Agent-Based Coordination of Human-Multirobot Teams in Complex Environments

Alan Carlin[1,2], Jeanine Ayers[1], Jeff Rousseau[1], Nathan Schurr[1]

[1]Aptima Inc., 12 Gill Street, Suite 1400, Woburn, MA 01801

[2]University of Massachusetts, 140 Governors Drive, Amherst, MA 01003-9264

ABSTRACT

Room clearing, in which building surveillance is conducted to search for criminals, continues to be a dangerous and difficult problem in urban settings, for both the military as well as for police. In a typical setting, an unknown number of hostile forces may be located in a building, and they may be armed. Furthermore, there may be innocent civilians. The goal of the friendly units is to enter the room and secure it, but without loss of life of friendly forces, hostile forces, and most especially of innocent civilians. It would be beneficial to allow robots to be a part of the friendly team, however it is very challenging to have robots that do not either slow down or obstruct their human teammate. This is especially difficult since nearly all robots in use by the military and police today are tele-operated. In this paper, we describe work we have developed in cooperation with the army, for the room clearing domain. We constructed an algorithm whereby multiple agents, in the form of robots, can accomplish a room clearing task. We augmented the agent algorithms to introduce Adjustable Autonomy, allowing cooperation with humans. We describe simulated results of the algorithm on building maps, and furthermore we describe how we intend to next conduct hardware tests, and eventual plans to field the system. This agent-based solution has great potential to increase the acceptance and leverage of robotics in complex environments.

Categories and Subject Descriptors

I.2.9 [**Robotics**]: Commercial Robots and Applications

General Terms

Algorithms, Human Factors

Keywords

Adjustable Autonomy, Markov Decision Process, MMDP, Room Clearing

1. INTRODUCTION

Room clearing continues to be a difficult problem in urban settings, for both the military as well as for police [Department of Army 1979]. In a typical setting, an unknown number of hostile forces may be located in a building, and they may be armed. Furthermore, there may be innocent civilians. The goal of the friendly units is to enter the room and secure it, but without loss of life of friendly forces, hostile forces, and most especially innocent civilians.

At present, room clearing tasks are performed by human teams only. As part of a research effort called CHAMP (Coordinating with Humans by Adjustable-autonomy for Multirobot Pursuit), Aptima has constructed and simulated a method of coordinating mixed teams, consisting of humans and robots, in order to perform room clearing tasks. There are several challenges to this task. First, the robots and the robot sensors need to be chosen. Which sensors are chosen (laser, IR, RF) determines the capabilities of the robots. Second, an exploration algorithm for a multirobot team needs to be established, which guarantees that the room is explored. Third, human and robot capabilities need to be evaluated, and it needs to be determined how to best use the robots in cooperation with humans for room clearing.

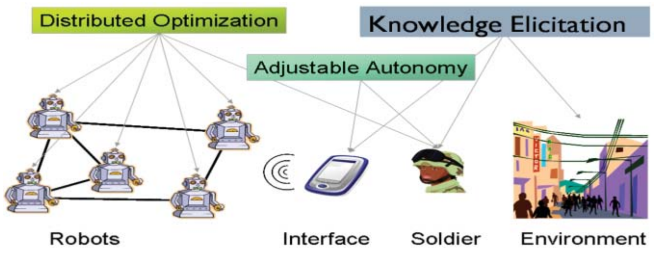


Figure 1 CHAMP architecture

Due to the recent work of researchers in the field of robotics and artificial intelligence, there are promising opportunities for the incorporation of semi-autonomous robotic entities within the small-unit tactical team that may positively impact the day to day effectiveness of the team. The future of Stability, Security, Transition, and Reconstruction (SSTR), Fixed Site Security (FSS), Cordon & Search (C&S), and Close Quarter Combat (CQC) include a limited number of human ground forces augmented by a team of robots capable of autonomous and coordinated action. To achieve this vision, Aptima along with Imprimis, Inc., acting as a Subject Matter Expert (SME), have developed a prototype system for Coordinating with Humans by Adjustable-autonomy for Multirobot Pursuit.

Figure 1 above shows the major technologies that were used in developing the CHAMP prototype. The robot teams are controlled by a distributed optimization algorithm, for the distributed framework we selected a Multiagent Markov Decision Process (MMDP). Adjustable Autonomy allows the degree of automated control between the human and robot to be dynamic. The environment was developed using knowledge elicitation sessions with subject matter experts (SMEs) and guided the concept of

Cite as: Agent-Based Coordination of Human Robot Teams in Complex Environments, Alan Carlin, Jeanine Ayers, Jeff Rousseau, Nathan Schurr *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1747-1754

operations for the prototype. The goal for the first phase of the project, reported on in this paper, was to simulate simple scenarios, and scale in complexity in later phases. Thus, we assume a map of the building is known to the robots.

