Intelligent Agent Supporting Human-Multi-Robot Team Collaboration*

Ariel Rosenfeld[1,†], **Noa Agmon**[1], **Oleg Maksimov**[1] **Amos Azaria**[2], **Sarit Kraus**[1]
[1] Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel 52900.
[2] Machine Learning Department, Carnegie Mellon University, Pittsburgh, USA.
[†] Corresponding author, rosenfa5@cs.biu.ac.il.

Abstract

The number of multi-robot systems deployed in field applications has risen dramatically over the years. Nevertheless, supervising and operating multiple robots at once is a difficult task for a single operator to execute. In this paper we propose a novel approach for utilizing advising automated agents when assisting an operator to better manage a team of multiple robots in complex environments. We introduce the *Myopic Advice Optimization (MYAO) Problem* and exemplify its implementation using an agent for the Search And Rescue (SAR) task. Our intelligent advising agent was evaluated through extensive field trials, with 44 non-expert human operators and 10 low-cost mobile robots, in simulation and physical deployment, and showed a significant improvement in both team performance and the operator's satisfaction.

1 Introduction

Multi-robot systems are being applied to different tasks, such as Search And Rescue (SAR) [Liu and Nejat, 2013], automatic aircraft towing [Morris *et al.*, 2015], fire-fighting [Saez-Pons *et al.*, 2010], underwater missions [Kulkarni and Pompili, 2010] and construction [Parker and Zhang, 2002]. Common to most of the work in multi-robot systems is the assumption that either the robots are autonomous, or they are controlled centrally by one computer. A hidden assumption in this case is that the robots perform relatively smoothly, with the infrequent need to overcome failures.

The deployment of low-cost robots in real-world environments has shown that they usually face difficulties in completing their tasks. Specifically, failures are common. In such situations, a human operator must get involved in order to solve the problem. That is, robots are usually *semi-autonomous*, and should be supported by an operator whenever they cannot handle the situation autonomously. For example, during the deployment of robots at the World Trade Center disaster, each robot got stuck 2.1 times per minute (on average), and required human assistance [Casper and Murphy, 2003].

In the context of multi-robot systems, the supervision and control of multiple robots at once can be overwhelming for a single human operator—resulting in sub-optimal use of the robots and a high cognitive workload [Chen and Terrence, 2009; Squire and Parasuraman, 2010]. Wang et al. [2009] claimed that this *fan-out* (the number of robots that a human operator can effectively operate at once) plateau lies "somewhere between 4 and 9+ robots depending on the level of robot autonomy and environmental demands".

Improving the performance of supervised multi-robot systems can be done using one of the following dominant approaches: Either (1) Improving the robot's hardware and software—thus, relying less on human supervision (making the robots more autonomous), or (2) Improving the efficiency of the Human-Robot Interaction (HRI). Assuming we are given a team of robots, and we cannot control the reliability of its hardware or software, this paper deals with improving the HRI in order to allow a person to control a team of many (possibly unreliable) robots.

As shown in most multi-robot systems controlled by a human operator, a single operator may get overwhelmed by the number of requests and messages, resulting in sub-optimal performance. For example, Chien et al. [Chien *et al.*, 2013] have studied robots that could self-report encountered faults. In their reported experiment, participants performed the foraging task while assisted by an alarm system, under different task loads (3 robots vs. 6 robots). The results show that participants in the 6-robot scenario did not perform better than those controlling only 3, while some even performed significantly worse. The results also show that operators devoted their resources in a sub-optimal way, leaving fewer resources for more urgent and critical tasks. These findings are consistent with previous studies with ground robots [Velagapudi and Scerri, 2009; Wang *et al.*, 2009; Chen *et al.*, 2011; Squire and Parasuraman, 2010; Lewis, 2013] and areal robots [Miller, 2004; Cummings *et al.*, 2007; Panganiban, 2013]. Rosenthal and Veloso [2010] suggest a different approach, in which robots should request (and receive) help from humans for actions they could not have performed alone due to lack of capabilities. However, in the presence of many robots' requests, Rosenthal and Veloso's approach could (potentially) overwhelm an operator.

In this paper, we present a novel methodology that enhances operators' performance by using an intelligent advising agent. The agent provides advice for the operator regarding which actions she should take and acts as a

*We thank Cogniteam Ltd. for all their support.

smart filter between the robots' requests and the human operator. Our methodology is not restricted to any specific hardware or algorithm used by the robots, and we consider these factors constants. Intelligent advising agents have been successfully implemented in different domains, for example [Rosenfeld and Kraus, 2015; Rosenfeld *et al.*, 2015; Azaria *et al.*, 2015]. However, to the best of our knowledge, this is the first work on utilizing advising agent technology in the HRI field.

