Learning to Avoid Collisions

Elizabeth Sklar[1,4], **Simon Parsons**[1,4], **Susan L. Epstein**[2,4], **A. Tuna Özgelen**[4], **J. Pablo Muñoz**[4], **Farah Abbasi**[3], **Eric Schneider**[2] and **Michael Costantino**[3]
[1]Brooklyn College, [2]Hunter College, [3]College of Staten Island, [4]The Graduate Center
The City University of New York
New York, NY USA
sklar@sci.brooklyn.cuny.edu, parsons@sci.brooklyn.cuny.edu, susan.epstein@hunter.cuny.edu, tunaozgelen@gmail.com, jpablomch@gmail.com, farah.abbasi@cix.csi.cuny.edu, nitsuga@pobox.com,michael.costantino@cix.csi.cuny.edu

Abstract

Members of a multi-robot team, operating within close quarters, need to avoid crashing into each other. Simple collision avoidance methods can be used to prevent such collisions, typically by computing the distance to other robots and stopping, perhaps moving away, when this distance falls below a certain threshold. While this approach may avoid disaster, it may also reduce the team's efficiency if robots halt for a long time to let others pass by or if they travel further to move around one another. This paper reports on experiments where a human operator, through a graphical user interface, watches robots perform an exploration task. The operator can manually suspend robots' movements before they crash into each other, and then resume their movements when their paths are clear. Experiment logs record the robots' states when they are paused and resumed. A behavior pattern for collision avoidance is learned, by classifying the states of the robots' environment when the human operator issues "wait" and "resume" commands. Preliminary results indicate that it is possible to learn a classifier which models these behavior patterns, and that different human operators consider different factors when making decisions about stopping and starting robots.

Introduction

We are interested in deploying human-robot teams to solve problems that are dangerous for human teams to tackle, but are beyond the capabilities of robots alone. Examples of such tasks include *urban search and rescue* (Jacoff, Messina, and Evans 2000; Murphy, Casper, and Micire 2001) and *humanitarian de-mining* (Habib 2007; Santana, Barata, and Correia 2007). In urban search and rescue, robots explore an enclosed space, such as a collapsed building, and try to locate human victims. In humanitarian de-mining, robots explore an open space, such as a field in a war zone, and search for anti-personnel mines that may be concealed. The goal is to locate mines so that they can be disarmed and the region rendered safe.

In both cases, teams of robots are deployed to locate targets of interest in terrain that is potentially unsafe for people, and in both cases the robots typically need a human operator to help with parts of the task that they cannot easily handle on their own. In urban search and rescue, this might be identifying a human victim; and in de-mining, this might be defusing a device that the robot team has located.

In our work (Sklar et al. 2011; 2012), we focus on the use of inexpensive, limited-function robots since we believe that teams of such robots are more practical for wider deployment on the kinds of tasks we are interested in than teams of fewer, more expensive and more capable robots. Although individual robots may require human assistance, large teams of robots present additional challenges for human-robot interaction. There are many ways in which robots can profitably learn from a human trainer, and perhaps lessen the need for human assistance in some circumstances.

This paper reports on a feasibility study of one such instance, where a human operator suspends robot movement in order to avert collisions. Our preliminary investigation involved collecting data from 5 human subjects and assessed the possibility of learning effective behavior patterns from this data. The results detailed here are promising and warrant further investigation.

Related Work

The idea that an interactive system can improve its behavior through observation of human users' key strokes and mouse clicks, i.e., *data mining the clickstream*, is not new. In the 1960's and 1970's, Teitelman developed an automatic error correction facility that grew into DWIM (Do What I Mean) (Teitelman 1979). In the early 1990's, Cypher created Eager, an agent that learned to recognize repetitive tasks in an email application and take them over from the user (Cypher 1991). Maes used machine learning techniques to train agents to help with email, filter news messages, and recommend entertainment. These agents gradually gained confidence in their understanding of users' preferences (Maes 1994).