The CHAMP approach created a simulated team of robots designed to traverse and cover all necessary terrains of the indoor environment while collaborating and communicating with each other as well as a human-in-the-loop. In addition, CHAMP allowed for command and status updates from/to human users, to allow both an understanding of the multirobot team's actions and a mechanism for adjusting them. The CHAMP approach leveraged Adjustable Autonomy in order to allow the coordination of the robotic team to *become more dynamic and adjust to the goals, demands, and constraints of the current situation* as it unfolds. Consequently, CHAMP introduces distributed control strategies that are not static and singular, but rather are represented by control policies that vary over time and circumstances. By casting the human-robot interaction problem as a problem of adjusting autonomy between human-automation links, we are able to have the robot team choose how to explore the map or allow the human to direct exploration. In determining the coordination plan, the robots reason about unexplored areas and patrol known areas while keeping the other teammates (human or robot) in sight.

In this paper, we describe the details of the CHAMP approach. This involves the construction of the MMDP and the implementation of Adjustable Autonomy. It also includes the construction of a simulation environment for the algorithms using the Player/Stage robot simulation tool, the construction and parsing of maps to run the simulation, and the evaluation of the results of our algorithms based on the maps. Finally, we discuss hardware that we have acquired and instrumented, and a plan for commercialization of the eventual resulting product.

Before continuing, we want to first identify some aspects of CHAMP's industrial relevance, as the insights we gained from customer calls may be of interest to the community:

- *Importance*: The room clearing task is considered high priority for the military as well as for police forces. Furthermore, the cooperative human-robot team algorithms can be expanded to a range of diverse domains involving humans and robots, including rescue scenarios and exploration scenarios.
- *Rationale:* An agent-based approach to controlling robots is a natural fit. Furthermore, we can integrate research from the agent literature, including Multi-agent Markov Decision Processes, literature on Adjustable Autonomy, and pursuit-evasion literature.
- *Barriers:* Customer enthusiasm is certainly not a barrier. As our customer told us in our first customer call, "If you build something we'll field it tomorrow". Rather, it's the sheer complexity of the problem. This is why we think it wise to start with simpler models containing more centralization and more certainty, then to expand to more uncertain models.
- *Financial:* Later in the paper we will provide an overview of specific market estimates. We reported on this information to the army. Although it is difficult to provide exact figures for specific markets, we anticipate a large amount of interest provided we demonstrate feasibility in hardware (interest for simulation results is currently high, but is more subject to change). The key challenge was to obtain hardware within development budgets, but which can also demonstrate agent algorithms. Fortunately, options are increasing in quantity, quality, and affordability for both robots and robot sensors. We feel that the Create platform from iRobot provides a nice tradeoff of features for expense.

2. RELATED WORK

Our problem is similar to the "pursuit-evasion" problem in the literature [Parsons, 1976]. In this problem, an intruder hides and must be found by a pursuer. The problem can be formulated as a graph, where pursuers and evaders are located at vertices and move along the edges at each step, and there is zero visibility . It can also be formulated as a search through a polygon region [Suzuki & Yamashita, 1992].

In the graph-search case, where adjacent nodes are visible, Isler et al show that the intruder can be caught by multiple pursuers with high probability (Isler et al, 2004). In recent work, Borie et al build upon prior solutions where all edge widths are assumed to be one, and generalize the results to graphs with unit-width arbitrary length [Borie, 2009]. Additional graph-theoretic approaches are numerous, including [Bienstock et al, 1991] and [Lapaugh, 1993]. The search of a polygon region has inspired approaches in the field of robotics. Some work is concerned with robots with a detection beam, monitoring an exit [Gerkey et al, 2004], [Lee et al, 2000].

The work most similar to ours is Pellier and Fiorino, in which the authors define an approach in a simulated 2D polygon environment where robots are assumed to have omni-directional sensors exploring an unknown environment [Pellier & Fiorino, 2005]. The authors construct an algorithm, using a construct called *critical points* , whereby a robot team explores the area, and robots are added to the team until it can be assured that any evader is found. In contrast to this work, we assume a small number of robots on our team, as will be shown in the next section.

3. PROBLEM ENVIRONMENT

The full problem is somewhat difficult for a graph-theoretic technique. If one were to represent the complete problem as a graph, each pixel would represent a graph vertex. Furthermore, visibility would not be limited to adjacent graph nodes, rather, it would be determined by the location of the obstacles. Thus visibility rules would differ for each map. The resulting graph problem is intractable.

In CHAMP, we find a meaningful subgraph that captures the important elements of the full map. We do this by leveraging critical points. We begin with an empty graph, G=(V,E). The map is examined for vertices (intersections of lines on the graph). A vertex is determined to be critical if and only if the angle formed by its adjacent edges is greater than π. Each of these critical vertices of the map are added to the graph G. It can be proved that all reachable areas on the map can be viewed from some critical vertex.

Once all the vertices are added to G, we examine them. For each critical vertex, we examine which other critical vertices are reachable in a straight line, without encountering a barrier. For each vertex u, for each vertex v that is reachable from it, we add edge (u,v) to G.

