A Mixed Initiative Controller and Testbed for Human Robot Teams in Tactical Operations

Vikram Manikonda, Priya Ranjan and Zachary Kulis

Intelligent Automation Incorporated
15400 Calhoun Drive, Suite 400
Rockville MD 20855
{vikram, pranjan, zkulis}@i-a-i.com

Abstract

Output

Abstract

This paper discusses a mixed initiative controller and testbed for Human Robot Teams (HRT) in tactical operations. We focus on a tactical operation that involves large teams of robots collaborating with one or more human agents, working collectively to achieve a common tactical goal. Our approach to HRT builds on earlier work in Mixed Initiative Control (MIC) [12][18][22]. The proposed proactive-MI approach allows the robots to assume or relinquish roles depending on their own capabilities and understanding of the environment. A key contribution to this research effort, and the focus of this paper, is the development of a HRT planning-execution framework and hardware-testbed to demonstrate HRT in tactical operations. The framework is built using software agents and Motion Description Languages (MDLe) [8]. The agent architecture provides the framework for collaboration, and communication, between the human agents and the robot agents. The framework is also designed to enable easy addition of both low-level robot "drivers" and high-level control algorithms/plans as are needed for a particular tactical operation. Metrics that are relevant to HRT in this particular domain are also presented.

Introduction

Over the last few years there has been significant research in various aspects of Human Robot Teams. In [1][24] research has focused on aspects of Human-Robot Interaction, where robots are performing tasks on behalf of humans, typically in teleoperation type applications. Examples of these include robots performing extra vehicular activities in space, robot welding and control of unmanned air vehicles. In these applications, robots are typically subordinates to the human controllers. While in some settings the human robot ratio is one-to-one, in other application such as mars mission or control the Global Hawk UAV the human-robot ratio is many to one. In [4][5] aspects of human robot teaming are considered in applications where humans and robots work independently with loose coordination and humans are in the loop only when needed and appropriate. In [11][23] human-robot interaction in a social domain is considered. Here socio-cognitive skills needed to support human robot collaboration are explored. More recently the Urban Search And Rescue (USAR) domain has led to research into mixed initiative control approached for human robot teams [12][16][18][22]. Human-robot team solution in this domains explore the human-robot interactions as a tradeoff between supervised autonomy (where the operator is a part of the loop) to full autonomy.

In this paper we explore a class of problems where multiple robots collaborate with one or more humans and work on a highly constrained tactical maneuver. The specific tactical scenario considered here is one of neutralizing a potential threat (e.g. intruder or IED) in an urban environment. The HRT team consists of robots and humans (soldiers) where, due to human safety reasons, numbers of robots are much greater than the humans. The goal of the HRT in this operational scenario is to locate the

IED/intruder, and diffuse/neutralize it while identifying and eliminating (if possible) any potential threats that may exists. We further assume that the IED/intruder is located in an area such as building that has multiple entries and exits, each of which is a potential source of threat, and other human agents, friend and foe, can enter or exit through these areas. The robots and human have varying levels of autonomy and intelligence. The robots are equipped with sensors and effectors enabling them to play the roles of "surveillance" and "IED diffusion" agents. Humans are capable of surveillance and threat neutralization. Surveillance is performed autonomously by robots using video cameras to cover a particular area and relay potential threat information to the humans. Only humans are capable of distinguishing between "friend" and "foe". Both human and robots share sensor and intent information providing common situational awareness. The robot teams work with humans as one cohesive unit. The robots are completely autonomous and are not remote/teleoperated. The robots and humans share a common goal, and collaborate to generate and execute a plan. The nature of the task and the robot capabilities typically result in interdependent tasks betweens humans and robots.

To address some of these challenges, in this paper we present a Mixed Initiative Control (MIC) approach and a HRT planning-execution framework and hardware-testbed to demonstrate HRT in tactical operations. Our MIC efforts are motivated in part earlier work in [12][18][22]. We demonstrate how the robot agents, depending on the environment and team goal, take the initiative to assume appropriate roles (e.g. surveillance, attack, diffuser) to support the mission. While the initial tasking may be initiated by a human, the proposed proactive-MI approach allows the participating agents to assume or relinquish roles depending on their own capabilities and understanding of the environment. A key contribution to this research effort, and the focus of this paper, is the development of a new HRT planning-execution testbed. An integral part of the HRT testbed is the Distributed Control Framework (DCF) that enables the deployment and execution of plan on the physical robots. The framework and testbed provide an environment to demonstrate and validate our mixed initiative plan framework in real-hardware.

This paper is organized as follows - We first describe the mixed initiative planning control architecture implemented in this effort. We then discuss the HRT planning and execution framework developed for this effort. This is the primary focus of this paper. Metrics for HRT in this domain are identified in the following section. The final section presents the HRT hardware testbed developed for this effort and outlines the proposed experimentation plan to validate the MIC and HRT planning and execution framework.