We will first present the *Myopic Advice Optimization (MYAO) Problem*. The optimization problem models the maximization of the operator's performance by selecting when and which advice to provide in a greedy fashion. Then, we evaluate our approach using the SAR task. In the SAR task, a single human operator has to search for certain objects[1] using a large team of unreliable robots (see [Liu and Nejat, 2013] for a review on SAR settings). We present an intelligent advising agent which solves the MYAO problem (adapted to our SAR task).

We assume *non-expert* operators, which are common in many real world applications. For example, firefighters which deploy robots in burning buildings to detect flame sources or human victims cannot be expected to train solely on the control of robots and cannot be considered to be experts. Furthermore, we could not expect a fire squad to have a robot expert in every given moment of the watch. Nevertheless, we assume that the operator has decent technological skills and underwent some (basic) training with the robots.

We have tested extensively the performance of our advising agent in both simulated environments (using the Gazebo robot simulation toolbox[2]) and physical deployment (using Hamster robots (Figure 2)) with 44 non-expert operators. Experimental results show that our advising agent was able to enhance the operators' performance when managing a team of 10 mobile robots in terms of the task's goals (finding and correctly classifying green objects in a clustered terrain) and reduced the cognitive workload reported by the operators in two distinct SAR environments: an office building floor and a simulated urban open terrain. On average, while equipped with our agent, an operator covered 19.5% more terrain, detected and correctly classified 16.7% more desired objects, reported 7% less workload (using NASA-TLX [Hart and Staveland, 1988]) and reduced 22.3% of the robots' idle time, compared to her benchmark performance without our agent while operating 10 mobile robots. Despite training with simple simulated environments, our agent has shown that it is able to provide solid advice in both complex simulated environments and physical deployment.

The main contribution of the paper is in showing, for the first time, that an intelligent agent that supports the operator can lead to better performance of the human-multi-robot team.

A short video of the project is available at *http://vimeo.com/119434903*.

[1] The objects can be victims, flames, weapons, etc.

[2] *http://www.gazebosim.org/*

2 The Advice Optimization Problem

In this work we consider a set of N *semi-autonomous* robots engaged in a cooperative task, supervised and controlled by a *single, non-expert* human operator, denoted by O. The *state space* S consists of all information regarding the robots (e.g., robot location, battery capacity, operational status) which is domain specific (an instance of the state space is denoted by $s \in S$). The operator, O, can perform *actions* during the task from a predefined set—A. Note that O can choose to execute no actions, i.e., $NULL \in A$. *Advice* is defined as a possible action $a \in A$ suggested by the system for the operator to perform (note that the operator is not obligated to accept the advice, i.e., to perform this action).

In state s, the operator, according to his abilities, endures a cost (usually in terms of time) for performing action a, denoted by $C_o(s, a)$. If a is *infeasible* in state s (defined by the domain characteristics), then $C_o(s, a) = \infty$. We assume that $C_o(s, NULL) = 0$. We refer to this cost function as the *operator model*.

The *transition function*, denoted by $P_o(s_1, a, s_2, C_o(s_1, a))$, provides the probability of reaching state s_2 given action a in state s_1 and the operator model.

The *reward function*, $R_o(s)$, provides a real value representing the expected reward of state s in terms of task fulfillment. For example, in a janitorial task, one can define $R_o(s)$ to be the expected time to finish cleaning the area given the current state s.

$\gamma \in (0, 1]$ is the discount factor, which represents the importance difference between future (uncertain) rewards and present rewards. For instance, γ can capture the uncertainty of receiving future rewards.

Ideally, we would like the operator to execute the **optimal policy** $\pi_o^* : S \to A$ which maximizes the expected accumulative reward given $S, A, P_o(\cdot), R_o(\cdot), C_o(\cdot)$ and γ. Finding π_o^* naturally translates into a Markov Decision Process (MDP) [White III and White, 1989] consisting of state space S, action space A, transition function P_o, discount factor γ and a reward function determined by R_o and C_o. However, calculating π_o^* is generally intractable, due to the exponential size of the state space and the high uncertainty induced by the environment, robots and inexperienced operators, making it extremely challenging to use the appropriate MDP. This difficulty also stems from the complexity of many multi-robot problems, such as the NP-hard task allocation problem [Korsah *et al.*, 2013] and the terrain coverage problem [Zheng *et al.*, 2010].