4. MMDP FRAMEWORK

We solve the problem described above by using the graph to form an MMDP [Boutilier, 1996]. Below we describe both an MMDP and our implementation of it. We defined the MMDP is as tuple **<Ag,S,A,P, R,T>** where:

- **Ag** is a finite set of agents indexed 1..*n*
- **S = <S1,S2,…>** is the state space. A joint state is a finite set of states, one state for each agent. That is, agent 1's state is in S1, agent 2's state is in S2, etc. The state space S is specified by all combinations of the individual robot states. For this project, we consider a robot's state its location. In order to limit the size of the state space, only the "critical points" on the map were considered. Thus a state such as <1,3,5> would mean that the first robot is at location 1, the second robot is at location 3, and the third robot is at location 5. The benefit of the critical point framework is that every point on the map is visible from at least one of the critical points.
- **A = <A1,A2…>** is the action space. A joint action is a finite set of actions, one action for each agent. Thus, agent 1 takes an action a1, selected from the set of possible actions A1. Agent 2 takes action a2, etc. All actions are taken simultaneously. We considered robot movement to be the action. An example action is <MoveToLocation2, Stay, Stay>, stating that robot 1 moves to location 2 and the other two robots stay in place.
- **P = SxAxS** is the probability matrix governing state changes. For example, if the state is <1,3,5>, as described above, and the action is <MoveToLocation2, Stay, Stay>, the probability matrix might say that there is a 90% chance that robot 1 moves to location 2, and a 10% chance that it fails, while there is a 100% chance that robot 2 and robot 3 stay in the same place.
- **R = SxA -> $\mathscr{R}$** is the reward for being in joint state S and taking joint action A. Further description is below.
- **T** is the horizon of the problem.

An MMDP can be viewed as having the same transition rules as *n* separate MDPs, where *n* is the number of agents. Each agent has its own set of states, and its own defined transition function. Rewards, however, are joint, a joint reward is received for the state of all the agents. In order to enforce a desirable property, that only one robot be allowed to move at a time, we assigned an infinite negative reward for joint actions where more than one robot moves to a different location on the same step.

In an MMDP model, state is fully observable. Thus, each agent can fully observe its own state, as well as the states of the other agents (after reviewing our robots and this problem, we found it most realistic to assume full communication). We selected the MMDP model in part because MMDPs can be solved efficiently (MMDPs are P-complete), and because, if necessary as the CHAMP project expands in scope, an MMDP can easily be expanded into a richer model. If ubiquitous communication were not assumed, the resulting problem could be viewed as a Decentralized Markov Decision Process (Dec-MDP), that is, a multi-agent MDP where each agent can fully observe its own state. If partial rather than full observability of state were assumed, the problem can be represented as a Dececentralized Partially Observable Decision Process (Dec-POMDP). Thus, the MMDP gives us a problem that can easily be solved for our initial implementations, but is expandable into richer problems for future implementations.

In constructing the reward function, we had three broad goals in mind:

(**Goal 1**) We want to locate the evader, preferably quickly. Thus, at any given time, all other factors being equal, we prefer that the robots keep as much of the area as possible within sensor range.

(**Goal 2**) We want to keep the robots safe, and more importantly, we want to avoid a situation where a robot becomes disabled without knowing the reason. Therefore, at any given time, all other factors being constant, we prefer that the robots keep each other within sensor range.

(**Goal 3**) We want to use a framework that is scalable to future work; in particular we may want to account for uncertainty in robot perception (sensor error, etc) as well as the result of robot actions.

A reward function R was constructed so that behaviors that are most likely to find an evader on the map were rewarded. For our implementation, the reward was based on the current state, and we narrowed the broad goals above to reward the following specific behaviors:

1. The team should receive reward for keeping visible to each other.
2. The team should receive reward for keeping a large portion of the map visible at each point in time.
3. The team should receive penalty for each point unseen, as a function of how long it has been unseen.
4. The team should receive penalty for each point not visited, as a function of how long it has been unvisited.

As a result, the following reward function was constructed. The reward equation drives the planned movement of the team.

Reward = $\mathbf{p + 3q - \sum_v C^{t1(v)} - \sum_v D^{t2(v)}}$ where:

- p is the number of critical points visible
- q is the number of robots visible to each other
- t1(v) is the time since critical point v has been seen. (In the above we sum this over all the critical points).
- t2(v) is the time since critical point v has been visited. (In the above we sum this over all the critical points).

The equation has taken into account the possible scenarios when there are a lot of critical points or a lot of robots. The penalty for each unexplored critical point grows with time. Therefore, exploration dominates the equation and the robots will tend to split up. The term "q" offsets that behavior by rewarding the robots that remain in sight with each other. This trade-off gives us opportunities for adjusting the team's behavior given a mission objective or environmental constraint. The equation is also centered from the viewpoint of the team. It does not matter which robot explored an area last, as long as a member of the team has explored it. For our current effort, the robot team is assumed to be in perfect communication with one another. The theoretical effect is that the robots can then be logically considered to be one centralized team rather than independent teams.