In robotics, the idea that robots can learn from humans has been explored with *learning from demonstration* (Argall et al. 2009), also known as *programming by demonstration* (Hersch et al. 2008). This is commonly viewed as a form of supervised learning, which learns a policy from a sequence of state/action pairs (Argall et al. 2009). Other approaches to robots learning from people include (Katagami and Yamada 2000), where human teachers provide examples that seed evolutionary learning; (Lockerd and Breazeal 2004), where a robot tries to identify a human's goal and make its own plan to achieve it; and (Nicolescu and Matarić

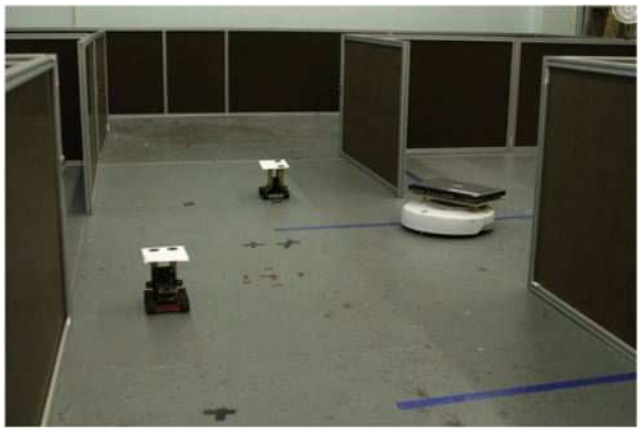


Figure 1: The robots' physical environment.

2001), 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 (Matarić 1997; Parker 2000; Bowling and Veloso 2003; Pugh and Martinoli 2006), but these involve learning by trial and error, not learning from a human teacher.

In earlier related work, we learned behaviors from data collected while humans played video and educational games, using these data to train neural networks that then served as controllers for opponents playing these games (Sklar 2000; Sklar, Blair, and Pollack 2001). The aim was not to produce the best player, but rather to derive a population of players that represented different characteristics of play. This technique was later extended beyond games to generate populations of agents that emulated students performing at different skill levels on an educational assessment (Sklar et al. 2007; Sklar and Icke 2009). The same idea is applied here, to capture and then represent characteristic behaviors of different human operators as they interact with a multi-robot team performing a complex task.

Our Approach

The work reported here involves a single human operator interacting with a team of three robots. This section describes the physical environment in which experiments were carried out and the design of the experiments.

Physical setup

Our experimental testbed, or *arena*, models the interior of a building, with a large space of six rooms and a hallway. The robots explore this space, as described below. The physical testbed is shown in Figure 1, and maps of the arena are shown in Figures 3 and 4. The area of the arena is approximately 400 square feet.

The robots used for the experiments described here are Surveyor SRV-1Blackfins[1], which is a small tracked platform equipped with a webcam and 802.11 wireless. The Blackfin is pictured in Figure 2a. Localization is provided by a network of overhead cameras (Sklar et al. 2011). To help these cameras distinguish between robots, a "hat" is mounted atop each Blackfin (see Figure 2b). Each hat contains a unique symbol, selected from the set of Braille letters that do not have any rotational symmetry; thus the hat provides orientation as well as position.

[1] `http://www.surveyor.com/SRV_info.html`

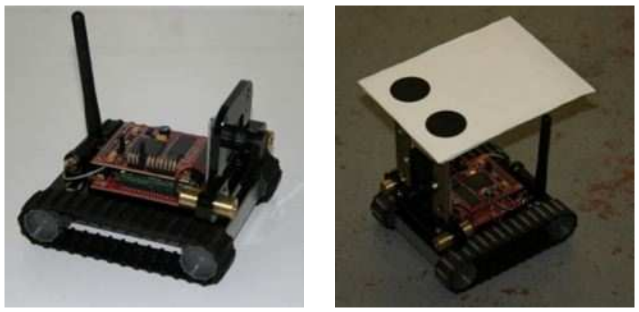

(a) unmodified (b) with hat

Figure 2: The Surveyor SRV-1 Blackfin.

Because the Blackfin has limited on-board processing, the controller for these robots runs off-board, communicating with the robot over a wireless network. The robot controllers, software that allocates tasks to robots (described in (Sklar et al. 2012)), and software that extracts robot positions from the overhead cameras run on a small network of computers facilitated by a central server process. Details of the software architecture are described in (Sklar et al. 2011).

