Robocup Rescue - Virtual Robots Team STEEL (USA) MrCS - The Multirobot Control System

Nathan Brooks[1], Prasanna Velagapudi[1], Alexander Kleiner[1], Paul Scerri[1], Michael Lewis[2], and Katia Sycara[1]

[1] Carnegie Mellon University, Pittsburgh PA, USA
[2] University of Pittsburgh, Pittsburgh PA, USA

Abstract. This paper describes the software system supporting the Carnegie Mellon/Univ. of Pittsburgh team of simulated search and rescue robots in the Robocup Rescue 2010 Virtual Robots competition. Building on the Machinetta agent software, robot command and control is decomposed into a hierarchy of subtasks managed by independent agents both on the robot and colocated with human operators. By encapsulating all robot and human operator interactions into interfaces to these agents, the system can perform with a high level of robustness and reusability.

1 Introduction

In human-robot interaction, how and when the operator intervenes in the robotic system are the two predominant issues [Endsley, 1996]. How a human works with the system is a function of the level of autonomy (LOA), which describes the static function assignments between the human and the robot. The LOA can range from full manual control to full autonomy, with intermediate levels of LOA generally being superior to full autonomy or full manual control. This is because an LOA that is too high leads to degradation in manual or mental skill, loss of situation awareness, decision bias, and decrease in vigilance. The low LOA of full manual control leads to high mental demand, human decision bias, complacency, boredom, and inconsistent control behavior, all of which degrade performance. In systems with adaptive autonomy (AA), the allocation of control between the human and the robot can be dynamically changed and is usually triggered by a critical event, performance measurement, operator's workload, or the operator model. Carefully calibrated AA maximizes the amount of time the human operator can spend doing tasks which humans perform better than robots, such as victim identification and navigation of robots out of stuck or dangerous positions, problems present with the current state-of-the-art for robots. If utilized effectively, it is clearly the case that adding robots to a search and rescue team will increase the speed and rate at which useful imagery for the operator is acquired by the robots. However, as the number of robots increases, a single operator synchronously monitoring the incoming video feeds for victims

becomes infeasible. Asynchronously reviewing the video feeds retains sources of wasted operator time, such as viewing video feeds of robots that are stuck, are passing through a previously explored areas or have poor visibility. In this system, we introduce the idea of a priority queue of images which addresses these problems and additionally eliminates the context switching penalties associated with localizing victims from a video feed. Thus, the focus of the 2010 Steel team is *user-centered autonomy* and *context-free victim identification*. First we will look at the capabilities of our software packages.

2 Machinetta

The teamwork algorithms used in MrCS are general algorithms that have been shown to be effective in a range of domains [Tambe, 1997]. To take advantage of this generality, the emerging standard approach is to encapsulate the algorithms in a reusable software proxy. Each team member has a proxy that he works with closely, and the proxies work together to implement the teamwork. The current version of the proxies utilized in MrCS is Machinetta [Scerri *et al.*, 2005b], which is implemented in Java and is freely available on the internet. This type of proxy differs from many other multi-agent toolkits in that it provides the coordination algorithms, e.g., algorithms for allocating tasks, as opposed to the infrastructure, e.g., APIs for reliable communication.

The Machinetta software consists of five main modules, three of which are domain-independent and two of which are tailored for specific domains. The three domain-independent modules are designed for coordination reasoning, maintaining local beliefs (state), and adjustable autonomy. The domain-specific modules are designed for communication between proxies and communication between a proxy and a team member. These modules interact with each other only via the local state with a blackboard design and are designed to be "plug and play". This means, for examples, that new adjustable autonomy algorithms can be used with existing coordination algorithms.

The coordination reasoning is responsible for reasoning about interactions with other proxies, thus implementing the coordination algorithms. The adjustable autonomy algorithms reason about the interaction with the team member, providing the possibility for the team member rather than the proxy to make any coordination decision. For example, the adjustable autonomy module can reason that a decision to accept the role of rescuer for a civilian in a burning building should be made by the human who will enter the building rather than the proxy. In practice, the overwhelming majority of coordination decisions are made by the proxies, and only key decisions are referred to the human operators. Teams of proxies implement team-oriented plans (TOPs) which describe joint activities to be performed in terms of the individual roles to be performed and any constraints on those roles. Typically, TOPs are instantiated dynamically from TOP templates at run-time when pre-conditions associated with the templates are filled. Constraints between these roles specify interactions, such as the required execution ordering and whether one role can be performed if

another is not currently being performed. It is important to note that TOPs do not specify the coordination or communication required to execute a plan. Instead, the proxy determines the coordination that should be performed.