There is a subtle difference in the last two terms, the difference between leaving a critical point *unseen* and leaving a critical point *unvisited*. The difference roughly corresponds to a difference in granularity. When critical points have been seen recently, this means that broad swaths of the map have been explored recently, so that a region has not gone unattended. Still, an individual corner, only visible from one critical point, can be undetected. However, as the penalty for leaving a critical point *unvisited* increases, this means that the agents will be increasingly incentivized to reach this point, and explore any hidden corners of the map.

Although we will see in the Results section that we initially considered static evaders, the formulation is designed to encourage detection of both static and dynamic evaders. Static evaders will be detected, because the penalty for unvisited points increases exponentially until they drive the reward function. Dynamic evaders will also tend to be detected, however, because the robots are encouraged to keep large amounts of the map visible at each time step.

5. ADJUSTABLE AUTONOMY

Our next objective was to augment our approach by investigating the application of adjustable autonomy to the human-multirobot team for the most effective execution of the mission of room clearing. The methods employed included: risk/benefit trade-offs, action durations, and expected utility [Schurr et. al 2009].

We have implemented adjustable autonomy, describing how humans can integrate with the robots. Our SME has noted that some missions are time critical, and speed is the utmost consideration, whereas other missions take place in a cordoned area and are not time critical. We use these different types of scenarios to motivate different behaviors on the part of the robots. In the latter case, we want use the additional time to be slow and methodical in the search, to especially avoid unsafe situations to the humans, and even to the robots, as time affords it.

We have implemented adjustable autonomy by allowing the human on the team to adjust the parameters p,q,C, and D of the Reward equation. The parameters incorporated into the algorithm correspond to the parameters in the equation in the above section, and are named and defined as follows.

Swarming Bonus: Rewards robots for being within sight of each other.

Visibility Reward: Rewards robots for seeing large parts of the map at one time. This is useful if targets can be moving, as the robots are vigilant to a large portion of the map.

Unseen Penalty: This induces the robots to explore the map, in broad swaths. Robots want to go to portions of the map that have not been seen recently.

Unvisited Penalty: As discussed in the previous section, the Unseen Penalty encourages exploration of broad swaths, the Unvisited Penalty encourages us to visit specific nodes. Since there may be some corners of the map that are only visible from one or two critical points, it is the Unvisited Penalty that drives exploration of the nooks and crannies of the map.

When swarming and visibility are large, the agents are encouraged to remain visible to each other. Swarming is encouraged over exploration. When unseen and unvisited terms are large, agents will tend to explore. These parameters will be configurable in real-time by the human. A screenshot of the possible adjustments can be seen in Figure 2 below. The slider bars allow for fine tune adjustment and are intended for system designers to create preset profiles for particular situations and particular users. The drop down box for presets is intended to allow the pursuer to quickly change the behavior of the team as desired.

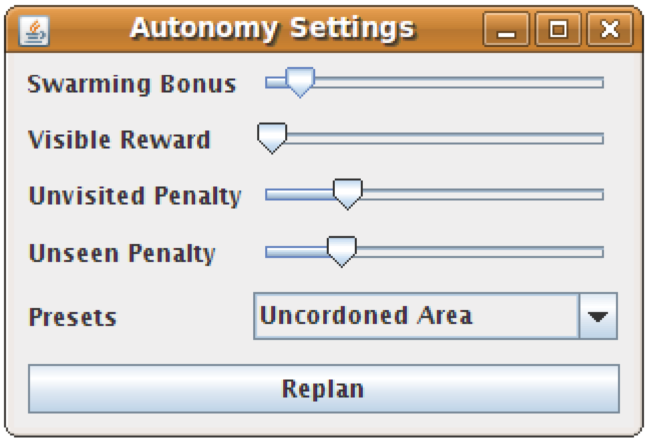


Figure 2

6. PROTOTYPE

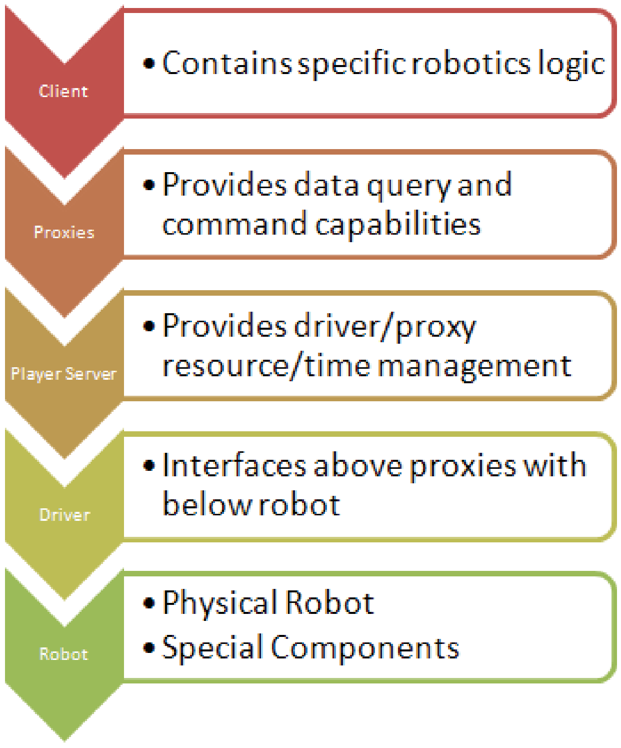
Our next objective was to build a prototype multirobot team simulation to prove the feasibility of our approach and to support future development and experimentation of CHAMP. The methods employed included: evaluation and selection of a simulation environment, a strategy for operating on multiple maps and processing critical points, the development of the coordination algorithm, and finally the coding of the robot agents.