Mixed Initiative Control Architecture

The MIC paradigm provides a good framework for modeling human-robot teams in a tactical operation. While each agent can operate independently, the mixed initiative control problem lies in allowing the agent who knows the best to process and coordinate the activities of the other team members. Related work in the is area includes [12][16][18][22].

Figure 1 shows the overall MIC architecture adopted in this effort. The choice of the architecture is similar to that of [18] and [22]. The control architecture developed here attempts to address and resolve the following MIC issues

- When should a particular agent take initiative?
- How is initiative assumed?

The lowest layer of the control architecture is the DCF layer that manages the execution of the plans for the physical robots. Robot state estimation, control algorithms, and reactive behaviors such as obstacle avoidance are implemented by this layer. The highest layer is the human-robot team planner that generates plans at an abstract level. A multi-agent-based approach is adopted, where planning is done in a distributed fashion and coalition among partners, if needed, is facilitated using a plan mediation agent. In our architecture we assume that the final authority lies with the human agent. It is our intent to use Extended Hierarchical Task Networks (EHTN) to represent the goals contingencies and uncertainties associated with the plan. This is still under implementation and is not discussed in this paper. Specific details of using EHTN for multi-agent planning can be found in [19] and references therein. The intermediate layer is a translation layer that maps EHTN plans into partial plans into Motion Description Language (MDLe) scripts that can be interpreted and executed by the DCF.
In our approach we assume that initial plan generation is performed by the human. The plan is decomposed into a set of partial plans for each member of the team with associated interdependencies. The execution of the partial plans is done by each robot under the DCF as discussed in the section below.

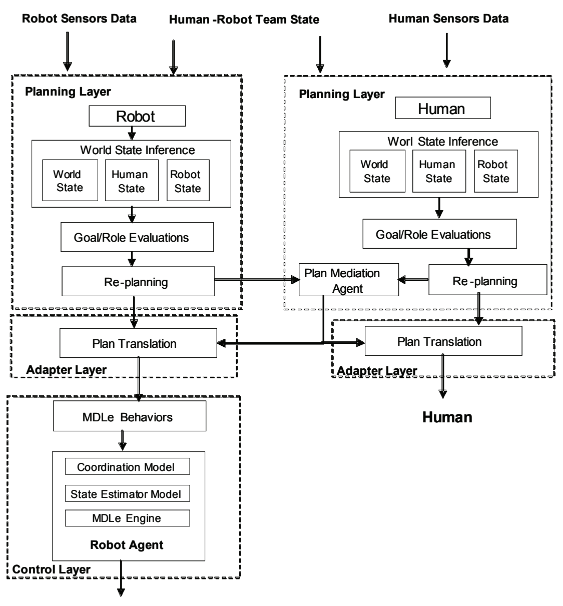


Figure 1: Human-robot MIC architecture.

Once this plan is being executed the human and robot agents "sense" changes to the world using their sensors and broadcast this information to the rest of the team. Each human/robot agent in our framework maintains a model of the world state, the state of the robot team and the state of the robot/humans they are supporting. While several architectures implement "cognitive" models [14][21] of the human agent, for this implementation we will adopt a fairly simple probabilistic model. The agents use on-board-sensor information, coupled with information received from other members of the team to update their internal representation of the world, robot agents and human agents, and predict the human agent intent. Based on these state updates and predictions, the robots take the initiative to dynamically modify their goals and associated roles. Depending on the level of autonomy and authority of a particular robot, these new goals are mediated with other agents and may require approval from the human. Once the goals are approved, re-planning occurs to generate a new set of partial plans, that are then executed by the DCF.

HRT Planning and Execution Framework

The key elements of HRT planning and execution framework include the DCF, the implementation of robots control using motion description languages, and the DCF command and control interface. Each of these are discussed in the sections below.

Distributed Control Framework

A core component of this research is the development of a Distributed Control Framework (DCF) to enable distributed planning and execution of plans for the human robot teams. Initial research efforts in this direction include [4][15][17]. The DCF architecture is built using software agents and Motion Description Languages (MDLe) [8][9]. The agent architecture provided the framework for collaboration, and communication, between the human and the robot agents. While humans are capable of executing plans or tasks provided to them, execution of plans by robots is more complex. A plan needs to be translated into a set of control laws that are derived based on the dynamics and sensors available on the robots. In DCF, the MDLe engine translates HRT plans into commands that can be executed by the robots. One advantage of an MDLe-based implementation is that robot mission plans and control behavior can be represented at a level of abstraction that does not depend on the particular robot. The connection to the particular robot is made during execution/simulation time when the appropriate robot dynamics or drivers are loaded by the MDLe engine.

DCF is written entirely in the Java programming language, and is a small and lightweight framework that may be deployed on any computing architecture that supports the Java Virtual Machine (JVM). DCF is built on top of the CybelePro® agent-based framework [6][7] that provides the runtime environment for control and execution of agents and enforces the agent-centric computing model. Since all interactions between agents are via messaging[1], robot agents in this framework do not distinguish between software and "physical" robot entities, providing the ability to combine virtual and real agents in an experiment.