Machinetta includes algorithms for plan instantiation, role allocation, information sharing, task deconfliction, and adjustable autonomy. Many of these algorithms utilize a logical associates network that statically connects all team members. The associates network is a scale free network which allows the team to balance the complexity of needing to know about all the team and maintaining cohesion. The associates network's key algorithms, including role allocation, resource allocation, information sharing, and plan instantiation, are based on the use of tokens that are "pushed" onto the network and routed to where they are required by the proxies. For example, the role allocation algorithm LA-DCOP [Scerri *et al.*, 2005a] represents each role to be allocated with a token and pushes the tokens onto the network until a sufficiently capable and available team member is found to execute the role. The implementation of the coordination algorithms uses the abstraction of a simple mobile agent to implement the tokens, leading to robust and efficient software.

3 MrCS

The system architecture of MrCS is shown in Figure 1. Each robot connects with Machinetta through a robot driver that controls the robot on both low and middle levels of control. For low-level control, it serves as a broker that translates robot sensory data into local beliefs and that translates the exploration plan into robot control commands (e.g., wheel speed control). For middle-level control, the driver analyzes robot sensory data to perceive its states and local environment. Then, based on this perception, the driver overrides the control commands when it is necessary to ensure safe movement. Possible adjustments include changing the direction of motion to avoid obstacles and recovering from becoming immobilized and from a dangerous pose. The operator connects with Machinetta through the user interface agent. This agent collects the robot team's beliefs and visually represents them on the interface. It also transfers the operator's commands in the form of a Machinetta proxy's beliefs and passes them to the proxies network to allow human intervention in the loop cooperation. The operator can intervene with the robot team on three levels. On the lowest level, the operator takes over an individual robot's autonomy to teleoperate it. On the middle level, the operator interacts with a robot via editing its exploration plan. For example, the operator is allowed to delete a robot's plan to force it to stop and regenerate a plan or issue a new plan (a series of waypoints) to change its exploration behavior. On the highest level, the operator intervenes with the entire robot team by altering values of the occupancy grid in areas which are thought to contain victims, altering the regions that the robots will explore.

In this human-robot team, the human maintains the highest authority to adjust the robot team's behavior. For example, the human can change a plan during plan execution, and this plan can be further adjusted by the robot to

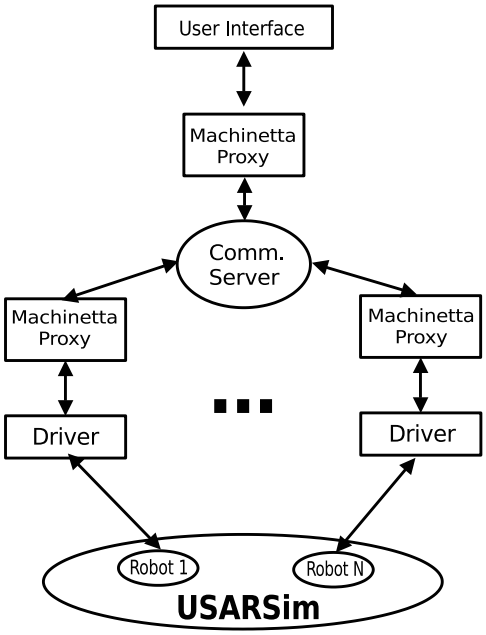

Fig. 1: MrCS architecture.

avoid obstacles or a dangerous pose. When critical events occur, such as detecting being in a dangerous pose, the robot adjusts its own behavior and informs the operator. In this case, the robot initiates the interaction and the operator can either accept the robot's adjustment or change the robot's plan. One of the challenges in a mixed-initiative system is that the user may fail to maintain situation awareness of the robot team and of the individual robots when control switching and may therefore make faulty decisions. Moreover, as the team size increases, the interventions from the robots may overwhelm the operator's cognitive resources [McFarlane and Latorella, 2002] and the operator may be limited to reacting to the robots instead of proactively controlling the robots or identifying victims [Trouvain, 2003]. We address these issues in the robot autonomy and user interface design described below.

4 Concept

The overall Steel team system can be divided into two parts. The first part is the autonomous robot behaviors that provide the infrastructure which the operator will utilize. The second part is the interface through which the operator monitors and interacts with the team. Below we describe these two elements.

5 Autonomy

Previous Steel teams have utilized some minimal levels of autonomy, but in 2010 three new components will make the Steel robots capable of significant amounts of autonomous activity. The three components are: (1) vision-coverage based autonomous search; (2) Simultaneous Localization And Mapping; and (3) robot self-reflection. Below we give a short overview of the planned capabilities of each of these components.