In order to cope with the aforementioned factors and adapt in real-time to the dynamically changing environment, we consider a more tractable advising problem which uses a myopic heuristic.

2.1 The *Myopic* Advice Optimization Problem

The Myopic (minimization[3]) Advice Optimization (MYAO) Problem is defined as follows.

Given state $s \in S$, offer advice a^* such that:

$$a^* = \text{argmin}_a \gamma^{C_o(s,a)} \cdot \int_{s' \in S} P_o(s, a, s', C_o(s, a)) \cdot R_o(s') ds'$$

[3] The maximization form uses argmax instead.

At time t, a^*—which minimizes (maximizes) the expected *short-term* reward – is selected (See Algorithm 1). By definition, a^* is relatively simple to calculate and comprehend given s at current time t. This property holds an additional advantage as people are more likely to adhere to an agent's advice which they can understand [Elmalech *et al.*, 2015].

Algorithm 1 Advice Provision

```
1: s ← InitialState()
2: repeat
3:     min ← ∞
4:     for each a ∈ A do
5:         expRwd ← E_{s'∈S}[R(s')]
6:         if γ^{C(s,a)} · expRwd ≤ min then
7:             advice ← a
8:             min ← γ^{C(s,a)} · expRwd
9:     OUTPUTadvice
10:    Spin(k−millisec).
11:    s ← GetCurrentState()
12: until TerminalCondition(s)
```

3 Search And Rescue

In this section, we instantiate the MYAO problem (Section 2.1) to the Search And Rescue (SAR) task. We will first describe the SAR task in detail, followed by our agent design.

3.1 The SAR Task

In our SAR environment, the operator remotely supervises and controls 10 unmanned ground robots using a computer program (See Figure 3). The operator is required to find and correctly classify green objects in a clustered terrain.

Our SAR task is conducted in two distinct environments: **Environment 1** – An office building floor consisting of an "L" shaped, narrow corridor with small and mid-sized offices (See Figure 4). We evaluate Environment 1 in both simulation and physical deployment. We denote the simulation as Environment 1s and the physical deployment as Environment 1p. **Environment 2** – A medium-sized warehouse yard taken from the popular CounterStrike© computer game called "Assault". "Assault" is considered to be very realistic and is one of the most popular maps in the whole CounterStrike© series[4] (See Figure 1). Environment 2 was only evaluated in simulation and will be denoted as Environment 2s.

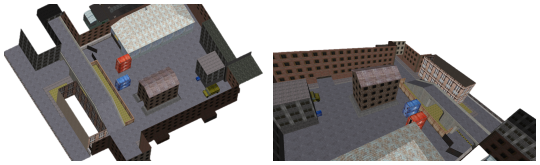

Figure 1: Environment 2s; an open terrain model from CounterStrike©.

We used 10 *Hamster* AUGVs (autonomous unmanned ground vehicles) (See Figure 2). Hamster is a 4WD rugged platform with a built-in navigation algorithm that allows it to explore, map and localize in unknown areas. Hamster has 2 on-board Raspberry PI Linux servers for algorithm execution and an Arduino for low level control. Hamster is mounted with an HD camera with h264 video streaming over WiFi and a 360° 6-meter range LIDAR laser. Each Hamster is 190mm in width, 240mm in length and 150mm in height.

Figure 2: Hamster AUGV; one of the 10 identical robots used in this study.

The Hamster can be either in autonomous mode or manual mode. While in autonomous mode, the Hamster travels through the terrain without the operator's intervention. In our task, the robots are required to explore the environment, given a 2D blueprint of the area (no mapping is required). However, the blueprint does not include randomly placed objects and obstacles scattered in the environment, for example bookstands and cupboards in Environment 1 and containers and barrels in Environment 2. The robots are given their initial position (deployment point) and localize using ICP laser matching [Besl and McKay, 1992] and the AMCL algorithm[5].

The robots execute a simple exploration algorithm, based on the given map. Given an estimation of the location of all robots in the team, a robot simply chooses to travel away from its peers, to a less crowded area (thus, a less explored area). The robot travels to its desired destination using the A* path planning algorithm, and uses basic obstacle avoidance techniques to avoid both obstacles and other robots while doing so.

We account for three malfunction types that the Hamster can experience:

Sensing The camera/laser stops working. In such cases, the operator can send the robot back to home base, where the experiment operator can fix it.