In our implementation (see Figure 2) a robot is modeled as an agent that is composed of a Robot Planner, the State Estimator, and the Robot Engine (the robot planner, the state estimator, and the robot engine are implemented as Cybele Activities). The Robot Planner is responsible for high-level mission planning and maintaining the state of the robot team. In coordination with the other agents, the Robot Planner has the ability to change the currently active mission on-the-fly. In addition, the Robot Planner aggregates and stores state information received from remote agents for further processing by an appropriate Coordination Model. The Coordination Model is a user-defined module encapsulated by the Robot Planner activity that interprets the collected peer state data. For example, a Coordination Model might be developed to determine the two closest peers on either side of the agent. The State Estimator processes sensor data collected from hardware sensors mounted on the agent. Since the State Estimator supports hardware-driven events, the processing of new data occurs with minimal latency. The State Estimator relies upon a user-defined Estimation Model to process the received sensor data. The complexity of the Estimation Model is entirely application dependent and can vary from simple low pass filtering to more complex algorithms including Kalman and nonlinear filtering. The Robot Engine parses control plans and executes distributed control algorithms on the agent.

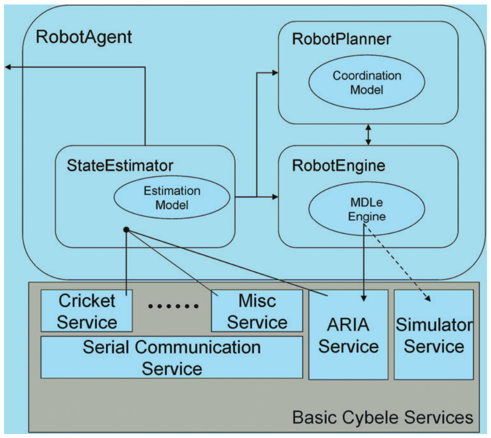


Figure 2: Robot Agent Model

Motion description language –based implementation of controllers and mission plans

Currently, the Robot Engine handles the parsing and execution of control/mission plans written in the Motion De-

[1] Under the paradigm of software agents, agents are autonomous and event driven entities, that do not share their encapsulated data, and interact via messaging

scription Language Enhanced (MDLe) [8][9]. In this section we provide a brief discussion of MDLe and present our implementation of MLDe under the DCF framework.
MDLe is a high-level programming (scripting) language used to specify hybrid control laws in terms of atoms, actions, and interrupts. In MDLe, at the lowest level, the system is modeled by a so-called kinetic state machine depicted in Fig. 3 . The robot is governed by a differential equation of the form

$$\dot{x} = f(x) + G(x)U;\ y = h(x)$$

$$where\ x(\bullet): R^{+} \to R^{n},\ U(\bullet,\bullet): R^{+} \times R^{n} \to R^{m}$$

and G is a matrix whose columns g_i are vector fields in R^n. The robot has access to a set of timers T_i and can evaluate a set of boolean functions $\xi_i : R^p \to \{0,1\}$ defined on the space of sensor outputs R^p. An input $U(x,t)$ is to be thought of as a general feedback law which can be suspended by the interrupt functions ξ_i or timers T_i.

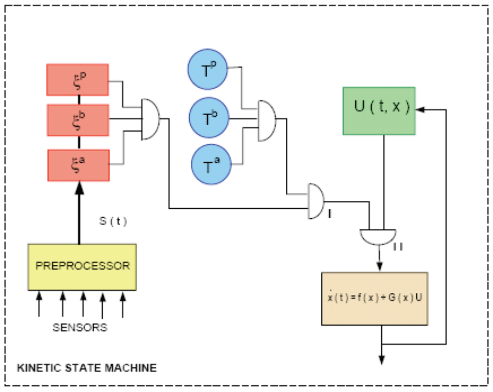


Figure 3: Kinetic State Machine

The building blocks of an MDLe script are atoms. In MDLe an atom is defined by the triple σ = (U, ξ, T) where $U(x,t)$, ξ_i are as defined above. An atom is executed by the MDLe engine by applying the input $U(x,t)$ to the kinetic state machine until the interrupt function ξ is triggered or until T units of time, as defined by the timer elapse, whichever occurs first. T is allowed to be ∞ and U may be an open loop or feedback control law. In MDLe a string of atoms can be used to capture certain behaviors (e.g., obstacle avoidance, surveillance). For example, one could use the atoms σ1 = (U1, ξ1, T1), σ2 = (U2, ξ2, T2) to define the behavior π = ((σ1, σ2), ξb, Tb). In MDLe interrupts associated with atoms are called level 0 interrupts while, interrupts associated with Behaviors are called Level 1 interrupts. Given a kinetic state machine and a world model a plan γ is defined as an ordered sequence of behaviors which when executed achieves the given goal. For example a plan γ could be generated from a given language where each behavior is executed in the order in which they appear in the plan. Although MDLe strings are sequential, the order of execution of atoms in a plan does not have to coincide with their order of appearance in that plan. This is a consequence of allowing the interrupt logic in the behaviors to allow for a transition to a user determined atom/behavior. We refer the reader to [8][9] for more details on MDLe