5.1 Vision-based coverage

Finding victims eventually relies on the operator looking at an image with the victim visible. Thus, a search should attempt to provide images of the whole environment as quickly as possible. However, most autonomous search and autonomous SLAM algorithms focus on obtaining laser scans of the whole environmentin order to build maps. While the laser scan results are critical input to computing a map of the environment, their coverage is not what should be optimized for the operator. Vision-coverage generates the same map as traditional laser-coverage based SLAM, but builds the occupancy grid for exploration purposes based on what the robot believes is the quality of imagery that has been collected for each location. This involves estimating the area of the map viewable by the camera based on the robot pose and laser scanner data with adjustments to compensate for the camera's narrow field of view and limited resolution.

We will build the capability into the autonomous search for an operator to direct one robot or a group of robots to go directly to some location and continue the search from there. While the search will remain overwhelmingly autonomous, this ability to provide some high-level input is expected to make the search significantly more efficient, especially in cases where the operator accurately recognizes the type of environment and can utilize background knowledge to guide the search.

5.2 Simultaneous Localization And Mapping (SLAM)

For our 2D representation of the environment, we generate maps from laser-based scan matching. During the last decades a rich set of solutions for building maps from 2D laser range data has been proposed, such as [Lu and Milios, 1997; Gutmann, 2000; Hähnel, 2005]. In contrast to scan matching methods, more sophisticated methods, such as *FastSlam* [Montemerlo *et al.*, 2002], and *GMapping* [Grisetti *et al.*, 2005], were introduced that correct the entire map at once when loop-closures, revisitations of locations, are detected.

Although existing methods are capable of dealing with sensor noise, they do require reasonable pose estimates such as wheel odometry as an initial guess for the mapping system. Wheel odometry tends to become unreliable given an unpredictable amount of wheel slip, which is frequently the case on the rough

terrain encountered in USAR missions. Furthermore, methods performing loop-closures are mostly not applicable in real-time since their computational needs can unpredictably increase in unknown environments.

The mapping approach utilized for our robot team focuses on the application scenario of realistic teleoperation. Under certain constraints, such as low visibility and rough terrain, first responder teleoperation leads to very noisy and unusual data. For example, due to environmental make-up and failures in control, laser scans are frequently taken under a varying roll and pitch angle, making it difficult to reliably find correspondences from successive measurements. In contrast to artificially generated data logs, logs from teleoperation seldom contain loops. Most existing methods follow the principle of minimizing the squared sum of error distances between successive scans by searching over scan transformations, such as rotations and translations. Scan point correspondences are decided only once before the search starts based on the Euclidean distance. In contrast to other methods, our scan matching approach re-considers data associations during the search, which remarkably increases the robustness of scan matching on rough terrain. The algorithm processes data from the laser range finder and gyroscope only, making it independent from odometry failures, which are likely occur in such domains due to slipping wheels. The mapping approach has been extensively tested on robot platforms designed for teleoperation in critical situations, such as bomb disposal. Furthermore, the system was evaluated in a test maze by first responders during the Disaster City event in Texas in 2008. Experiments conducted within different environments show that the system yields comparably accurate maps in real-time when compared to more sophisticated, offline methods, such as Rao-Blackwellized SLAM. More details on the utilized mapping approach are found in [Kleiner and Dornhege, 2009].

5.3 Self-reflection

In environments that are not simple and open, it is inevitable that robots will get stuck in positions from which they require human help to escape. If the team of robots is large, the operator can spend a large amount of their time simply checking whether the robots need any help. This distracts them from more important and useful tasks and can dramatically limit the number of robots a single operator can manage. We intend to reduce the time spent monitoring robots for problems by giving robots an ability to self-reflect and determine whether they are in a position that requires human help. We have a simple version of self-reflection implemented that looks at the pose of the robot, whether it is progressing along its planned path and whether it has been forced to re-plan many times in order to decide whether human input is required. As operator attention is a precious commodity, we want to ensure that robot requests for assistance will pay off in the near future. When there are many robots searching an environment, not all of them will be in equally useful positions. For example, midway through the search, some of the robots may have completely searched the part of the environment that they are in and would have a long path to travel to get to a new frontier. If there are multiple robots needing operator

assistance, there may be different values to the overall objective for assisting each of the robots. We have been developing heuristics for assessing the relative importance of each robot to the team's medium term performance. If a robot decides it is stuck but has very low importance and the operator has a large workload, it will not request help but will instead try to fix itself for a period of time before evaluating its status and importance and the operator's workload again. We hope that this ability of self-reflection will eventually free the operator from all the time they currently spend monitoring the robots for problems and fixing trivial problems.