Motivation

As explained above, efficient exploration of the physical space is a key feature of the tasks that we would like our robot team to perform. We have been running experiments in which robots are allocated particular *interest points* to visit. As described in (Sklar et al. 2012), a market-based component allocates these points to the robots on the team. Each robot controller then calculates a path that spans all the points allocated to that robot, with no knowledge of what other robots are planning to do. The robots then simultaneously manoeuvre to their allocated points.

Given that the robots are in a restricted space, that they may start in the same part of the space (modeling situations where all robots enter a space from the same point), and that they have no knowledge of what the other robots are planning to do, the robots naturally get in each other's way. This mutual interference is clear in Figure 3, which shows the motion of three robots during one experimental run. Because of this interference, the robot controller is programmed to prevent collisions, and it does so very conservatively: it suspends any robots that get too close to one another, based on a fixed threshold distance. The robot that is closest to its target interest point is given right-of-way and plans a modified route around the stationary robots. Meantime, the other robot(s) wait until their paths are cleared and then they are allowed to move again. The amount of time the robots wait

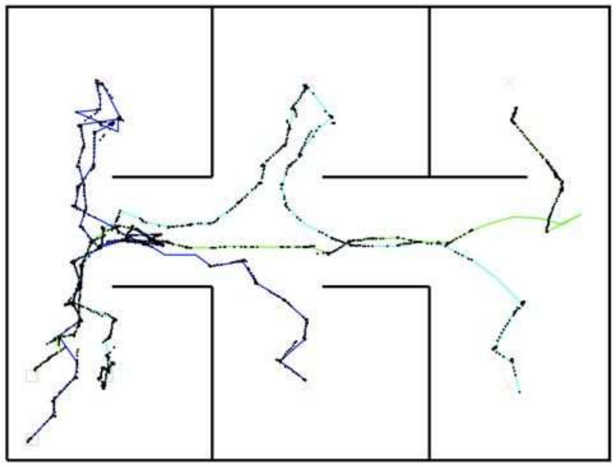


Figure 3: Paths traced by three robots exploring the arena. The robots start in the lower left "room," at the points marked with black □'s. Each robot visits an assigned subset of the 8 "interest points" indicated by red ×'s.

is recorded as *delay time*, which can accrue rapidly if many robots are trying to manoeuvre in the same small space.

This *fixed-threshold method* is a naïve way to prevent collisions. So we wanted to investigate the question of whether a human operator could provide a better model of how avoid collisions, in a way that ultimately reduces overall delay time. The next section describes our initial attempt to learn such behaviors from human operators.

Experimental setup

The robot team was set up in the configuration shown in Figure 3, with all three robots starting in the same room in the arena. The team was allocated eight interest points, and these were distributed among the robots as described in (Sklar et al. 2012). The robots then planned how to visit their assigned points and started to follow the paths that they had planned. The conservative collision avoidance mechanism mentioned above was disabled.

In the experiments described here, collision avoidance was handled by the human operator. This person sat at a workstation physically remote from the robots' arena, in a separate section of the research lab. The human operator could not see or hear anything from the arena. The only information that the operator had about the robots appeared on a user interface, shown in Figure 4. The interface displays the current position and orientation of each robot and its planned path to its current interest point. The robots' position information is derived by the overhead cameras and is therefore subject to error and time lag.

The operator could send two different commands to the robots using the interface: wait and resume. To send a command, the user first clicks on the robot's icon to select it, and then presses a key corresponding to the command. Feedback from human subjects participating in the experiment indicated that this input method was somewhat cumbersome. An alternate input method is under consideration where the user clicks on a robot to pause it if the robot is moving or to resume its movement if the robot is stopped.