The first thing that was done was the evaluation of several simulation environments and the selection of Player/Stage as our target for the phase I implementation platform. The environments that were evaluated include: USARSim, MS Robotics Studio, Robot Operation System (ROS), and Player/Stage. The team selected Player/Stage because it met all of our requirements and transfers nicely to real robotic hardware. A Player/Stage overview will now be discussed.

<u>Player/Stage System Architecture</u>

The Player/Stage system employs a client-server architecture which offers a level of abstraction for dealing with mobile robot and sensor hardware. The 'Player' software is a network server that provides an interface for communicating with a robot's sensors and actuators over a TCP/IP socket. In most situations the server runs directly on the robot.

By employing a proxy system for individual drivers, such as a laser rangefinder or bumper switch, a client program can access the proxy sensor's data (provided by a Player server) without needing to know what specific model or brand of hardware is being used under the hood. This allows a client program to be hardware/robot agnostic. Pieces of hardware or entire robots could be easily exchanged for others without having to rewrite any of the robot's control code, as long as any substitute robot is capable of exposing the same interfaces (e.g. it has a laser, bumper, or whatever is required for the algorithms). The 'Stage' software is a two-dimensional simulation environment capable of providing simulated drivers to a player server allowing users to test their algorithms without the need for real hardware.

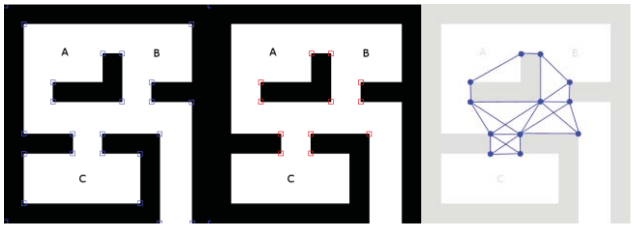

Client
• Contains specific robotics logic
Proxies
• Provides data query and command capabilities
Player Server
• Provides driver/proxy resource/time management
Driver
• Interfaces above proxies with below robot
Robot
• Physical Robot
• Special Components

Player/Stage

Unfortunately, 'Stage' can only run on Unix-based platforms. In an effort to circumvent installing another operating system on the workstations (which run Windows XP), a virtualized Linux environment was created that runs on top of our existing operating system.

The virtualized OS is Ubuntu 9.04, a modern, relatively easy to use Linux distribution. The virtualization software used was VMWare's Player (no associate with Player/Stage). VMWare's Player software is freely available at their website. Ubuntu includes community-supported software repositories that include version 2.0.4 of Player/Stage which we installed on our default virtual image.

Next, we developed a map processor that accepts a geographic layout in the Scalable Vector Graphics (SVG) format at the beginning of a mission and generates a series of vertices, solves for all the critical points, and then generates an adjacency graph from those points which becomes the input to the distributed coordination algorithm. The figure below explains our approach which is based on the already-described [Pellier, 2005]. Given a map, on start up, the program searches and identifies all vertices as identified in the figure on the left. The program removes all vertices that don't meet the criteria of the angle being greater than 180 degrees as identified in the middle figure. The final figure on the right is the constructed adjacency graph that informs the distributed coordination algorithm with the critical points in the area of interest.



This approach to generating critical points does not require a map or preprocessing. For the current work, the assumption is that a map is available prior to the execution of a room clearing scenario. But, in future work or in an operational setting a robot scout could identify critical points and use that information as input into the adjacency graph generator that then informs the distributed coordination algorithm.

Finally we developed a Player/Stage prototype that implements the distributed coordination algorithms while searching for a human evader with a given geographic area. The user interface contains several aspects. There is a high-level view of the map and the obstacles, humans, and robots traversing around map area. Each robot team member displays its path in its own separate window as it traverses the adjacency graph. The final component is the Adjustable Autonomy control which allows the human to control the behaviors of the robot team.