6 User interface

The 2010 Steel user interface will build on the Multi-robot Control System (MrCS) interface that has been used in several competitions and been the basis for large human-factors experiments. Revisions to the interface have focused on the three primary tasks of the operator: (1) viewing imagery; (2) assisting robots; and (3) localizing victims. This year, the interface will consist of the following components: a 2D navigation map, a dashboard of synchronous video feed thumbnails of the entire robot team, a color coded robot status panel, a teleoperation panel and a filmstrip viewever with a list of filmstrips below it. The 2D navigation map creates a visualization of our occupancy grid and displays the location each of the robots. The map can be translated and scaled by the operator. Selecting an idividual robot on the navigation map allows the operator to edit the robot's waypoints or teleoperate the robot. Additionally, the map can be annotated by the user to mark victims, hazards or other useful information. The synchronous video feed thumbnails are present so the operator can get an initial impression of the mission environment. The color coded robot status panel serves to give the operator a quick overview of the overall team status with a single glance. In the panel each robot is represented by a small square which is colored to indicate the robot's status or task, such as exploring, performing victim localization, network anchoring, attempting a self-fix, waiting for assistance, idle or dead. The teleoperation panel provides a video feed of the robot being controlled and a simple mouse interface to drive the robot. The filmstrips are each a queue of images selected from the entire history of images taken by the robot team during their current mission. Associated with each image is the robot pose and laser scan taken by the robot at the time of capture. Visually, a filmstrip consists of textbox with its name and small thumbnails of the images in the queue to provide a sense of length. The filmstrip viewer displays the currently selected image in the active filmstrip in one window and the minimap, a smaller copy of the navigation map, in a second window. The robot pose and laser scan associated with the displayed image are superimposed on the minimap and the minimap is centered on the robot pose. The core of the user interface depends on these filmstrips, which select images based on which of the three operator tasks it is associated with.

6.1 Viewing imagery

As the number of robots in the team is increased, the task of monitoring robot video feeds becomes more time-consuming and difficult as some portion of the imagery shown to the operator will be a part of the environment already seen. We can identify three sources of redundant imagery: (1) robots that are stuck; (2) robots that are passing through previously explored areas; and (3) robots with poor visibility. To address this issue we create a single priority filmstrip, a priority queue of images sorted by uniqueness, explained below, that is updated at predefined intervals. The operator selects the priority filmstrip to load it in the filmstrip viewer and clicks through it using different buttons to advance the image depending on whether or not a victim was seen in the current image.

The priority filmstrip is created in the following process; when a robot captures a video frame, it is added to a database along with the robot's current pose, laser scanner data and uniqueness score. The uniqueness score is obtained by subtracting the occupancy grid from the camera view estimate obtained from the laser scan data and calculating the remaining area, which is the map coverage unique to that image. When the priority filmstrip requests an update, the image with the highest score is added to the update priority queue and its camera view is permanently subtracted from the occupancy grid. Priority scores are then recalculated as the occupancy grid has changed, and the resulting highest scoring image is added to the updated priority queue. This process is repeated a predefined number of times and then the update priority queue is added to the end of the priority queue. With this approach, the first two sources of redundant imagery are eliminated simply by our camera centric occupancy grid. The third source, poor visibility, cannot be realized until further imagery has been taken. For example, as a robot begins to move into the doorway of a previously unexplored room, it captures many unique images. However, once the robot reaches the doorway, it will capture the imagery of the doorway plus imagery of the interior of the room and the images taken while approaching the doorway can be identified as being less unique, or having poor visibility. Updating our priority filmstrip at predefined intervals resolves this problem.

6.2 Robot assistance

When a robot self-reflects and determines it needs assistance and the operator is not overburdened, an assistance filmstrip is created, filled with time-ordered images taken by the robot and added to the list of filmstrips. By selecting this assistance filmstrip, teleoperation control of the robot is enabled. If the source of the problem is not immediately apparent, the operator can look backwards through the filmstrip images and make necessary annotations to the minimap to help prevent similar problems in the future. Once unstuck, the operator can manually assign robot waypoints around the tricky area using the navigation map. However, if the operator decides fixing the robot is not a priority and presses the "Ignore" button, the assistance filmstrip will be deleted and the robot will attempt to autocorrect itself for a period of time. If the operator

instead determines the robot is permanently stuck, the operator can update its status to "Dead" and no further assistance filmstrips will be generated for the robot.