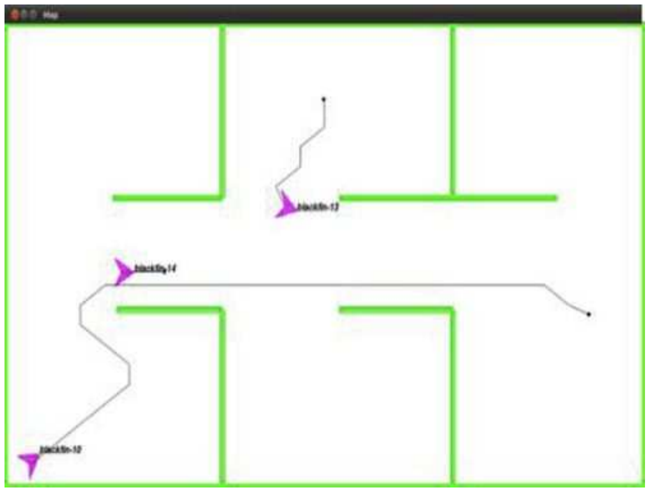


Figure 4: The user interface presented to the human trainer.

For each experimental run, three robots were positioned in the same starting locations (as per Figure 3), eight interest points were allocated, and the robots visited their assigned points while the human operator monitored the team for collisions. In each run, the operator made the robots wait when she thought stopping was necessary to avoid a collision, and she resumed their movement when she judged that the danger of collision was past. A run ended when the robots all reached their final interest point or when a robot collided with another robot or a wall.

Experiments and Results

The results reported here were obtained from data collected on five human subjects (considered "trainers" for the purposes of the behavior learning described below), each of whom was an undergraduate researcher working in our lab (2 female, 3 male). Some of the trainers had previously tested the interface, while others had not used it before; but in neither case was any extensive training required. When a run ended with a robot colliding with a wall, we discarded the data and repeated the run. Thus, each trainer completed five runs which ended either with a robot-robot collision or with successful completion of the full task.

During each run, the system logged data continuously, including the current positions of the robots and each robot's path to its target location (i.e., sequence of waypoints). Each time the human operator sent a command to stop a robot (presumably to avoid a collision with another robot), a "Wait" command was logged for that robot. Similarly, whenever the operator sent a command to a robot to resume its movement, a "Resume" command was logged. Thus, the log file for each run can be used to reconstruct the world state for each robot, at each point in time; and these states can be labeled with the human trainer's decision to perform an action or to do nothing (i.e., let the system perform autonomously).

	number of samples	*four categories*				*two categories*	
		moving-moving	moving-waiting	waiting-moving	waiting-waiting	no-action	action
complete set of training data							
H1	$20,800$	$20,768$	16	16	0	$20,768$	32
H2	$17,103$	$17,058$	23	22	0	$17,058$	45
H3	$22,503$	$22,457$	23	23	0	$22,457$	46
H4	$15,221$	$15,148$	35	35	3	$15,151$	70
H5	$17,451$	$17,389$	31	31	0	$17,389$	62
balanced subset of training data							
H1	65	32	16	16	0	32	32
H2	91	45	23	22	0	45	45
H3	92	45	23	23	0	45	46
H4	146	72	35	35	3	70	70
H5	125	62	31	31	0	62	62

Table 1: Training data. Each Hi indicates one of the 5 human trainers.

A robot's state is represented as:

$$\langle r_1, \theta_1, r_2, \theta_2, V_x, V_y, H_x, H_y \rangle \qquad (1)$$

where r_1, θ_1 and r_2, θ_2 are the range (in cm) and angle (in radians), respectively, from the subject robot to the other two robots in the arena; V_x and V_y are the x and y velocities of the subject robot, and H_x and H_y are the heading of the subject robot (i.e., the x and y distance from its current location to the next waypoint in its planned path). Distances are computed as straight-line Euclidean distances, without consideration of intervening walls or other robots or obstacles.

We analyzed the collected data to address three questions: (1) the ability to learn behaviors from log data, (2) the ability to distinguish between trainers, and (3) the difference between the trainers' methods and the fixed-threshold method for collision avoidance.

Learning from log data

To address the question of whether the log files could be used to learn behavior patterns for collision avoidance, we first needed to label the logged data. As detailed below, we compare two methods of labeling the data. The first method assigned labels from four different categories, and the second method assigned labels from two different categories.