MDLe Implementation in DCF

The MDLe Engine, implemented in the DCF consists of two core components: a MDLe Plan Executor and a Planner. Both components are written in Java and may be deployed on any computing platform supporting the Java Virtual Machine. Together, these two components give users the ability to dynamically create and execute MDLe plans on diverse robotic platforms.

The MDLe Plan Executor handles the low-level details of checking for fired interrupts, maintaining timers, and switching between the constituent Atoms and Behaviors comprising an MDLe plan. The Plan Executor features an MDLe compiler (developed using the SableCC framework [27]) which provides full support for the MDLe language, including scaled Atoms, evaluation of logical interrupt expressions, and Atom/Behavior group looping. In addition, Atom/Behavior groups may be included in larger groups, each with its own looping parameter.

The MDLe Planner provides a framework for implementing custom planning algorithms using the full power of JavaScript. Users are free to develop custom planner applications in JavaScript by adhering to the established MDLe Planner framework. Planner applications may import pre-existing MDLe scripts or dynamically create new Behaviors and Plans at run-time. For maximum code reuse, MDLe scripts may be parameterized, with parameter binding occurring at run-time (see Figure 4). This allows custom plans to be generated in response to user input.

```
#parameter DESIRED_X
#parameter DESIRED_Y
#parameter MAX_SPEED
#parameter ROTATION_SPEED
#parameter DESIRED_HEADING
atom MoveTo;
atom Rotate;
behavior MoveToPositionWithHeadingB = ( (<DE-
SIRED_X>,<DESIRED_Y>,<MAX_SPEED>,<ROTATION_SPEE
D>,1.0)MoveTo
   (<DE-
SIRED_HEADING>,<ROTATION_SPEED>,1.0)Rotate,   *,
NoInterrupt);
```

Figure 4. MDLe Behavior "MoveToPositionWithHeadingB" with parameterized Atom scale vectors

One of the primary tasks of the MDLe Planner is selecting the current plan for execution. Once an MDLe plan finishes, the MDLe Planner is notified via a callback (see Figure 5), and the MDLe Planner has the opportunity to select a new plan for execution. During this time, the MDLe planner can also dynamically create and schedule a new MDLe plan for execution. User input is incorporated into the planning process through planner notification (via a callback) whenever a user-defined planner message is received by the DCF. In response to a received message, the planner may switch plans, create a new plan, or, in cases where no response action is required – simply do nothing.

```
// included MDLe scripts
include("MoveToPositionWithHeadingB.mdle");

function initialize() {
  var plan = createPlan("Surveillance");
  plan.addBehavior(getBehavior("MoveToPositionWit
hHeadingB"));
  setActivePlan(plan);
  setExitListener(plan, function planExited(){
    // switch plans, create a new plan, etc...
  });
```

Figure 5. MDLe planner which uses the "MoveToPositionWithHeadingB" Behavior.

Together, the MDLe Plan Executor and Planner components provide a loose coupling between the specification of robot control plans and their execution on a particular robot platform. During the MDLe compile stage, Atoms and Interrupts for a particular robot platform are selected from a library of available components. The mappings between *named* Atoms and Interrupts in an MDLe file (or a dynamically created Behavior/Plan) and their physical implementations is provided via a robot-specific mappings file. In addition to enabling diverse robot platforms to execute identical MDLe plans, compile-time Atom/Interrupt mapping permits multiple implementations of the same component. Such functionality is useful when a particular implementation of an Atom or Interrupt is better suited to the actual operating conditions

The DCF architecture is designed to enable easy addition of both low-level robot "drivers" and high-level control algorithms. The advantage of this approach is that it can be used for all classes of robots. For a particular class of robots, the user simply loads up the appropriate environmental, sensor, estimation, kinematic/dynamic models for simulation purposes, while for execution purposes "appropriate" robot specific hardware drivers are loaded. Robot drivers are provided by writing new Cybele Supplemental services, and algorithms are implemented by coding MDLe missions and changing the Coordination model.

Command and Control Interface

The DCF Command and Control GUI enables human operators to interact with robot teams with an intuitive interface. Using the GUI, human operators can assign tasks to the robot team simply by dragging a robot icon to the desired position (and/or orientation) and selecting a task from a popup menu of available tasking assignments. Upon task selection, an appropriate MDLe Planner/Plan is dynamically loaded and compiled, and then transmitted to the deployed robot

As shown in Figure 6, the position and orientation of deployed (or simulated) robots is represented by a colored disk with a heading arrow. Robots with assigned tasks are depicted with an outlined "footprint," corresponding to the desired position and orientation of the tasked robot. Color-coded annotations provide the name of the robot and the name of the task currently assigned.