Stuck The robot cannot move (for example due to an obstacle), and needs the operator to help it get loose. In some cases the Stuck malfunction is terminal.

WiFi The robot stops sending and receiving data. In such cases the operator can mark the area as a "no entrance" zone. Upon losing the WiFi signal, the robot is programmed to return to its last point of communication.

The GUI (See Figure 3) provides the operator with on-line feedback from the cameras mounted on the robots (Thumbnail area), the 2D map of the terrain including the robots' reported positions and their footprints, an enlarged camera view of the robot of interest, an action control bar, and a joystick widget. The action bar's commands and joystick functions are also available using keyboard and mouse shortcuts inspired by strategic computer games. For example, in order to set interest on a specific robot, the operator could click its thumbnail camera or location on the map or could click on its number on the keyboard. Double clicking will center the map on the robot's location.

[4] *http://counterstrike.wikia.com/wiki/Assault*

[5] *http://wiki.ros.org/amcl*

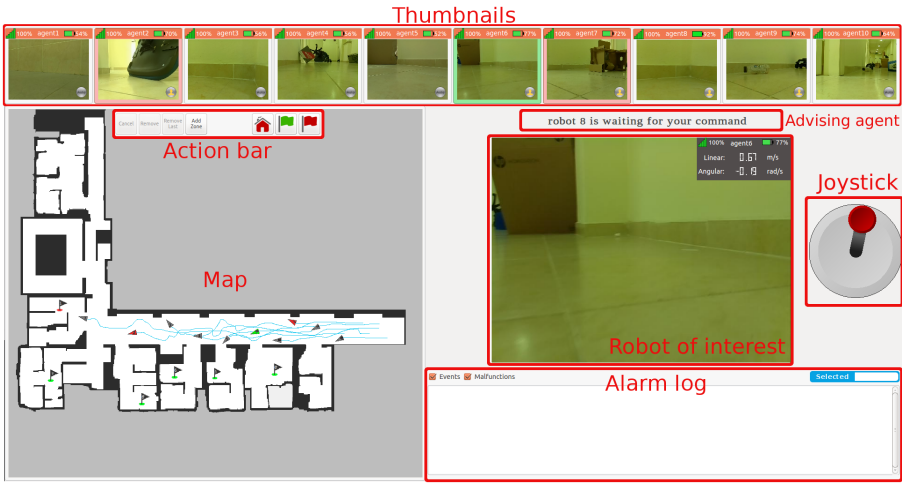

Figure 3: GUI for operating 10 robots (see Figure 4 for the physical robots in action)

The operator can perform the following actions: (1) change the mode for each of the robots (autonomous or manual), (2) send a robot to a desired point on the map using the mouse's right click option (the robot would autonomously navigate to the destination point, when possible), and (3) manually teleoperate a robot using the joystick widget or the keyboard arrow keys. When a robot needs the operator's attention—in case of malfunctions or a detected green object that needs classification—it changes its mode to manual and signals the operator by blinking for 5 seconds and playing a prerecorded, human-voice message in the operator's headset. The operator can view the robots' requests and provide solutions using the alarm log. For the operator's convenience, we maintain a first-come-first-serve alarm log indicating the (active) alarms.

3.2 Our Agent

Our agent implements the MYAO Problem (Section 2.1) in the SAR domain (Section 3.1). Recall that in order to correctly solve the optimization problem we need to estimate $C_o(\cdot)$, $P_o(\cdot)$ and $R_o(\cdot)$. We consider situations where it is not feasible to collect data on a specific operator. For example, when an operator has very little experience in robot supervision and control, and where such experience is expensive to attain. In particular, the agent should be able to support the operator from its first interaction with the system. In such cases, we suggest using *generalized models*.

Our agent uses three generalized models: an *operator model* to approximate $C_o(\cdot)$, a *robot model* to approximate $R_o(\cdot)$ and a transition function to assess $P_o(\cdot)$. We will first define the generalized models and describe their training.

The *operator model* quantifies the operator's abilities and technique in terms of expected time to execute *advice* a in state s. We use a general operator model $C(\cdot)$ independent of specific O, and define $C(\cdot)$ as the expected time for completing a by an average operator.

