UC Mercenary Team Description Paper: RoboCup 2008 Virtual Robot Rescue Simulation League

Benjamin Balaguer and Stefano Carpin

School of Engineering
[1] University of Califronia, Merced
Merced, 95340, United States of America
{bbalaguer, scarpin}@ucmerced.edu

Abstract. In this paper, we present a general description of the software architecture and algorithms to be used by the UC Mercenary team during the Virtual Robot Rescue Simulation League at RoboCup 2008. 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.

Keywords: Heterogeneous Controllers, Robotic Simulation, Multi-Robot Cooperation, Robotic Middleware

1 Introduction

The Virtual Robot Rescue Simulation League [1] provides a competitive environment where teams control simulated robotic platforms inside the Urban Search and Rescue Simulation (USARSim) [2]. 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) [3].

Our approach to the competition is the usage of strongly heterogeneous robotic platforms operated by strongly heterogeneous robotic controllers. More specifically, we use a combination of three robots, the P2AT, a tracked robot equipped with an arm, and the AirRobot, as shown in Fig. 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. As tracked robot we currently plan to use the TeleMax, although the Talon platform appears to be an interesting alternative. The TeleMax mounts a SICK LMS200, flippers, and an articulated arm encompassing four cameras, can operate at faster speeds than the P2AT and traverse uneven terrain but is more difficult to 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, to the detriment of 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. Historically, the robots for the virtual robot competition have been operated through the use of a single control interface [4] 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 controls ground robots (i.e. the P3AT and the TeleMax) and the Mobility Open Architecture Simulation and Tools controls aerial platforms (i.e. the AirRobot).

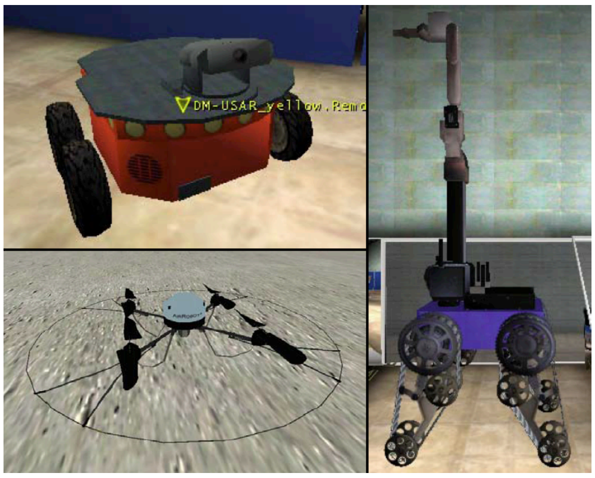

Fig. 1. Simulated pictures of the three robotic platforms used by the UCMercenary team. The upper-left screenshot displays the P2AT, the lower-left screenshot illustrates the AirRobot, and the right screenshot shows the Telemax. As can clearly be seen, the chosen robotic platforms are fervently diverse.

1.1 Microsoft Robotics Studio

Microsoft Robotics Studio (MSRS) provides a novel and promising control interface based on the Concurrency and Coordinate Runtime (CCR) [5, 15]. 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 [6] 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 [7] defines the message format and dictates service conformity. Specifically, messages, sent and received using ports, 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 control ground robots, the tasks incorporated in the controller, described in greater detail in Section 3, will include localization and mapping, communication, and mobility. In addition, an orchestration service that interfaces with USARSim will need to be implemented for the sake of the competition, since no such service exists.

Additionally, and one of the primary reason for using MSRS, the service architecture allows for the seamless transfer of code between a simulated to 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 [14]. The approach taken by MOAST differs somewhat significantly than other popular controllers since it employs a hierarchical design based on the 4-D/RCS Reference Model Architecture [8]. MOAST is comprised of five echelons, each of which performs the following similar functions: sensory processing, world modelling, value judgement, and behaviour 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 of each other, the Neutral Message Language [9] 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, Telemax, 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 TeleMax 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. We will try, however, to come up with an approximate localization algorithm in an attempt to use the AirRobot inside buildings. The aerial platforms will be controlled by MOAST and the ground vehicles will be controlled by MSRS. 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 tele-operated 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 waypoints locations (e.g. where a victim might be located) into 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 2008 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 problem facing the teams involves victim detection, which is commonly achieved by image processing or the USARSim victim pseudo-sensor. The third 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. Last but not least, robot communication is limited by a base station that is not strong enough to cover the entire environment.

3.1 Localization and Mapping

Thanks to the notoriety of the Simultaneously Localization And Mapping 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 [10] software and integrate it inside the MSRS controller. The reason for choosing GMapping over additional open-source algorithms stems from the resulting experiments of [11] 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 [12] and [13] to accomplish this task. Evidently, a vast amount of modifications will be made to the ideas presented in [12] and [13] 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…).

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.

