UC Merced Team Description Paper: Robocup 2009 Virtual Robot Rescue Simulation Competition

Benjamin Balaguer, Derek Burch, Roger Sloan, and Stefano Carpin

School of Engineering
University of California Merced
5200 North Lake Road, Merced CA 95343, USA,
{bbalaguer,dburch,rsloan,scarpin}@ucmerced.edu

Abstract. In this paper, we present a general description of the software architecture and algorithms to be used by the UC Merced team during the Virtual Robot Rescue Simulation Competition at RoboCup 2009. Building upon our good performance in the same competition last year, our framework and team description paper will closely resemble past work. Our goal is to concentrate on robotic cooperation and the related software issues. The approach presented will focus on the interaction of multiple and heterogeneous robotic platforms working together to achieve common goals: safely traversing terrain, localization, mapping, and victim identification. More specifically, we propose the use of two distinct controllers, each controlling different robotic platforms and exchanging information through a joint communication protocol. In addition to the discussion of specific algorithms and implementation methods, the paper will describe the overall approach used to maximize victim detection and ground coverage, essential components of the competition.

1 Introduction

The Virtual Robot Rescue Simulation League [4] provides a competitive environment where teams control simulated robotic platforms inside the Urban Search and Rescue Simulation (USARSim) [17]. During a twenty-minute run, teams attempt to achieve an assortment of tasks, which can be divided into mobility, wireless communication, victim detection, and mapping. The widely different assortment of components requires participants to develop generalized controllers capable of handling different robots and promotes reusable and portable coding. While some teams might focus on a specific research aspect of particular interest to them, they still need the other mechanisms in order to be successful in the competition. While the actual scoring scheme is unimportant for the purpose of this paper, it is worthwhile noting that the teams' performances will be determined based on the difficulty of the terrain traversed (i.e. mobility), the distance away from the base station (i.e. wireless communication), the information gathered from the victims (i.e. victim detection), and the quality of the map produced (i.e. mapping) [16].

Our approach to the competition is the usage of strongly heterogeneous robotic platforms operated by heterogeneous robotic controllers. More specifically, we use a combination of three robots, the P2AT, the Matilda equipped with its arm, and the AirRobot, as shown in Figure 1, each possessing their own strengths and weaknesses. The P2AT, on which is mounted a SICK LMS200 and a Pan-Tilt camera, offers a relatively slow but stable platform capable of operating in flat environments. The Matilda mounts a range scanner and an articulated arm. In terms of mobility, it can operate at faster speeds than the P2AT and traverse uneven terrain, to the detriment of ease of control. The AirRobot, on which is mounted a camera and a GPS sensor, gives a fast aerial presence capable of quickly spotting victims and relaying messages to the base station, in exchange for limited mapping capabilities and inferior control. The key to being successful in the competition relies on generating proper robot behaviors and cooperation schemes, described in Section 2, that exploit each robot's assets while reducing dependencies on their weaknesses. In addition, the control frameworks have to be extremely stable as competition is fierce and unrecoverable system failures separate winners from losers. Historically, the robots for the virtual robot competition have been operated through the use of a single control interface [18] guiding all the robots in the environment and administering robot cooperation. Since our research interest lies both in multi-robot cooperation and the interaction between diverse controllers, we step away from the traditional method of using a single controller. Instead, two controllers will be exploited, each controlling the different platform types: Microsoft Robotics Studio (MSRS) controls the aerial robot (i.e. the AirRobot) and the Mobility Open Architecture Simulation and Tools (MOAST) controls the ground platforms (i.e. the P3AT and the Matilda).

1.1 Microsoft Robotics Studio

Microsoft Robotics Studio provides a novel and promising control interface [12] based on the Concurrency and Coordinate Runtime (CCR) [3, 14]. The CCR framework, developed by Microsoft, offers a message-based communication protocol in which complicated and error-prone thread structures are replaced by macro-like definitions. The interesting aspect of MSRS comes from the Decentralized Software Services (DSS), built on top of the CCR, which allows developers to create services. These services, the implementation of which comes from the Representational State Transfer [9] model and the standard service framework, can be implemented as an abstraction of complete robotic platforms. In other words, a single service can individually represent mobility behaviors, manipulator operation, robot localization, mapping, etc... Each service is entirely and uniquely depicted by the service contract; an assortment of the service's characteristics, message protocol, state, and functions.