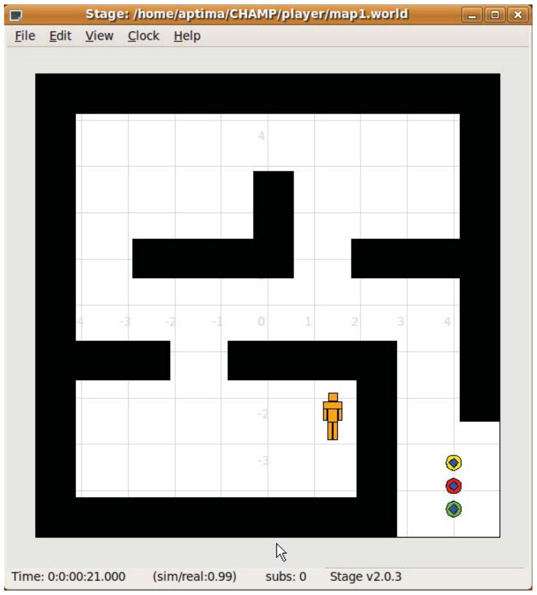
7. RESULTS

We present results that were gathered from the CHAMP simulation environment that includes: Player/Stage, distributed control algorithms and an adjustable autonomy interface. The results will be discussed in the following order: (1) a graphical representation of the map, (2) the challenges of the map from the perspective of a human combat team, (3) the challenges of the map for the CHAMP distributed coordination algorithm. (4) CHAMP behaviors as executed, (5) timing information, and (6) the future impact or recommended changes to augment the CHAMP approach.

On the last point, we note that from the point of view of the customer, this is an initial effort and that certain desirable room-clearing goals have never been quantified. Thus, analysis of simulated performance has served both to augment our own algorithms, as well as to help define goals for room clearing domain behaviors, to be quantified and elicited in future work.

7.1 Example Map 1

Example Map 1 was the first map that was developed to test and evaluate the CHAMP algorithms and prototype simulation. This layout is a particularly challenging space for a human team to clear. The top half of the map is more dangerous because of the open spaces and doorways. An evader can emerge, hide, ambush, or flee from multiple areas. The top half of the map would be the first area that a human team would explore and clear.


Stage: /home/aptima/CHAMP/player/map1.world
File Edit View Clock Help
Time: 0:0:00:21.000 (sim/real:0.99) subs: 0 Stage v2.0.3

Example Map 1

The CHAMP algorithm uses critical points to direct the robots on how to approach and explore the area. Critical points, because of their large angle, are the most exposed areas of the map. Therefore, the algorithm places the robots in the most danger as they clear this space. The observed behavior of the robotic team using the CHAMP system was to explore the top half of the map first, our subject matter expert informed us that this same behavior would be exhibited by a human team. The robots move one at a time, like a human team, although the robots move much slower than a corresponding human team. Table 1 highlights the differences in planning and search data given the adjustable autonomy strategies that were selected at startup of the given simulation run. As expected, the swarming behavior configuration, where the robots earn more reward for keeping one another in sight, increased search time. In the search scenario, the search time decreases as the robots are rewarded for searching broad areas and visiting specific nodes.

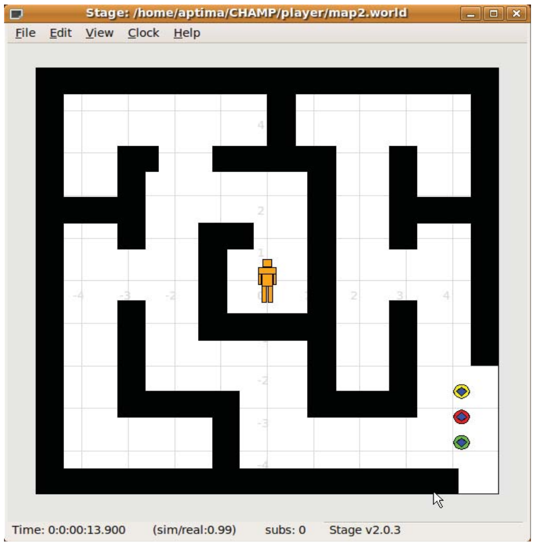
Table 1: Performance on Map #1

AA Strategy	Planning time (s)	Search time (s)	# Steps
Swarming	2.7	210.9	25
Search	2.6	144.7	14

An analysis of the robot paths suggests that in future work it may be valuable to include more than just the critical points of the map when planning a distributed coordination strategy. The critical points are the most exposed, and therefore the most dangerous, and it may be beneficial to direct the robots to complementary points that achieve the search and swarm objectives but place the robotic team members in less danger.

7.2 Example Map 2

Example Map 2 was the second map that was developed to test and evaluate the CHAMP algorithms and prototype simulation. This layout was designed to be "maze-like". There are about equal areas to the left and to the right of the entry point. The significant impact on the algorithm was that one robot continued to keep the right hand side of the map visible even after it was cleared and the human was not found.


Stage: /home/aptima/CHAMP/player/map2.world
File Edit View Clock Help
Time: 0:0:00:13.900 (sim/real:0.99) subs: 0 Stage v2.0.3

Example Map 2

This behavior demonstrated to the team that it may be valuable to change the weight of the critical points once they are determined to be insignificant to the mission. Once the area to the right has been searched, the robot should be free to move back to the entry point and allow a robot team member to progress further along the corridor to the left.