Table 1 describes the data collected from the five trainers, listed as H1 through H5. The first column contains the total number of samples in the training set produced by each operator; the next four columns contain the number of instances labeled in each of four categories; and the last two columns contain the number of instances labeled in each of two categories. The four-category labels are:

- moving-moving, which indicates that the robot was moving when the message was logged;
- moving-waiting, which indicates that the human operator sent a moving robot a command to suspend its motion;
- waiting-moving, which indicates that the human operator sent a suspended robot a command to resume its motion; and
- waiting-waiting, which indicates that the robot was waiting when the message was logged.

The two-category labels are:

- action, which indicates that the human operator did something: either sent a moving robot a command to suspend its motion or sent a suspended robot a command to resume its motion; and
- no-action, which indicates that the human operator did not enter any input when the message was logged.

First, we describe our efforts to learn which states, represented as in equation (1), belong to which of the four-category labels. The upper portion of Table 1, however, clearly indicates that the vast majority of instances are moving-moving. This imbalance has a serious negative impact on learning: most classifiers will simply default to the moving-moving label, since assignment of this overwhelming majority label will be correct 99.72% of the time. A robot guided by such a classifier would never wait. In order to mitigate this situation, we created a balanced training set by selecting a subset of the complete training set of each human operator. The balanced subset was created by retaining all the non-moving-moving instances, and then selecting an equal number of moving-moving instances from the complete data set, at equally-spaced intervals in the chronologically-ordered log file. Statistics for the balanced training set appear in the lower half of Table 1.

We employed the WEKA[2] tool (version 3.6.7) (Witten, Frank, and Hall 2011) and tested 10 different classifiers, using 10-fold cross validation: *k*-nearest neighbors, C4.5 decision trees, a rule extractor from a decision tree (PART), naïve Bayes, Holte's OneR rule learner, a support vector machine (SMO), logistic regression, AdaBoost, logit boost, and decision stumps. The default parameters were used for each method (values are listed in the Appendix).

We ran each classifier on the balanced subset data, attempting to learn the four-category labels. Table 2 shows

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

the accuracy of the best classifier for each trainer and contains the percentage of correctly classified instances. None is much better than random.

H1	rule learner	54.69%
H2	support vector machine	50.00%
H3	C4.5 decision tree	59.34%
H4	logistic regression	56.16%
H5	rule learner	56.80%

Table 2: The best results on learning 4 classes from the balanced subset data.

The resulting confusion matrices clearly indicate that the moving-waiting and waiting-moving labels are not reliably distinguished. For example, Table 3 shows the confusion matrix for the best result learning four categories from the balanced subset data for trainer H3 (corresponding to the third row in Table 2). Clearly, moving-waiting and waiting-moving are confused with each other more often than either is classified correctly. Inspecting the state data reveals that the conditions under which the human operator chooses to suspend a robot's movement because it is about to collide with another robot are almost identical to the conditions under which the operator determines that it is safe for a suspended robot to begin moving again.

classified as $\rightarrow$	moving-moving	moving-waiting	waiting-moving
moving-moving	38	4	3
moving-waiting	5	13	5
waiting-moving	5	15	3

Table 3: Confusion matrix for the best result learning four classes from the balanced subset data for H3. This training set did not contain any instances of waiting-waiting because H3 never invoked that state.

Next, we describe our efforts to learn which states, represented as in equation (1), belong to which of the two-category labels. The two categories distinguish between the states in which the human performs an action and when the human does nothing. The moving-moving and waiting-waiting instances were relabeled as no-action instances, and the moving-waiting and waiting-moving instances were relabeled as action instances. We ran each classifier again, on the balanced subset data, this time attempting to learn the two-category labels. Table 4 shows the best results, which represent a considerable improvement over Table 2.

Table 5 shows the confusion matrix for the best of these classifiers, on the balanced subset data for trainer H1. Again, this is a substantial improvement over the results shown in Table 3. Our current work is investigating other techniques to address the imbalanced data, in order to be able to use more of the data for training.

H1	k-nearest neighbor	90.63%
H2	logit boost	76.67%
H3	logit boost	87.91%
H4	AdaBoost	82.86%
H5	k-nearest neighbor	87.10%

Table 4: The best stratified 10-fold cross validation results, learning 2 classes from the balanced subset data.

classified as $\rightarrow$	no-action	action
no-action	30	2
action	4	28

Table 5: Confusion matrix for the best result learning 2 classes from the balanced subset data (H1).