Human operators can change (or cancel) tasking assignments by clicking on a tasked robot and selecting a new task. The position and orientation of the tasked robot can also be changed by dragging the outlined footprint of the tasked robot to a new position and orientation. The interface is well-suited to deployment on a tablet PC, where users may use a stylus to drag robots into position easily and then provide tasking assignments by tapping on the desired menu item.

The GUI periodically receives state updates from the robot team (e.g. position, orientation, task progress) and presents this information to the operator clearly and concisely. When human input is required in a decision-making process, the GUI may be instructed (by the tasked robot) to provide a dialog to the operator. The robot will then await input from the operator before proceeding with an assigned task.

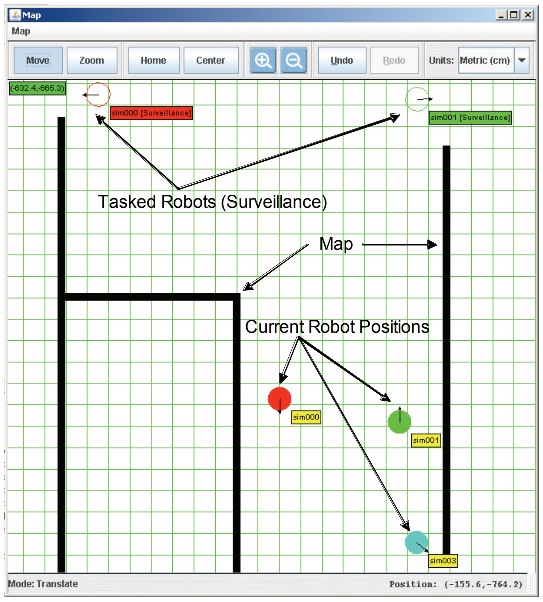


Figure 6. Screenshot of the DCF GUI

Another feature of the GUI is that it provides the ability to draw and edit maps using the integrated Map Builder. Such maps are useful for navigation, obstacle avoidance, and overall situational awareness. Maps may be entered manually, or determined experimentally using measurements taken by the robot team. Changes made to the map are immediately transmitted to the entire robot team.

Operators may customize the GUI for a particular application by providing their own implementations of the "UI Task Delegate Model" and "UI Task Viewer" models. Instances of these models determine how robot task lists are rendered on the screen and how user input is collected and converted into robot tasking assignments. For example, an implementation of the UI Task Delegate Model should provide a list of the available task assignments and handle the creation and transmission of an appropriate (MDLe) plan when a task is invoked. In addition, the UI Task Delegate Model and UI Task Viewer should handle the presentation and processing of requests for user input initiated by the robot team.

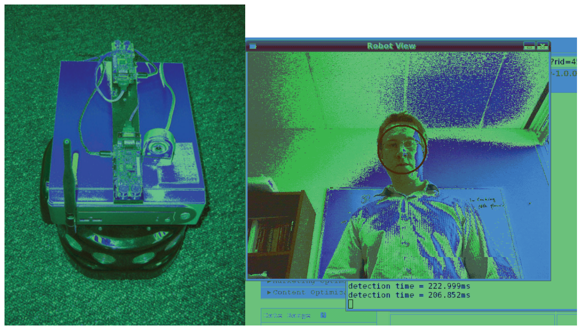
Human Robot Teaming Metrics

There has been a considerable amount of work in metrics for human-robot interaction, but these are not directly applicable to HRT in the proposed tactical scenarios [25][26]. For example conventional a metric which finds application in HRI is the time-to-completion for a particular task [26]. While this metric is certainly important, it does not provide any insight into the non time-based benefits achieved using a human/robot team with mixed initiative. We argue that for our scenario we need more sophisticated metrics that address the reduction in risk, increased situational awareness, and overall effectiveness attained using HRI with mixed-initiative versus tele-operation. While, exact metrics will evolve when system is completely built, for our preliminary work we explore following metrics:

1. Mixed Initiative Utilization: This metric provide a way to quantify and compare the human/robot team performance under two different operational modes: mixed initiative control operation and an exclusive tele-operation. The metrics can be computed based on the number of interactions (messages) passed between the human and robot team, the time spent by the human directly supervising the robot team, and the success in meeting the overall mission objectives under various failure modes (e.g. communication failure between human and robots).
2. Robot Controllability: In our HRT approach robot navigation is one of the fundamental building blocks. In a MIC framework this is done autonomously while in a tele-operated mode this is done by the human operator. This metric attempts to quantify the ability to correctly position the robots, either via tele-operation or autonomously, once the goal/intent in determined. This is of particular interest in an environment with obstacles and limited sensor information. This metric can be computed based on a combination of positional accuracy and time to maneuver to a desired state.
3. Scalability: This metric measures the performance of the human robot team as the total number of robots in the human-robot team increase. In a HRT setting we measure this as combination of time to complete the task, number of human-robot interactions, time to mediate between plans, robot-human and robot-robot communication latency, robot utilization (were all the members of the team utilized).
4. Situational Understanding - This metric measures the ability of the HRT to effectively assess the environment based on distributed sensor information. In tele-operated scenarios this is typically done in a centralized way based on sensor information collected from all members of a team. In our proposed approach situation awareness is distributed. For this effort situational understanding is measured in terms of "incorrect" assessments of the environments (intruder, intruder intent, IED location) and the number of corresponding actions taken by the human-based on the incorrect situational understanding.

HRT Testbed

To demonstrate the MIC and DCF framework for this effort we have implemented a particular setup in our laboratory that captures the Scenario discussed in Section II. It consists of a room with three entrances. The HRT consists of one human and three Amigobot robots. The AmigoBots (see Figure 7) are outfitted with a 1 GHz VIA processor (Mini-ITX form factor), an 802.11g card for ad-hoc wireless networking, and a 4 GB microdrive. For robot state (position heading and velocity) estimations a combination of the onboard odometry (incremental encoders) and a "cricket" ultrasonic indoor positioning system [10].



```
detection time = 222.999ms
detection time = 206.852ms
```

Figure 7. Robotic vehicle and integrated vision with face detection capabilities

We have developed an Extended Kalman Filter (EKF) solution to fuse the range estimates provided by the ultrasonic sensors with velocity measurements obtained from the vehicle odometry. The EKF uses measurements from two ultrasonic sensors located on the robot as far apart from each other as possible. Two sensors are needed in order to estimate the vehicle heading.

The EKF models the odometry errors as multiplicative biases. Separate biases are tracked for the translational and rotational velocities. In addition, the EKF considers the location of the ultrasonic sensors in the robot frame in order to properly account for lever arm effects. The bias estimates are fed back to the navigation system in real-time to improve the accuracy of the navigation solution. The Amigobot has also been integrated with a USB Logitech web-cam which is configured to send either a single snapshot or a video stream of a detected object of interest. Due to network bandwidth considerations, we send only a single image of the detected intruder to the human as shown in Fig 7. We use the open-source OpenCV libraries for image detection.

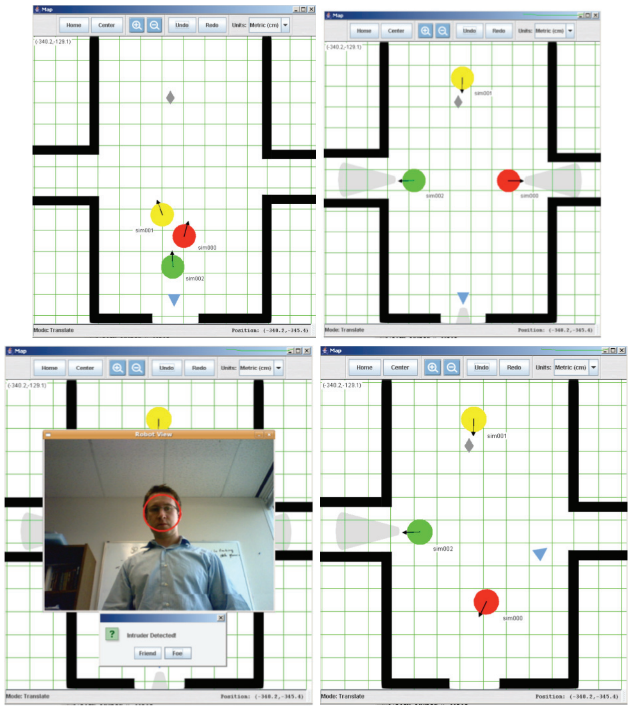


Figure 8. Simulation of the HRT scenario in DCF

Fig. 8 shows a simulation of the HRT tactical scenario in our DCF toolkit. The objective of the HRT is to secure the room and disable the IED. We assume that the IED can be located in one of the many locations in the room that are known *a priori* by the human. One of our objectives in our demonstrations is to illustrate the evolution of a sequence of actions where a human takes the initiative after identifying a "foe" based on the images transmitted by a surveillance robot. In this scenario two robots cover the east and west entrances, and one robot attempts to diffuse the IED located in north half of the room.. The robot covering the west entrance detects an intruder who is identified as "foe". Since a foe can only be neutralized by a human, the human takes the initiative to move to the west entrance. Once the robot observes the "human" intent to leave his/her position to neutralize the foe, the robot covering the east entrance repositions itself to provide coverage for the unguarded door. Another interesting scenario is when the intruder is a friend and the robot takes an initiative to jointly dismantle the IED since an additional resource (friend) is now available for the surveillance task (See Fig. 12).