Even though each service is created independently of one another, they can communicate based on event notifications and state management through HTTP. The Distributed System Services Protocol [13] defines the message format and dictates service conformity. Specifically, messages, sent and received using ports,

Fig. 1. Simulated pictures of the three robotic platforms used by the UC Merced team. The upper-left screenshot displays the AirRobot, the upper-right screenshot illustrates the P2AT, and the lower screenshot shows the Matilda. As can clearly be seen, the chosen robotic platforms are fervently diverse.

are stored inside a FIFO queue until a specific condition is met. Once the condition is satisfied, an operation checked by the Arbiter, the proper message handler is initiated. Based on the service architecture, an MSRS controller can be created by implementing a collection of services, each of which supplies the overall application with a specific task-related solution. Since MSRS will be used to both control the aerial robot and provide a user-friendly interface capable of generating valuable information for first-responders, the tasks incorporated in the controller, described in greater detail in Section 3, will include communication, grouping, victim placement, localization, etc... In addition, an orchestration service that interfaces with USARSim will be used (from last year's competition), since no such service is part of the MSRS distribution.

Additionally, and one of the primary reason for using MSRS, the service architecture allows for the seamless transfer of code between a simulated and a real robotic platform; an appealing characteristic of the MSRS system.

1.2 Mobility Open Architecture Simulation and Tools

The Mobility Open Architecture Simulation and Tools (MOAST) package, an open-source software originally developed by the National Institute of Standards and Technology, is a universal controller capable of interfacing with simulated USARSim robots as well as real robots governed by the Player interface [8]. The approach taken by MOAST differs somewhat significantly from other popular controllers since it employs a hierarchical design based on the 4-D/RCS Reference Model Architecture [1, 10]. MOAST is comprised of five echelons, each of which performs the following similar functions: sensory processing, world mod-

eling, value judgment, and behavior generation. The five echelons are the servo, primitive, autonomous mobility, vehicle, and section echelons.

The 4-D/RCS hierarchy is built in such a way that as developers "move" up the levels, the extent of information and capabilities provided by the controller increases. Such a hierarchy can be thought of as a collection of levels, each of which is dependent on the one below it, with their own state space, resources, and resolution. Taking a laser range scanner as an example, the Servo echelon would provide the set of raw data points returned by the sensor whereas the Autonomous Mobility echelon would yield an occupancy grid map. Similar extrapolation can be applied to the other echelon levels. Since the echelons are dependent on each other, the Neutral Message Language [15] is used as a means of communication. Effectively, the MOAST controller delegates tasks to many smaller modules that can focus on individual robotic problems.

MOAST enables the portability of code from a simulated to a real robot, as is the case for MSRS, by creating low-level interchangeable wrappers that form a communication between a robotic platform (either simulated in USARSim or a real robot) and the echelon-based controller.

2 Methodology

The number of chosen robots from each categories (i.e. P2AT, Matilda, and AirRobot), will depend on the a-priori data given before each round of the competition. For outdoor environments, we expect to use a couple of P2ATs to explore trouble-free terrain comprised of even surfaces, one or two Matildas to explore uneven terrain and gain access to hard-to-reach locations, and one or two AirRobots to quickly navigate through the environment and find potential victims. Evidently, the robot composition for indoor scenarios will include a few more P2ATs since we can expect flatter terrain. The use of the AirRobot inside buildings will surely pose problems due to the lack of GPS readings, which are critical to the effective deployment of the AirRobot. The aerial platforms will be controlled by MSRS and the ground vehicles will be controlled by MOAST. Each robot will be semi-autonomous, where they can manually be controlled by an operator or explore an area autonomously. Evidently, a communication interface needs to be created so that each robotic controller can communicate with each other to successfully cooperate, as is described in further details in the next sections.

2.1 Overall Robot Behaviors and Cooperation

The incorporation of the AirRobot as part of the team of robots exploring the environment provides a significant advantage over "ground-only" teams. Indeed, with only a camera and GPS sensor, the AirRobot is the fastest, most efficient method for quickly exploring the environment. Since the AirRobot cannot help in the mapping process due to its restricted payload and cannot localize victims on its own, we utilize the AirRobot as a tool to assess the environment and

find out where possible victims might be located as well as a relay point to the communication base station. More specifically, the AirRobot will be teleoperated by the operator, through the use of a dual-axis joystick, while the ground robots autonomously explore the world. The operator will then send interesting waypoint locations (e.g. where a victim might be located) to a shared priority queue accessible by the ground robots. The priority queue will need to encompass information about the waypoints' surroundings to assure that only robots capable of reaching the objective are assigned such a waypoint. Based on the priority, location, and surrounding environment of the waypoints, one of the ground robots will incorporate the waypoint location into its current exploration scheme to, eventually, reach it.