Distinguishing between trainers

To address the question of whether the data could be used to differentiate between trainers, we examine the decision trees produced by the C4.5 classifier. Although the operator-guided collision avoidance task appears relatively simple and straightforward, distinct differences between operators' behaviors are detected. Figure 5 shows the C4.5 trees learned for the two most accurately modeled trainers.

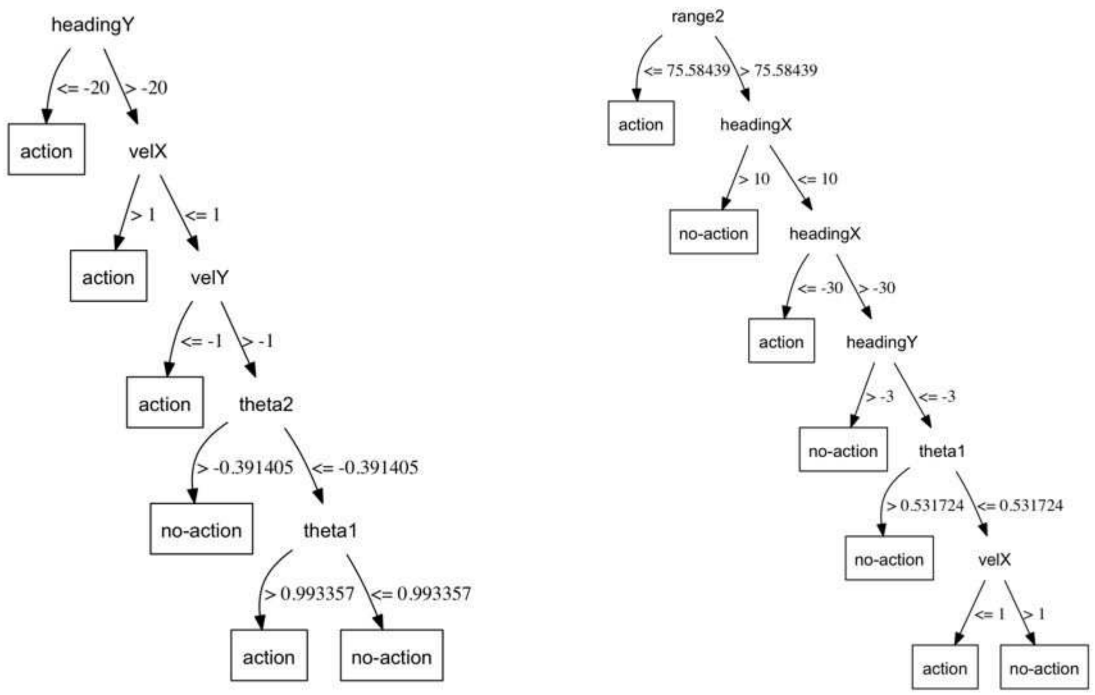
It is interesting to note that the decision trees for different trainers depend on different aspects of the robot's state. The decisions of H1, as captured by the decision tree, depend on the direction that the robot is heading (headingY, its speed (velX (V_x) and velY (V_y)), and the directions that the other two robots are heading (theta1 (θ_1) and theta2 (θ_2)). In contrast, the decisions of H3 depend on the distance from one of the other robots (range2 (r_2)), the robot's heading (headingX (H_x) and headingY (H_y)), its speed (velX (V_x)) and the direction of the third robot (theta1 (θ_1)), not the same robot as the one whose distance is considered. H1 appears to focus on trajectories alone, while H3 attends first to proximity.

We also analyzed the differences in behaviors identified by the decision stump results. Table 6 shows the rules obtained using this method for the five trainers. Again, it is clear that the decisions of different humans, as extracted by the decision stump algorithm, are influenced by different components of the robot's state.

While these observations are preliminary, and we have not yet collected enough data from human subjects to produce definitive models of particular individuals' behaviors, these results do tell us that our direction is promising. Current work is focused on programming the decision trees and decision stump rules into the robot controllers in order to evaluate how well the system performs when following the strategies modeled on different human subjects.