Table 2: Performance on Map #2

AA Strategy	Planning Time(s)	Search Time (s)	# steps
Swarming	13.3	249.5	32
Search	15.6	180.1	14

7.3 Example Map 3

Example Map 3 was the third map that was developed to test and evaluate the CHAMP algorithms and prototype simulation. This layout was designed as a hallway in a hotel with longer sight area and rooms on the left and the right to be cleared. The significant impact on the algorithm was that it was easier for the robots to explore more areas while still keeping their teammates in sight. The planning time significantly increased because of the size of the map and the number of critical points.

Table 3: Performance on Map #3

AA Strategy	Planning Time (s)	Search Time (s)	# Steps
Swarming	54.9	135.2	8
Search	42.8	191.3	14

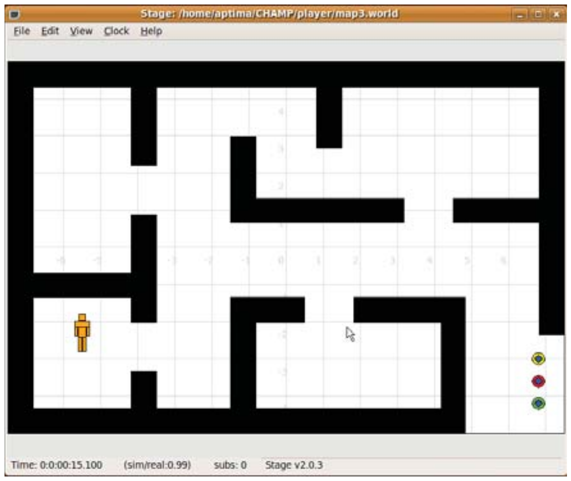

Stage: /home/aptima/CHAMP/player/map3.world
File Edit View Clock Help
Time: 0:0:00:15.100 (sim/real:0.99) subs: 0 Stage v2.0.3

Figure 3: Example Map 3

8. Hardware

We have obtained 3 iRobot Create programmable robots. The code for the algorithms described earlier in this paper can be directly ported to these robots. Before selecting sensors, we first performed an analysis as to sensor packages for the robots.

We evaluated a passive infrared motion sensor, a standard CCD/CMOS camera, an Infrared Camera, a Thermal Imaging Camera, and a laser rangefinder.

Table 4: Sensor Options

Device Name	**Notes**	**Cost**
PIR motion sensor	Senses moving warm bodies	< $100
CCD/CMOS Camera	Standard digital camera	$100+
Infrared Camera	Similar to above, but infrared	$200+
Thermal Imaging Camera	Can see through heat-permeable barriers	$1000
Laser Rangefinder	Uses a laser to scan at 10Hz, to construct a 2D or 3D map of environment	$5000+

The passive sensor is the least expensive, but requires the host vehicle to be stationary when scanning. Also stationary targets would be undetected. The digital and camera is also inexpensive, its disadvantage is it may require sophisticated processing algorithms. The Infrared (IR) Camera is also inexpensive, but due to its IR nature it does not detect color as well. The thermal imaging camera is interesting because it can detect warm bodies, but it is moderately expensive and requires vision processing algorithms as well. Finally, the laser rangefinder was the most expensive option we contemplated. It allows the robots to construct a map of its environment by sweeping its laser. The most obvious drawbacks are: (1) 3D scanning requires a pan/tilt armature. (2) It is challenging to map the environment if there are fast-moving objects.

As a result of the analysis, we instrumented our robots with a Sokuiki laser range finder. Angular resolution on the laser is .352 degrees, and power consumption is 2.5W. We have also equipped the robots with small digital (visual) cameras, and are implementing detection algorithms on those cameras.

For the operating environment, we built a small maze for these robots, to replicate in hardware the simulated problems from the results section. The maze is shown below.

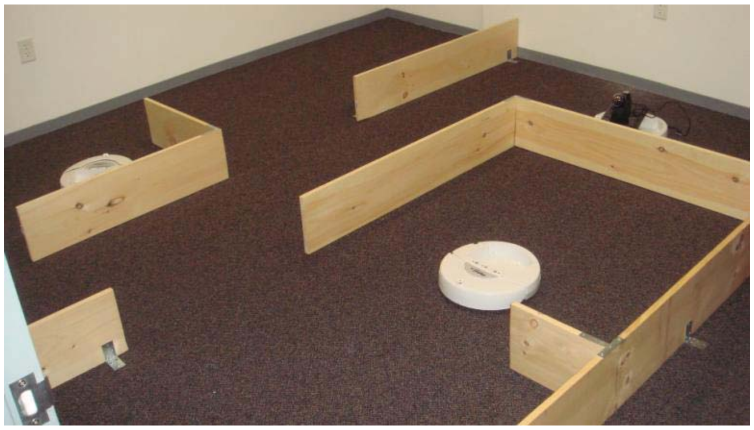


Figure 4: Maze in Aptima Robotics Laboratory

Results from this paper will be replicated in the maze.