The high-level overview of the cooperative scheme approached by the team requires a strong graphical user interface and communication protocol between the two controllers that is omitted in this paper.

3 Competition Challenges

As was briefly mentioned in the introduction, the Virtual Robot Rescue competition of RoboCup 2009 places several significant tests often observed in real world disaster scenarios. The first obstacle that teams have to bypass is the well-known localization and mapping problem. Indeed, a robot finding a victim serves no purpose if that victim cannot be localized in a geo-referenced map created by the robot. The second challenging task is to navigate through a wide selection of environment types, ranging from the office-space flat-floored surface to the uneven terrain full of debris and potentially-paralyzing crevices. The third problem facing the teams involves wireless communication, where robots need to strategically be placed in order to effectively create a communication link from any robot to the base station. Last but not least, robot cooperation is required to quickly reach a multitude of important locations.

3.1 Localization and Mapping

Thanks to the notoriety of the Simultaneously Localization And Mapping (SLAM) problem within the robotics research community, a good collection of data and algorithms recently became readily and publicly available. Consequently, the localization and mapping requirements of the competition will be achieved using an open-source SLAM algorithm running on each of the ground robots. More specifically, we will use the GMapping [11] software and integrate it inside the MOAST controller. The reason for choosing GMapping over additional open-source algorithms stems from the resulting experiments of [2] as well as its popularity. GMapping has already been integrated into USARSim controllers in past competitions and its popularity creates a knowledgebase of users available to help in the code integration.

In addition to the SLAM algorithms running on each ground robotic platform, an offline map merging algorithm needs to be implemented to yield a single

map from the collection of ground robot maps. Such a problem has also extensively been researched and we will use the techniques and results described in [5, 7] to accomplish this task. Evidently, a vast amount of modifications will be made to the ideas presented in [5, 7] so that the maps can include the following geo-referenced information: 1) victim location along with a picture of the victim, 2) explored and cleared areas of the environment, and 3) locations and shapes of particular landmarks (e.g. cars, sidewalks, debris, etc). Sample maps that include such information are shown in Figure 2.

With our interest primarily focusing on heterogeneous platforms and controllers rather than the well-understood SLAM problem, we delegate localization and mapping tasks to open-source software, which we integrate within our controllers.

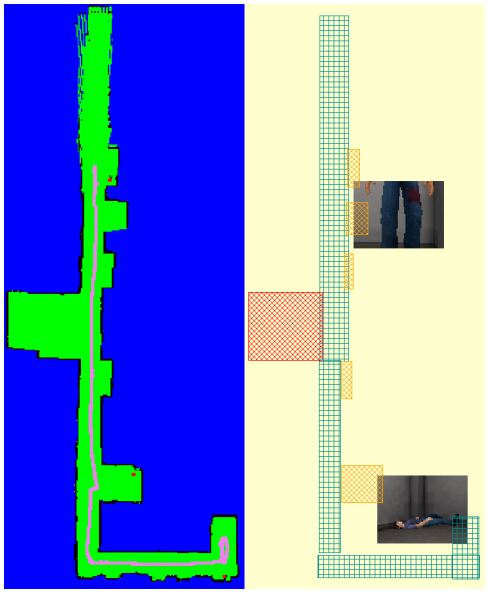

Fig. 2. Two layers of a sample map produced by our algorithm. The left side shows the occupancy grid map produced by GMapping as well as the robot's path and the victims' location. The right map shows grouping information along with victims' picture.

3.2 Mobility

In the a-priori data given before each competition round, sectors of easy, medium, and difficult mobility are given to the participants, who choose whether or not they want to go into the more difficult areas. As explained in the introduction, our team plans to explore easy mobility areas using two P2ATs, while the moderate mobility areas are explored by two Matildas. In other words, the mobility

challenge is mainly tactical and strongly depends on which robotic platforms are used (i.e. a P2AT will probably not even be able to enter a zone of moderate mobility). Evidently, better SLAM algorithms that consider the rotation of the robot or the sparce number of features would be required to appropriately map difficult-to-traverse terrain.