Comparison to the fixed-threshold method for collision avoidance

To address the question of how collision avoidance decisions made by human operators compare to our fixed-threshold



H1 (85.94%) H3 (86.81%)

Figure 5: Decision Trees (C4.5) for the two most accurately modeled trainers. The percentage of correctly identified instances appears in parenthesis.

H1	(68.75%)	*act if you are heading not too far to the south*
H2	(66.67%)	*act if you are going quickly west*
H3	(71.43%)	*act if you are closer than 77cm to one of the other robots*
H4	(75.00%)	*act if you are heading not too far to the east*
H5	(78.23%)	*act if you are closer than 124cm to one of the other robots*

Table 6: Decision Stump rules for all five trainers, with the percentage of correctly classified in instances in parenthesis.

method, we analyzed the distances between robots when the trainers suspended and resumed robots' motion. Our intuition and initial hypothesis was that the human operators would allow robots to get closer to each other than the fixed-threshold method would. As a result, the amount of time that a robot spent waiting for others to move out of the way would decrease. The fixed-threshold method used a distance of 50cm, with results reported in (Sklar et al. 2012).

Interestingly, the human operators who made decisions based on distances between robots (i.e., either r_1 or r_2) acted on distances greater than 50 cm: 77cm and 124cm in the decision stump rules for H3 and H5, respectively; and 76cm, 61cm and 75cm in the decision trees for H3, H4 and H5, respectively. The other trainers, however, did not consider the distance to other robots at all. Instead, they examined other factors, such as the directions in which robots were heading. In the decision trees, where more complex decision rules can be coded, even those trainers who did consider distance also considered the directions in which the robots were heading. So our preliminary conclusion is that the fixed-threshold method does not take into consideration all the important factors in collision avoidance. Our current work is analyzing the delay time for all the experiments described here to identify whether the humans' performance in regard to this metric is better than that of the system when using the fixed-threshold method for collision avoidance.

Summary

This paper presents preliminary results from experiments in which a human guides a team of robots to avoid collisions. We recorded data from human operators who were instructed to help members of a robot team to avoid collisions with one another. Operators could suspend or resume robots' movement. Based on data recorded while the human subjects were engaged in this task, we trained different classifiers to predict when the robots should change state (toggling from moving to waiting and waiting to moving). The best-fitting classifier achieved 90% accuracy. A more rigorous evaluation will use the learned classifier rules to control the robots autonomously and measure their ability to avoid collisions. This is necessary since the fact that we can learn a classifier that performs well on the data we logged does not mean that we have learned a classifier that will be good at stopping robots from colliding. Performing this validation step on the robots is the next major activity in this investigation.

Acknowledgments

This work was supported by the National Science Foundation under grants #IIS-1117000 and #CNS-0851901.

Appendix: WEKA defaults

weka.classifiers.trees.J48 (C4.5): confidence threhold for pruning = 0.25; minimum number of instances per leaf = 2; seed for random data shuffling = 1.

weka.classifiers.lazy.IBk (k-nn): nearest neighbour search algorithm = weka.core.neighboursearch.LinearNNSearch; number of nearest neighbours (k) used in classification = 1.

weka.classifiers.rules.PART: confidence threhold for pruning = 0.25; minimum number of instances per leaf = 2; seed for random data shuffling = 1.

weka.classifiers.bayes.NaiveBayes (naïve Bayes without kernel): use normal distribution for numeric attributes.

weka.classifiers.rules.OneR (OneR): minimum number of objects in a bucket = 6.

weka.classifiers.functions.SMO (svm): kernel = weka.classifiers.functions.supportVector.PolyKernel; complexity constant C = 1; normalize; epsilon for round-off error = $1.0e - 12$; use training data for internal cross-validation; random number seed = 1; size of the cache = 250007; exponent = 1.0; do not use lower-order terms.

weka.classifiers.functions.Logistic (logistic regression): maximum number of iterations = -1 (until convergence).

weka.classifiers.meta.AdaBoostM1 (AdaBoost): percentage of weight mass to base training on = 100; random number seed = 1; number of iterations = 10; base classifier = weka.classifiers.trees.DecisionStump.

weka.classifiers.meta.LogitBoost (logit boost): percentage of weight mass to base training on = 100; number of folds for internal cross-validation = 0 (no cross-validation); number of runs for internal cross-validation = 1; threshold on the improvement of the likelihood = $-\mathrm{Double.MAX\_VALUE}$; shrinkage parameter = 1; random number seed = 1; number of iterations = 10; base classifier = weka.classifiers.trees.DecisionStump.