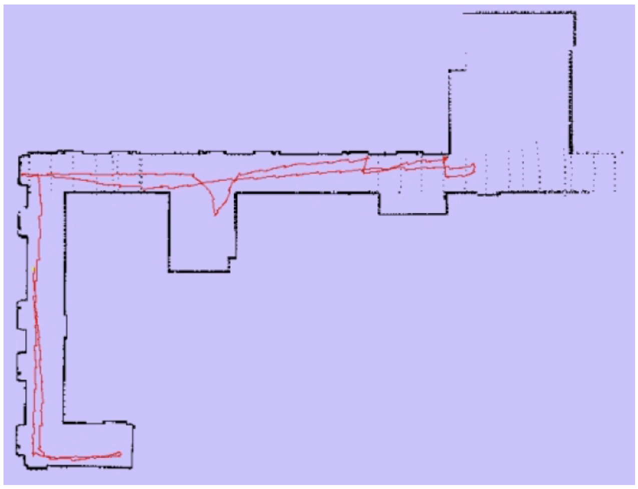

Fig. 2. A sample map produced by GMapping, using data captured by the simulated P2AT inside an office-space USARSim virtual world.

3.2 Victim Detection

We chose to try two different approaches to victim detection, both of which involve the USARSim victim pseudo-sensor. The victim sensor, which mimics body part detection image-processing algorithms, returns a set of body parts with erroneous estimates of where the body parts are. The main challenges in detecting victims using this pseudo-sensor are the error in information received, both in terms of body part location and false positive, and the dynamic movement of victims. The first approach to solve the problem focuses on machine learning. In this approach, the sensor is "trained" by being placed in front of moving "victims" and false positive. The offline nature of the training process allows for the training sessions to provide as much information as possible about what the sensor is seeing (e.g. standing, sitting, walking, etc…). The second approach uses the kinematics model for a human. Given that we know a correct kinematics model, the victim sensor should be able to extrapolate the movement of body parts and see if such movement is possible under the kinematics model. Both approaches will successfully remove false positives but require the operator to be in the loop because they depend on the amount of body parts being observed. Indeed, if a single body part is observed, there is no way for either algorithm to guarantee whether the part is a false positive or whether it is a victim being detected. As such, we include picture-taking capability to the process so that a picture is taken whenever the algorithm is not sufficiently sure of its findings.

3.3 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 TeleMax. 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 would be required to appropriately map difficult-to-traverse terrain.

3.4 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 use an additional AirRobot, whose sole purpose is to act as a communication relay. As long as the ground robots are within communication range, they send their position information to the AirRobot. When the AirRobot expects a position reading from one of the robots but does not get one, it knows that the robot in question is out of communication and flies to the last known position of the robot. From that position, the AirRobot acts as a communication relay between the ground robot and the base station. The challenge becomes more complicated when multiple ground robots are out of communication range, in which case a priority list has to be established and the AirRobot flies from one location to the next, exchanging information with different ground vehicles.

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. This paper presents a first attempt at integrating multiple heterogeneous controllers to control different robots as part of an urban search and rescue scenario, in simulation. 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.

References

1. Balakirsky, S., Scrapper, C., Balaguer, B., Farinelli, A., Carpin, S.: Virtual Robots: progresses and outlook. In: International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster (2007)
2. Wang, J., Lewis, M., Gennari, J.: USAR: A Game-Based Simulation for Teleoperation. In: Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society (2003)
3. Visser, A., Carpin, S., Balakirsky, S.: RoboCup Rescue Simulation League Virtual Robots competition Rules document, http://www.robocuprescue.org/wiki/images/Rules2008V1.pdf
4. Wang, J., Lewis, M., Scerri, P.: Cooperating robots for search and rescue. In: Proceedings of the AAMAS 1st International Workshop on Agent Technology for Disaster Management (2006)
5. Microsoft Robotics Studio Runtime, http://msdn2.microsoft.com/en-us/library/bb483056.aspx
6. Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures. PhD Dissertation. University of California, Irvine (2000)
7. Neilson, H., Chrysanthakopoulos, G.: Decentralized Software Services Protocol, http://msdn.microsoft.com/robotics/media/dssp.pdf
8. Albus, J.: 4-D/RCS Reference Model Architecture for Unmanned Ground Vehicles. In: Proceedings of IEEE International Conference on Robotics and Automation (2000)
9. Shackleford, W., Proctor, F., Michaeloski, J.: The Neutral Message Language: A model and Method for Message Passing in Heterogeneous Environments. In: Proceedings of the 2000 World Automation Conference (2000)
10. Grisetti, C., Stachniss, C., Burgard, W.: Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In: Proceedings of the IEEE International Conference on Robotics and Automation (2005)
11. Balaguer, B., Carpin, S., Balakirsky, S.: 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)
12. Carpin, S., Birk, A., Jucikas, V.: On map merging. In: Robotics and Autonomous Systems 53(1):1-14 (2005)
13. Birk, A., Carpin, S.: Merging occupancy grids from multiple robots. In: Proceedings of the IEEE 94(7): 1384-1397 (2006)
14. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: Bridging the gap between simulation and reality in urban search and rescue. In: Robocup 2006, Robot Soccer World Cup X, Springer, LNAI (2006)
15. Balaguer, B., Fernando, J., Carpin, S.: Quantitative validations of robotic simulators: a case study with Microsoft Robotics Studio (submitted for publication)