The *robot model* quantifies the robots' productivity in terms of expected time to find the next green object. In order to collect data on the robots' behavior and productivity, which can be very expensive as an operator is needed to support them, we suggest using a simulated *utopic environment*. In a utopic environment, there will be no need for human interference with the robots' work, as the robots will never malfunction. Accordingly, we define $R(\cdot)$ to be the expected time to find the next object in the *utopic environment*. That is, we learn from a *simulated utopic environment* and we utilize the gathered data to learn an advising policy which is tested in both physical and simulated environments which are far from utopian (Section 3.1). Similarly, we define $P(\cdot)$ as independent of O. Recall that $P(s, a, s', C(s, a))$ is the probability for transitioning from s to s' using advice a, given $C(s, a)$. As S is continuous, we cannot assess the actual $P(\cdot)$, yet given s,a and $C(s, a)$ we can consider 2 options; 1) a was performed successfully and reached s'; 2) a was not performed or was unsuccessful and the robots reached state $s'' \neq s'$ (for example, the operator was unable to fix a malfunction). Due to the short time frames in which pieces of advice are executed (SAR is a fast-paced task) , we assume that $s'' = s$.

γ is defined as the probability for the task to end, given the task's expected duration. We assume a geometrical distribution, so γ is set to 1-(the *calculation interval* divided by the *expected task duration*).

Combining the above models, the agent solves the optimization problem (using Algorithm 1) online at every second and provides the best possible advice in both textual format (See Figure 3) as well as a prerecorded, human-voice message. The presented advice might change at time t if the advice was successfully completed or when more urgent and productive advice is available according to the current state s_t, making our agent *adaptive* to the environment. We used an efficient priority queue and updating methods to avoid long calculations.

Training The Models

In order to train the above models, we used the Gazebo robot simulation toolbox. We manually mapped an office building floor (Environment 1) using a 30-meter laser and constructed a 3D virtual model to scale. Also, we attained a 3D model of "Assault" terrain from the CounterStrike source kit (Environment 2). These models were mounted to the Gazebo alongside 10 Hamster robot models using ROS[6]. In our simulation,

[6] *http://www.ros.org/*

the robots experience malfunction according to a pre-defined malfunction schedule which determines when and which malfunctions will occur to each robot.

Recall that we defined $R(s)$ as the estimated time to find the next green object given state s in a utopic environment. Hence, to learn $R(\cdot)$, we ran 150 1-hour sessions on each environment. During these sessions, the robots autonomously searched for 40 randomly placed green objects. In each session we placed random obstacles and used a different number of robots ranging from 1 to 10. To avoid the need for a human operator, we set an empty malfunction schedule and instructed the robots to avoid waiting for an operator's response on detected objects.

In order to construct a generic (non-negative) $R(\cdot)$ function, which can be used in different terrains (both in physical and simulated deployment), we could not use terrain specific features such as the robots' positions, room sizes etc. Hence, using the 150 sessions and a careful feature selection, we estimated $R(s)$ using the following model:
From s, we extract the number of active robots ($Active(s)$), the number of detected objects ($Objects(s)$), the average distance between robots ($Distance(s)$) and the minimal distance between a pair of robots ($Minimal(s)$) and calculate:

$$R(s) = \alpha_0 - \alpha_1 \cdot ln(Active(s)) + \alpha_2 \cdot Objects(s)^2 - \alpha_3 \cdot Objects(s) - \alpha_4 \cdot Distance(s) - \alpha_5 \cdot Minimal(s)$$

Where $\alpha_0, \ldots, \alpha_5$ are learned using regression.

This form of function assumes that there is α_0 time to find an object, which is then reduced by the average distance between robots and the minimal distance between a pair of robots in a linear way. It also assumes that the number of active robots reduces the expected time in a marginally decreasing fashion. The objects detected so far increase the expected time (as objects are harder to find), yet at the early stages of the task finding the first objects is assumed to indicate that the robots have achieved some effectiveness in terms of position. This form of function was compared to more than 100 other forms and yielded the greatest fit to the data we collected in both simulated environments[7]. All parameters are assumed to be positive, and in fact reached positive values.

To learn the operator's model $C(s, a)$ and transition function $P(s, a, s', C(s, a))$, we first define A—the action set that the operator can take. A is also the advice set from which the agent can propose advice. We defined A as the 96 instantiations of the following 6 action schemes: "Tele-operate robot i to a better position", "Tele-operate robot i away from robot j", "Send robot i home for repairs", "Robot i is stuck, try to get it loose", "Robot i detected an object", "Robot i is waiting for your command" and "Spread the robots around, they are too close to each other" where $i, j \in [1, \ldots, 10]$.