References

Argall, B.; Chernova, S.; Browning, B.; and Veloso, M. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Bowling, M., and Veloso, M. 2003. Simultaneous adversarial multi-robot learning. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.

Cypher, A. 1991. Eager: Programming repetitive tasks by example. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*.

Habib, M. K. 2007. Humanitarian Demining: Reality and the Challenge of Technology. *Interational Journal of Advanced Robotic Systems* 4(2):151–172.

Hersch, M.; Guenter, F.; Calinon, S.; and Billard, A. 2008. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*.

Jacoff, A.; Messina, E.; and Evans, J. 2000. A standard test course for urban search and rescue robots. In *Proceedings of PerMIS*.

Katagami, D., and Yamada, S. 2000. Interactive classifier system for real robot learning. In *IEEE International Workshop on Robot and Human Interaction*, 258–263.

Lockerd, A., and Breazeal, C. 2004. Tutelage and socially guided robot learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Maes, P. 1994. Agents that reduce work and information overload. *Communications of the ACM* 37(7):31–40.

Matarić, M. 1997. Reinforcement learning in the multi-robot domain. *Autonomous Robots* 4:73–83.

Murphy, R. R.; Casper, J.; and Micire, M. 2001. Potential tasks and research issues for mobile robots in RoboCup Rescue. In *Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Artificial Intelligence*. Springer.

Nicolescu, M. N., and Matarić, M. J. 2001. Learning and interacting in human-robot domains. *IEEE Transactions on Systems, Man, and Cybernetics* 31(5):419–430.

Parker, L. 2000. Multi-robot learning in a cooperative observation task. In *Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems*.

Pugh, J., and Martinoli, A. 2006. Multi-robot learning with particle swarm optimization. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*.

Santana, P. F.; Barata, J.; and Correia, L. 2007. Sustainable Robots for Humanitarian Demining. *International Journal of Advanced Robotic Systems* 4(2):207–218.

Sklar, E. I., and Icke, I. 2009. Using simulation to evaluate data-driven agents. In *Multi-agent Based Simulation IX*, volume 5269 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Sklar, E. I.; Salvit, J.; Camacho, C.; Liu, W.; and Andrewlevich, V. 2007. An agent-based methodology for analyzing and visualizing educational assessment data. In *Proceeding of the Sixth International Conference on Autonomous Agents and Multiagent Systems*.

Sklar, E. I.; Özgelen, A. T.; Muñoz, J. P.; Gonzalez, J.; Manashirov, M.; Epstein, S. L.; and Parsons, S. 2011. Designing the HRTeam framework: Lessons learned from a rough-'n-ready human/multi-robot team. In *Proceedings of the Workshop on Autonomous Robots and Multirobot Systems*.

Sklar, E. I.; Özgelen, A. T.; Schneider, E.; Costantino, M.; Muñoz, J. P.; Epstein, S. L.; and Parsons, S. 2012. On transfer from multiagent to multi-robot systems. In *Proceedings of the Workshop on Autonomous Robots and Multirobot Systems*.

Sklar, E. I.; Blair, A. D.; and Pollack, J. B. 2001. Training intelligent agents using human data collected on the internet. In *Agent Engineering*. Singapore: World Scientific. 201–226.

Sklar, E. I. 2000. *CEL: A Framework for Enabling an Internet Learning Community*. Ph.D. Dissertation, Department of Computer Science, Brandeis University.

Teitelman, W. 1979. A display oriented programmer's assistant. *International Journal of Man-Machine Studies* 11:157–187.

Witten, I. H.; Frank, E.; and Hall, M. A. 2011. *Data Mining, Practical Machine Learning Tools and Techniques*. Elsevier Inc., third edition.