(7) (1994) 31–40
4. Lee, W., Stolfo, S.J., Mok, K.W.: A data mining framework for building intrusion detection models. In: Proceedings of the 1999 IEEE Symposium on Security and Privacy. (1999) 120–132
5. Mukkamala, S., Xu, D., Sung, A.H.: Intrusion detection based on behavior mining and machine learning techniques. Advances in Applied Artificial Intelligence, Lecture Notes in Computer Science **4031** (2006) 619–628
6. Yang, Y.M., Wang, H., Li, L., Li, T.Y., Li, W.M., Yang, Q., Lv, W., Huang, P.: Multi-dimensional model-based clustering for user-behavior mining in telecommunications industry. In: Proceedings of the Third International Conference on Machine Learning and Cybernetics. Volume 3., IEEE (2004) 1650–1655
7. Xue, L., Chen, M., Xiong, Y., Zhu, Y.: User navigation behavior mining using multiple data domain description. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). Volume 3. (2010) 132–135
8. Shan, X., Sun, H.: The research of web users' behavior mining based on association rules. In: 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), IEEE (2011)
9. Schur, M., Roth, A., Zeller, A.: Mining behavior models from enterprise web applications. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE), ACM (2013) 422–432
10. Sklar, E.I.: CEL: A Framework for Enabling an Internet Learning Community. PhD thesis, Department of Computer Science, Brandeis University (2000)
11. Raphael, J., Sklar, E.I.: Exploring the challenge of learning well from a sparse training set. In: Proceedings of the Workshop on Human-Agent Interaction Design and Models (HAIDM) at Autonomous Agents and MultiAgent Systems (AAMAS), St Paul, MN, USA (May 2013)
12. Sklar, E.I., Blair, A.D., Pollack, J.B.: Chapter 8: Training Intelligent Agents Using Human Data Collected on the Internet. In: Agent Engineering. World Scientific, Singapore (2001) 201–226
13. Argall, B., Chernova, S., Browning, B., Veloso, M.: A survey of robot learning from demonstration. Robotics and Autonomous Systems **57**(5) (2009) 469–483
14. Hersch, M., Guenter, F., Calinon, S., Billard, A.: Dynamical system modulation for robot learning via kinesthetic demonstrations. IEEE Transactions on Robotics (2008)
15. Chernova, S., Veloso, M.: Confidence-based policy learning from demonstration using gaussian mixture models. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS), ACM (2007)
16. Moore, A.W., Atkeson, C.G.: Prioritized sweeping: Reinforcement learning with less data and less time. Machine Learning **13** (October 1993) 103–130
17. Knox, W.B., Stone, P.: Interactively shaping agents via human reinforcement: The tamer framework. In: The Fifth International Conference on Knowledge Capture. (September 2009)

18. Ramon, J., Driessens, K.: On the numeric stability of gaussian processes regression for relational reinforcement learning. In: In ICML-2004 Workshop on Relational Reinforcement Learning. (2004) 10–14
19. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific (1996)
20. Böhm, N., Kókai, G., Mandl, S.: Evolving a heuristic function for the game of tetris. In Abecker, A., Bickel, S., Brefeld, U., Drost, I., Henze, N., Herden, O., Minor, M., Scheffer, T., Stojanovic, L., Weibelzahl, S., eds.: LWA, Humbold-Universität Berlin (2004) 118–122
21. Szita, I., Lörincz, A.: Learning tetris using the noisy cross-entropy method. Neural Computation **18** (December 2006) 2936–2941
22. Katagami, D., Yamada, S.: Interactive classifier system for real robot learning. In: IEEE International Workshop on Robot and Human Interaction. (2000) 258–263
23. Lockerd, A., Breazeal, C.: Tutelage and socially guided robot learning. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2004)
24. Nicolescu, M.N., Matarić, M.J.: Learning and interacting in human-robot domains. IEEE Transactions on Systems, Man, and Cybernetics **31**(5) (2001) 419–430
25. Matarić, M.: Reinforcement learning in the multi-robot domain. Autonomous Robots **4** (1997) 73–83
26. Parker, L.: Multi-robot learning in a cooperative observation task. In: Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems. (2000)
27. Bowling, M., Veloso, M.: Simultaneous adversarial multi-robot learning. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico (2003)
28. Pugh, J., Martinoli, A.: Multi-robot learning with particle swarm optimization. In: Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems, Hakodate, Hokkaido, Japan. (2006)
29. Schneider, E., Balas, O., Özgelen, A.T., Sklar, E.I., Parsons, S.: An emperical evaluation of auction-based task allocation in multi-robot teams. In: Proceedings of the 13th international conference on Autonomous agents and multiagent systems. (2014) Extended Abstract.
30. Özgelen, A.T., Schneider, E., Sklar, E.I., Costantino, M., Epstein, S.L., Parsons, S.: A first step toward testing multiagent coordination mechanisms on multi-robot teams. In: Proceedings of the Workshop on Autonomous Robots and Multirobot Systems (ARMS) at Autonomous Agents and MultiAgent Systems (AAMAS). (2013)
31. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimal cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2) (1968)
32. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
33. Witten, I.H., Frank, E., Hall, M.A.: Data Mining, Practical Machine Learning Tools and Techniques. Third edn. Elsevier Inc. (2011)
34. Sklar, E.I., Özgelen, A.T., Schneider, E., Costantino, M., Munoz, J.P., Epstein, S.L., Parsons, S.: On Transfer from Multiagent to Multi-Robot Systems. In: Proceedings of the Workshop on Autonomous Robots and Multirobot Systems (ARMS) at Autonomous Agents and MultiAgent Systems (AAMAS). (2012)

35. Guy, S.J., Lin, M.C., Manocha, D.: Modeling collision avoidance behavior for virtual humans. In: Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2010) 575–582
36. Kimmel, A., Dobson, A., Bekris, K.: Maintaining team coherence under the velocity obstacle framework. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2012)
37. Durupinar, F., Allbeck, J., Pelechano, N., N.Badler: Creating Crowd Variation with OCEAN Personality Model. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2008)