Robot assistance can also be provided through the navigation map to supplement our mapping algorithms. While the mapping algorithms are capable of being completely autonomous, they can be inefficient because they do not take advantage of environment features that are easily understood by an operator, especially if that operator has access to imagery from the robots. For example, if robots start at the end of a long corridor with offices all the way down, it will take a significant amount of time for a SLAM algorithm to work this out and the search may be quite inefficient. However, an operator viewing the image will immediately recognize the type of environment and may be able to design a much more efficient search strategy, for example, sending some robots to the end of the corridor to explore beyond the corridor, while a small number are left to check the offices. In the case of a cluttered environment, a robot will spend a great deal of time navigating around thin chairs and tables in order to fully map the wall behind the objects using its laser scanner. An operator can look at image in the filmstrip viewer and the sparsely populated navigation map or minimap discern if there are victims present. If a victim is present, the operator reacts normally, outlined in the next section; if not, the operator can annotate the map by drawing a box around the sparsely, yet sufficiently, outlined walls of the room and mark the entire room as being clear, allowing the robot team to move on and explore other areas.

6.3 Localizing victims

When an operator viewing the priority filmstrip sees a victim in the current frame and presses the associated "Victim" button, a localization filmstrip is created, filled with imagery taken nearby using the robot pose information in the imagery databse and added to the list of filmstrips. Addidtionally, that section of the occupancy grid is updated to appear as a desireable area so a robot will return to the area and collect more imagery.

When a sufficient number of filmstrip thumbnails appears, the operator will select this localization filmstrip and can quickly click forward and backward through the images to find one with an unobstructed view of the victim. If no such image is found, the operator can press a button to request additional imagery, which is achieved again by modifying the occupancy grid. While additional imagery is being acquired, the operator can activate a different filmstrip. Once a suitable image is found, it is used for visual victim status assesment and the minimap is used to mark the victim on the occupancy grid.

7 Conclusion

The MrCS system addresses the complex problem of urban search and rescue with a hierarchy of simple, robust solutions. Rather than solely depending on

autonomy algorithms or operator skills, tasks are decomposed into cooperative efforts between robots and operators, allowing for speed and efficiency while minimizing the frequency and criticality of failures. For Robocup Rescue 2010 we have chosen to strengthen core robot features such as SLAM and path planning while adding new features to further maximize the use of the human operator's attention for tasks such as identifying victims and assisting stuck robots. Our updates to autonomy and user interface combine together to optimize use of the operator's attention.

References

[Balakirsky *et al.*, 2007] S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V.A. Ziparo. Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue. *Journal of Field Robotics*, 24(11–12):943–967, 2007.

[Endsley, 1996] M.R. Endsley. Automation and situation awareness. *Automation and human performance: Theory and applications*, pages 163–181, 1996.

[Grisetti *et al.*, 2005] G. Grisetti, Stachniss C., and Burgard W. Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling. pages 667–672, Barcelona, Spain, 2005.

[Gutmann, 2000] J.-S. Gutmann. *Robuste Navigation autonomer mobiler Systeme*. PhD thesis, Albert-Ludwigs-Universit"at Freiburg, 2000. ISBN 3-89838-241-9.

[Hähnel, 2005] D. Hähnel. *Mapping with Mobile Robots*. PhD thesis, Universität Freiburg, Freiburg, Deutschland, 2005.

[Kleiner and Dornhege, 2009] A. Kleiner and C. Dornhege. Operator-assistive mapping in harsh environments. Denver, USA, 2009.

[Lu and Milios, 1997] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *J. Intell. Robotics Syst.*, 18(3):249–275, 1997.

[McFarlane and Latorella, 2002] D.C. McFarlane and K.A. Latorella. The scope and importance of human interruption in human-computer interaction design. *Human-Computer Interaction*, 17(1):1–61, 2002.

[Montemerlo *et al.*, 2002] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In *Eighteenth national conference on Artificial intelligence*, pages 593–598, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[Nass, 2004] C. Nass. Etiquette equality: exhibitions and expectations of computer politeness. *Communications of the ACM*, 47(4):35–37, 2004.

[Scerri *et al.*, 2005a] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, page 734. ACM, 2005.

[Scerri *et al.*, 2005b] P. Scerri, E. Liao, J. Lai, K. Sycara, Y. Xu, and M. Lewis. Coordinating very large groups of wide area search munitions. *Theory and Algorithms for Cooperative Systems*, 2005.

[Tambe, 1997] M. Tambe. Towards Flexible Teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[Trouvain, 2003] Schlick C. Mevert M. Trouvain, B. Comparison of a map- vs. camera-based user interface in a multi-robot navigation task. In *Proceedings of the 2003 International Conference on Robotics and Automation*, pages 3224–3231, 2003.