To train the models, we recruited 30 Computer Science senior undergraduate students, ranging in age from 18 to 34 (mean=25, s.d.=3.4) with similar demographics, to participate in a simulated SAR task equipped with a *naïve* advising agent. Given state s, our naïve agent provided advice a such that, upon completion, is expected to bring about a new state s', where $R(s') + 5sec < R(s)$. That is, the agent suggests advice that, if completed successfully, will reduce (at least) 5 seconds from the expected time to find the next object. The agent only offers advice which can be completed (for example, if a robot is stuck it will not advise to teleoperate it). The agent presents the advice in both textual format (See Figure 3) as well as a prerecorded, human-voice message. Note that the agent *filters and prioritizes* robots' requests and messages, yet in a naïve way.

For motivation, subjects were paid 1 NIS (about 25 cents) per green object they found, and 0.5 NIS per advice to which they adhere.

Surprisingly, all of the advice that our agents gave was executed and completed. Therefore, we set $P(s, a, s', C(s, a))$ to be 1 whenever completing a advice is possible, and 0 otherwise. When analyzing the operators' behavior, we were unable to find good *cross-subject* features that will help us to predict the expected time it would take to complete advice. Therefore, we used the mean execution time. That is, $C(s, a)$ is equal to the average execution time of a across our subjects.

Despite major differences between our simulated, utopian training environment and the test environments, our approach yields solid advising policies used in both non-utopian simulated environments and in physical deployment of robots. We note that given personalized models, $C_o(\cdot)$, $P_o(\cdot)$ and $R_o(\cdot)$, our agent could also provide personalized advice. However, in the scope of this work these models are unavailable.

Evaluation

To evaluate our agent, we used a within-subject experimental design where each subject performed the SAR task twice (a week apart); once without the agent and once equipped with the agent. We recruited 32 subjects to participate in the simulated SAR task (16 per simulated environment (1s and 2s)) and 12 subjects participated in Environment 1p, totaling 44 subjects. Subjects who participated in the simulated task were BSc and MSc Computer Science students, ranging in age between 20 and 33 (mean 25), 14 females and 28 males, whereas the 10 participants in Environment 1p were researchers and workers from our university. Subjects were counter-balanced as to which condition was applied first.

Each subject was trained before each run; they underwent a structured 1-on-1 hands-on tutorial on the system by our research assistant and had to pass a short test. The test was conducted to make sure the operator was capable of successfully completing the task. During the test, a subject was in control of 3 robots and without any time limit had to find and classify 2 green objects in the terrain while she encountered 3 simulated malfunctions.

The SAR task took 40 minutes (in simulation) and 15 minutes (in physical deployment) in which the subjects had to search for the green objects placed in predefined (yet, randomly selected) positions in the environment. The subjects were motivated to find and classify as many green objects as possible using a small monetary reward (1 NIS per object). However, they did not receive any monetary reward for accepting the agent's advice. To assure all malfunctions and objects are equally distributed in the environment, we used the

[7]Some of the other functions that were tested included one or more of the following modifications to the above function: the use of $Active(s)$ or $Objects(s)$ as an additive variable; $Active(s)$ as having a multiplicative impact or $Objects(s)$ as having an impact depending on $Distance(s)$ or $Minimal(s)$.

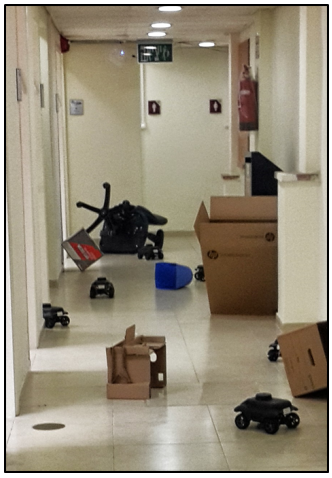

Figure 4: Part of Environment 1p during an experiment (see Figure 3 for the operator's point of view of this environment).

same malfunction schedule and object positions in both conditions. After completing the task, subjects were asked to answer a standard NASA-TLX questionnaire [Hart and Staveland, 1988].

Results reported in this section were found to be significant using the Wilcoxon Signed-Rank Test (an alternative to paired-sample t-test for dependent samples when the population cannot be assumed to be normally distributed).

In **Environment 1s (simulated office building)**, we placed 40 green objects around the office. We set a malfunction schedule such that each robot encounters 1.5 malfunctions (on average) during the simulation. All robots started from the main entrance of the floor. Subjects were given 40 minutes to complete the task. The results show a statistically significant increase in the average number of detected objects by the robots (37.7 vs. 31.2 objects, $p < 0.001$) as well as their average covered terrain (529 vs. 462 square meters, $p < 0.05$) for the condition in which the agent was present. Furthermore, a decrease in the average time that robots stay idle (1540 vs. 1860 seconds, $p < 0.05$) and a reduced average workload (55 vs. 62 TLX, $p < 0.1$) were also recorded.