9. Future Steps

We intend to push our future work in two directions. First, we intend to continue moving our operating environment from software to hardware. Second, we intend to augment our algorithms.

On the latter point, we intend to augment the following aspects of the algorithm:

1. Augment the Adjustable Autonomy and User Interface
2. Simulate uncertainty of observations
3. Simulate uncertainty of communication

For the first item, it is important to construct a friendly User Interface, because the intention is to field mixed teams of humans and robots. The human should be able to ascertain, through a PDA, the location of their robotic partners. Furthermore the humans should be aware of the robots' plans; the human should be able to anticipate each robot's next action, since they will be acting as a team. Similarly, a human user should be able to communicate her next actions to the robot team members in order to augment the plans, if so desired.

On (2), we note that the MMDP model can be easily expanded into a Decentralized POMDP model. The latter handles uncertainty by receiving an *observation* at every time step. The observation correlates to a state, with an attached probability.

Through observations, the agents must reason about their current state. It is likely that as our hardware implementation scales up, that the uncertainty of the robots will increase, and the richer Dec-POMDP model will prove useful.

Regarding (3), we will gradually scale up to environments where communication may not be instantaneous and ubiquitous. In the current operating environment, it is trivial for robots to simply communicate location with each other in order to form centralized plans. However, as we scale up, the robots will likely start making complex observations (in the form of image data) which may not be instantly communicable. Therefore we will investigate models that include communication.

At present, our work is sponsored through an SBIR Phase I grant, with an invitation to apply for Phase II. As the work continues, there are several possible commercial opportunities. They can roughly be divided into DoD and non-DoD markets. For the DoD, possible markets include Army Future Combat Systems and the Navy Combat Information Center. The former would be interested in applications similar to the examples in this paper, and the latter may be interested in adapting the techniques for automated vehicles. We estimate the market size for resulting products to be $15-50M/year for the Army and Navy.

For non-DoD, we have identified three potential markets. The first is Emergency Response and Law Enforcement, in scenarios similar to the examples. The second non-DoD market is Hazardous Environments, where such systems could be used to control multiple devices to reach dangerous places, such as underwater locations or for oil exploration. The benefit of the product would be improved safety for the user. The third possible market is for health care systems, for tele-medicine or remote control of medical instruments.

10. Conclusion

In the CHAMP project, we addressed the problem of room clearing with a mixed team of humans and robots. We developed code to pre-process a map and identify critical points on the map. We developed and coded an MMDP coordination algorithm for the team. We augmented the algorithm with mechanisms for Adjustable Autonomy. Then we developed a Player/Stage simulation prototype in which the algorithms could be tested, and we analyzed the results. We have reported these results to the customer, and expect to continue this line of research in both CHAMP and in other projects.

11. References

[1] Bienstock, D., and Seymour, P. Monotonicity in graph searching. Journal of Algorithms, 12, 239-245 (1991).

[2] Borie, R., Tovey, C., and Koenig, S. Algorithms and Complexity Results for Pursuit-Evasion Problems. Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI) (2009).

[3] Boutilier, C. Planning, learning and coordination in multiagent decision processes. In Proceedings of the Conference on Theoretical Aspects of Rationality and Knowledge, 195-210 (1996).

[4] Gerkey, B., Thrun, S., and Gordon, G. Visibility-based pursuit-evasion with limited field of view. In Proceedings of the National Conference on Artificial Intelligence (AAAI) (2004).

[5] Isler, V., Kannan, S., and Khanna, S. Randomized Pursuit-Evasion with Local Visibility. SIAM Journal on Discrete Mathematics, 20, 26-41 (2006).

[6] LaPaugh, A. Recontamination does not help to search a graph. Journal of the ACM, 40, 224-245 (1993).

[7] Lee, J., Park, S., and Chwa, K. Searching a polygonal room with a door by 1-searcher. International Journal of Computational Geometry and Applications, 10, 201-220 (2000).

[8] Military Operations on Urbanized Terrain (MOUT). FM 90-10. Headquarters, Department of Army. Washington, DC, 15 (August 1979).

[9] Parsons, T. Pursuit-evasion in a graph. Theory of Applications of Graphs, Lecture Notes in Mathematics, 426-441 (1976).

[10] Pellier, D., and Fiorino, H. Coordinated exploration of unknown labyrinthine environments applied to the Pursuit-Evasion Problem. Proceedings of 4th International Conference on Autonomous Agents and Multiagent Systems (2005).

[11] Schurr, N., Marecki, J., and Tamble, M. Improving Adjustable Autonomy for Time-Critical Domains. Proceeding of 8th International Conference on Autonomous Agents and Multiagent Systems (2009).

[12] Suzuki, I., and Yamashita, M. Searching for a Mobile Intruder in a Polygonal Region. SIAM J. Comp., 21, 863 (1992).