3.3 Wireless Communication

The communication challenge is similar to the mobility since the base station position, along with approximate coverage areas, are given as part of the a-priori data, and it is up to the individual teams to decide whether or not they want to adventure into areas of limited communication. In order to allow for robot exploration of areas with no communication coverage, we plan to rely on the capabilities of the AirRobot, which is capable of flying very quickly to the edge of the base station's communication range to effectively create a relay link. Additionally, ground robots will be exploited in similar fashion to [6].

3.4 Robot Cooperation

Our robot cooperation is two-fold. First, cooperation between the aerial robots and the ground robots is required. This cooperation is one-dimensional in the sense that the aerial robots dictate, to a certain extent, the locations and exploration areas that the ground robots need to achieve. Second, and more interestingly, cooperation between the ground robots is of crucial importance. Indeed, a cooperation scheme will be developed, where robot precedence is based on robot location, distance from goal point, mobility capabilities, and current goals, among others.

4 Real World Applicability and Future Work

The current bursting robotics research community produces a tremendous amount of new software, algorithms, controllers, and theories every year. Even though many research groups tend to develop and stick to their privately-constructed framework for robot control, open-source software is becoming more and more available. As the number of robot controllers grows along with multi-robot cooperation, a change will transpire where more attention will be given to the need for different controllers to effectively interact between each other. Our future interest is to incorporate the same methodology and architecture to real robotic platforms working together to achieve a common goal, each of which is controlled by a different controller. Since both MOAST and MSRS offer the capability of connecting to real platforms, they would be, evidently, chosen for the real world counterpart to the work presented in this paper.

Bibliography

[1] J. Albus. 4-d/rcs reference model architecture for unmanned ground vehicles. In *International Conference on Robotics and Automation*, 2000.

[2] B. Balaguer, S. Carpin, and S. Balakirsky. Towards quantitative comparisons of robot algorithms: Experiences with slam in simulation and real world systems. In *Workshop on "Performance Evaluation and Benchmarking for Intelligent Robots and Systems" at IEEE/RSJ IROS*, 2007.

[3] B. Balaguer, J. Fernando, and S. Carpin. Quantitative validations of robotic simulators: a case study with microsoft robotics studio. 2009.

[4] S. Balakirsky, C. Scrapper, B. Balaguer, A. Farinelli, and S. Carpin. Virtual robots: progresses and outlook. In *International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*, 2007.

[5] A. Birk and S. Carpin. Merging occupancy grids from multiple robots. In *IEEE*, pages 1384–1397, 2006.

[6] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks.* Series in Applied Mathematics. Princeton, 2009.

[7] S. Carpin, A. Birk, and V. Jucikas. On map merging. *Robotics and Autonomous Systems*, 53(1):1–14, 2005.

[8] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Bridging the gap between simulation and reality in urban search and rescue. In *Robocup 2006: Robot Soccer World Cup X*. Springer, 2006.

[9] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[10] V. Gazi, M. Moore, K. Passino, W. Shackleford, F. Proctor, and J. Albus. The rcs handbook. John Wiley and Sons, 2001.

[11] Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. C. grisetti and c. stachniss and w. burgard. In *International Conference on Robotics and Automation*, 2005.

[12] G. Mitsuoka. Microsoft robotics studio a technical overview for hobbysists and running the boe-bot under msrs. *Robot Magazine*, pages 50–54, 2007.

[13] H. Neilson and G. Chrysanthakopoulos. Decentralized software services protocol. http://msdn.microsoft.com/robotics/media/dssp.pdf.

[14] Microsoft Robotics Studio Runtime. http://msdn2.microsoft.com/en-us/library/bb483056.aspx.

[15] W. Shackleford, F. Proctor, and J. Michaeloski. The neutral message language. In *World Automation Conference*, 2000.

[16] A. Visser, S. Carpin, and S. Balakirsky. Robocup rescue simulation league virtual robots competition rules document, February 2009.

[17] J. Wang, M. Lewis, and J. Gennari. Usar: A game-based simulation for teleoperation. In *Human Factors and Ergonomics Society*, pages 493–497, Denver, CO, USA, October 2003.

[18] J. Wang, M. Lewis, and P. Scerri. Cooperating robots for search and rescue. In *AAMAS Workshop on Agent Technology for Disaster Management*, 2006.