In **Environment 1p (real deployment in an office building)**, we scattered 20 green objects (see Figure 4). We set a malfunction schedule such that every robot encounters 0.8 malfunctions (on average) during the deployment. Half of the robots started from the main entrance of the office space, whereas the other half started from the back entrance. Operators were given 15 minutes to complete the task. A major **100%** increase in the average number of detected objects was recorded (14.1 vs. 7 objects, $p < 0.001$) as well as an increased average for covered terrain (462 vs. 305 square meters, $p < 0.05$). A significant decrease in the average time that robots stay idle (2720 vs. 3244 seconds,$p < 0.05$) and a reduced average workload (55 vs. 62 TLX, $p < 0.1$) were also recorded.

In **Environment 2s (simulated open terrain)**, we scattered 40 green objects (see Figure 1). We set a malfunction schedule such that every robot encounters 2 malfunctions (on average) during the simulation. All robots started from the Assault deployment point (next to the Humvee). Operators were given 40 minutes to complete the task. Again, the results show an increased average number of detected objects by the robots (34.1 vs. 30.1 objects, $p < 0.001$) as well as their average covered terrain (1144 vs. 845 square meters, $p < 0.05$) for the condition in which the agent was present. Furthermore, a significant decrease in the average time that robots stay idle (1053 vs. 1455 seconds, $p < 0.01$) and a reduced average workload (57 vs. 61 TLX) were also recorded.

See Figure 5 for a graphical summary.

Overall, more than 95% of the advice was followed by the subjects. All but 4 of the 44 subjects showed an increased number of detected objects while equipped with our agent. Despite having the option, *none* of the subjects turned off the agent.

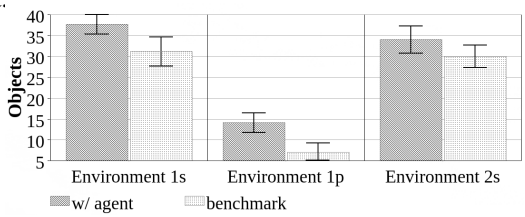

Figure 5: Detected objects across conditions and environments (error bars represent one standard error).

4 Conclusions and Future Work

In this work, we presented a new approach for the enhancement of operator performance in multi-robot environments. Our extensive empirical study, with 44 human subjects in Search And Rescue (SAR) environments, both in simulation and real deployment, shows that intelligent agents are able to significantly enhance the performance of operators in complex multi-robot environments, such as SAR.

Despite enduring high uncertainty and noisy signals, operators manage to take advantage of the agent's advice in our experiments more than 95% of the time and translate them into a better performance than their benchmark scores, suggesting that our agent gained the trust of the operators.

Surprisingly, even though the agent was trained in simulated environments, its effect in physical deployment was significantly higher than in simulated tests.

We conclude that the use of automated advising agents in robotics, and especially in multi-robot environments, is essential in bringing about better performance in real world applications and enabling operators to control a larger number of robots simultaneously. Our methodology can accommodate future advancements in robotics hardware and algorithms, and is not restricted to a certain type or quantity of robots in the environment.

We intend to expand this methodology and use the insights provided in this study to design and implement repeated-interaction agents. These agents could learn from previous tasks by the operator, in different terrains and environments, and tailor an advising policy for her. As part of this work we will examine the operator modeling for multiple interactions and the ability to deduce insights from one environment to another. Proceeding on a different path, we would also like to explore how our agent could be adapted to help operators in unknown and changing environments which require mapping and map adaptation.

References

[Azaria *et al.*, 2015] Amos Azaria, Zinovi Rabinovich, Sarit Kraus, Claudia V Goldman, and Yaakov Gal. Strategic advice provision in repeated human-agent interactions. *Journal of Autonomous Agents and Multi-Agent Systems*, 2015.

[Besl and McKay, 1992] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606, 1992.

[Casper and Murphy, 2003] Jennifer Casper and Robin R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics*, 33(3):367–385, 2003.

[Chen and Terrence, 2009] JYC Chen and PI Terrence. Effects of imperfect automation and individual differences on concurrent performance of military and robotics tasks in a simulated multitasking environment. *Ergonomics*, 52(8):907–920, 2009.