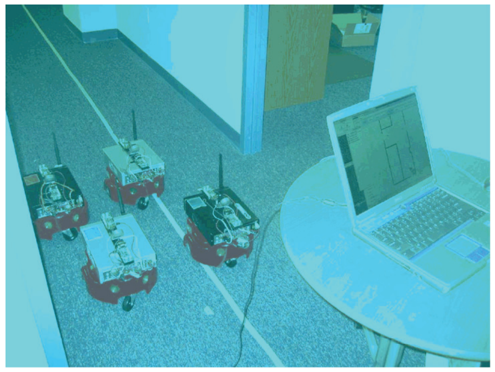


Figure 9: Tasking of Robots

Fig. 9 – 12 shows a few images of the HRT scenario being executed in the laboratory. The video of the demonstration can be found at http://www.i-a-i.com/view.asp?aid=273. In Figure 9 the Amigobot robots are in their initial configuration and are assigned tasks using the DCF command and control interface by the human. One robot is tasked to diffuse an IED , and the other three robots are tasked to provide surveillance for the east, west and north entrances. The human covers the south entrance. Fig. 10 shows the robots departing for their destination as per their tasking done by the operator. Figure 11 shows the detection of an intruder by the robot covering the west entrance. Once the intruder is identified as a "friend" who can provide surveillance of the west entrance, the robot covering this entrance takes the initiative to support the robot diffusing the IED. This can be seen in Fig 12.

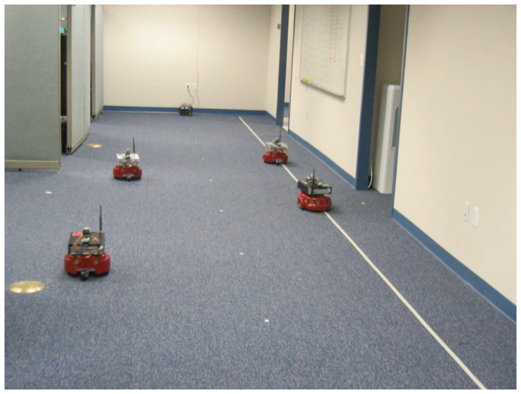


Figure 10. Robots executing their tasking

Acknowledgment

This paper was support in part from DARPA contract W31P4Q-07-C-0097. We thank Dr. Tom Wagner and Dr. Dan Corbett from DARPA for supporting this effort, their valuable suggestions and guidance. We also thank Dr. Babak-Sadjadi-Azimi for his feedback and insights on this problem.

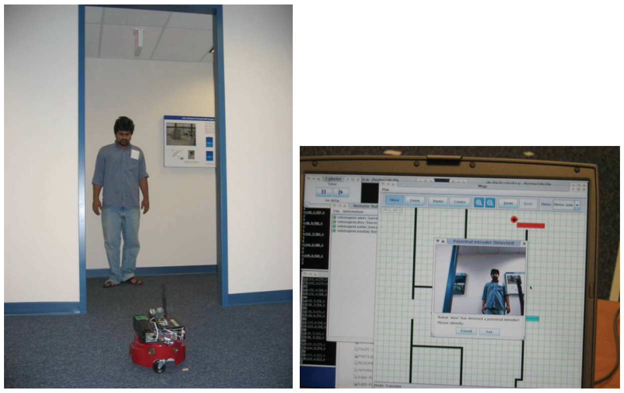


Figure 11: Robot detects a potential threat who is identified as "friend" by the human

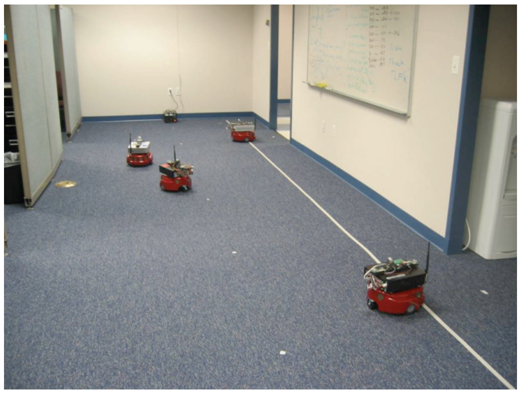


Figure 12: Robot covering the west entrance takes initiative to support the robot diffusing IED

References

[1] David L. Akin. Human-Robotic Cooperative Teams for Advanced Space Operations. ASCE Earth-Space 2004 Conference, Houston, TX, March 2004.

[2] D.V. Pynadath and M. Tambe. Automated teamwork among heterogeneous software agents and humans. Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS). 7:71–100, 2003.

[3] X. Hu, and B.P. Zeigler, "Model Continuity in the Design of Dynamic Distributed Real-Time Systems", IEEE Transactions On Systems, Man And Cybernetics— Part A: Systems And Humans, 35: 6, pp. 867- 878, November, 2005

[4] T.W. Fong, I. Nourbakhsh, R. Ambrose, R. Simmons, A. Schultz, and J. Scholtz. The Peer-to-Peer Human-Robot Interaction Project AIAA Space 2005, September, 2005..