[Chen *et al.*, 2011] Jessie YC Chen, Michael J Barnes, and Caitlin Kenny. Effects of unreliable automation and individual differences on supervisory control of multiple ground robots. In *Proc. of HRI*, pages 371–378. ACM, 2011.

[Chien *et al.*, 2013] Shih-Yi Chien, Michael Lewis, Siddharth Mehrotra, and Katia Sycara. Imperfect automation in scheduling operator attention on control of multi-robots. In *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, volume 57, pages 1169–1173. SAGE Publications, 2013.

[Cummings *et al.*, 2007] ML Cummings, S Bruni, S Mercier, and PJ Mitchell. Automation architecture for single operator, multiple UAV command and control. Technical report, DTIC Document, 2007.

[Elmalech *et al.*, 2015] Avshalom Elmalech, David Sarne, Avi Rosenfeld, and Eden Shalom Erez. When suboptimal rules. In *AAAI*, 2015.

[Hart and Staveland, 1988] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988.

[Korsah *et al.*, 2013] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[Kulkarni and Pompili, 2010] Indraneel S Kulkarni and Dario Pompili. Task allocation for networked autonomous underwater vehicles in critical missions. *Selected Areas in Communications*, 28(5):716–727, 2010.

[Lewis, 2013] Michael Lewis. Human interaction with multiple remote robots. *Reviews of Human Factors and Ergonomics*, 9(1):131–174, 2013.

[Liu and Nejat, 2013] Yugang Liu and Goldie Nejat. Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent & Robotic Systems*, 72(2):147–165, 2013.

[Miller, 2004] C Miller. Modeling human workload limitations on multiple UAV control. In *Proc. of the Human Factors and Ergonomics Society 47th Annual Meeting*, pages 526–527, 2004.

[Morris *et al.*, 2015] Robert Morris, Ernest Cross, Jerry Franke, Mai Lee Chang, Waqar Malik, Garrett Heman, Kerry McGuire, and Robert Garrett. Self-driving aircraft towing vehicles: A preliminary report. In *Artificial Intelligence for Transportation Workshop (WAIT)*, 2015.

[Panganiban, 2013] April Rose Panganiban. *Task load and evaluative stress in a multiple UAV control simulation: The protective effect of executive functioning ability*. PhD thesis, University of Cincinnati, 2013.

[Parker and Zhang, 2002] Christopher Parker and Hong Zhang. Robot collective construction by blind bulldozing. In *IEEE Conference on Systems, Cybernetics and Man*, pages 177–195, 2002.

[Rosenfeld and Kraus, 2015] Ariel Rosenfeld and Sarit Kraus. Providing arguments in discussions based on the prediction of human argumentative behavior. In *Proc. of AAAI*, pages 1320–1327, 2015.

[Rosenfeld *et al.*, 2015] Ariel Rosenfeld, Amos Azaria, Sarit Kraus, Claudia V Goldman, and Omer Tsimhoni. Adaptive advice in automobile climate control systems. In *Proc. of AAMAS*, 2015.

[Rosenthal and Veloso, 2010] Stephanie Rosenthal and Manuela Veloso. Using symbiotic relationships with humans to help robots overcome limitations. In *Workshop for Collaborative Human/AI Control for Interactive Experiences*, 2010.

[Saez-Pons *et al.*, 2010] Joan Saez-Pons, Lyuba Alboul, Jacques Penders, and Leo Nomdedeu. Multi-robot team formation control in the guardians project. *Industrial Robot: An International Journal*, 37(4):372–383, 2010.

[Squire and Parasuraman, 2010] PN Squire and R Parasuraman. Effects of automation and task load on task switching during human supervision of multiple semi-autonomous robots in a dynamic environment. *Ergonomics*, 53(8):951–961, 2010.

[Velagapudi and Scerri, 2009] Prasanna Velagapudi and Paul Scerri. Scaling human-robot systems. In *CHI*, 2009.

[Wang *et al.*, 2009] Huadong Wang, Michael Lewis, Prasanna Velagapudi, Paul Scerri, and Katia Sycara. How search and its subtasks scale in n robots. In *Proc. of HRI*, pages 141–148, 2009.

[White III and White, 1989] Chelsea C White III and Douglas J White. Markov decision processes. *European Journal of Operational Research*, 39(1):1–16, 1989.

[Zheng *et al.*, 2010] Xiaoming Zheng, Sven Koenig, David Kempe, and Sonal Jain. Multirobot forest coverage for weighted and unweighted terrain. *IEEE Transactions on Robotics*, 26(6):1018–1031, 2010.