[5] T.W. Fong, J. Scholtz, J. Shah, L. Flueckiger, C. Kunz, D. Lees, J. Schreiner, M. Siegel, L. Hiatt, I. Nourbakhsh, R. Simmons, R. Ambrose, R. Burridge, B. Antonishek, M. Bugajska, A. Schultz, and J.G. Trafton. A Preliminary Study of Peer-to-Peer Human-Robot Interaction International Conference on Systems, Man, and Cybernetics, IEEE, October, 2006.

[6] http://www.cybelepro.com

[7] V. Manikonda, G. Satapathy and R. Levy Cybele: An Agent Infrastructure for Modeling, Simulation, and Decision Support AgentLink, Issue 15, September 2004

[8] V. Manikonda, P.S. Krishnaprasad and J. Hendler. Languages, Behaviors, Hybrid Architectures and Motion Control. Mathematical Control Theory, (eds. John Baillieul and Jan C. Willems), pages 199-226, Springer, 1998

[9] Hristu-Varsakelis, D.; Egerstedt, M.; Krishnaprasad, P.S. On the structural complexity of the motion description language MDLe. Decision and Control, Dec. 2003 Page(s): 3360 - 3365 2003

[10] http://cricket.csail.mit.edu/

[11] G. Hoffman and C. Breazeal. Collaboration in human-robot teams. In Proc. of the AIAA 1st Intelligent Systems Technical Conference, Chicago, IL, USA, September 2004. AIAA.

[12] J. L. Burke and R. R. Murphy, "Human-robot interaction in USAR technical search: Two heads are better than one," Proceedings of the 13th International Workshop on Robot and Human Interactive

[13] Communication, Piscataway, NJ: IEEE, 2004, pp. 307-312.

[14] S. Balakirsky, E. Messina E., and J. Albus, Architecting a simulation and development environment for multi-robot teams, In Proceedings of the International Workshop on Multi Robot Systems,Washington, DC, 2002.

[15] X. Hu, B. P. Zeigler, "A Simulation-Based Virtual Environment to Study Cooperative Robotic Systems", Integrated Computer-Aided Engineering (ICAE), 12:4, pp. 353 – 367, October, 2005

[16] M. Lewis, K. Sycara, and I. Nourbakhsh. Developing a testbed for studying human-robot interaction in urban search and rescue. In 10th International Conference on Human-Computer Interaction, Crete, Greece, 2003.

[17] P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr,MSi, andM. Tambe. A prototype infrastruc- ture for distributed robot-agent-person teams. In Proc. of AAMAS'03, 2003.

[18] A. Finzi and A. Orlandini, "A mixed-initiative approach to human-robot interaction in rescue scenarios," in International Conference on Automated Planning and Scheduling (ICAPS), Printed Notes of Workshop on Mixed-Initiative Planning and Scheduling, pp. 36{43, 2005.

[19] W Chen, KS Decker. Managing multi-agent coordination, planning, and scheduling. Autonomous Agents and Multiagent Systems, 2004

[20] L. E. Parker, F. Tang, and M. Chandra, "Enabling autonomous sensor-sharing for tightly-coupled cooperative tasks," in Multi-Robot Systems Volume III: From Swarms to Intelligent Automata, L. E. Parker, A. Schultz, and F. Schneider, Eds. Boston, MA: Kluwer, 2005.

[21] Clodic, A.; Montreuil, V.; Alami, R.; and Chatila, R. A decisional framework for autonomous robots interacting with humans. IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), 2005

[22] Murphy, R. R., J. L. Casper, M. J. Micire and J. Hyams Mixed-Initiative Control of Multiple Heterogeneous Robots for Urban Search and Rescue, University of Central Florida:CRASAR-TR2000-1

[23] C. Laschi, C. Breazeal and Y. Nakauchi, "Collaborative Human-Robot Teamwork," ICRA Workshop 2006.

[24] S. Brown, J. Toh and S. Sukkarieh, "A Flexible Human-Robot Team Framework for Information Gathering Missions," ACRA 2006

[25] Freedy, A., McDonough, J.G., Freedy, E.T., Jacobs, R., Thayer, S.M., and Weltman, G. "A mixed initiative team performance assessment system (MITPAS) for use in training and operational environments," SBIR Phase I Final Report, Perceptronics Solutions Contract No. N61339-04-C-0020. Performance Metrics for Intelligent Systems Workshop, 2004

[26] A. Steinfeld, T. Fong, D, Kaber, M. Lewis, J. Scholtz, Alan Schultz, M. Goodrich, "Common Metrics for Human-Robot Interaction," Proc. of *HRI'06,* March 2–3, 2006, Salt Lake City, Utah, USA.

[27] http://sablecc.org/