https://doi.org/10.2352/ISSN.2470-1173.2017.9.IRIACV-261

The Challenge of Preparing Teams for the European Robotics League: Emergency

J. Röning, M. Kauppinen, V. Pitkänen, A. Kemppainen, A. Tikanmäki; BISG (InfoTech Oulu) University of Oulu; Oulu, Finland; M. Furci; University of Bologna; Bologna, Italy; M. Palau Franco, A. Winfield, E. Stengler; Bristol Robotics Lab and Science Communication Unit; UWE Bristol, UK; B. Brueggemann, F. Schneider; Fraunhofer FKIE; Bonn, Germany; A. Castro; Oceanic Platform of the Canary Islands (PLOCAN); Canary Islands, Spain; M. Cordero Limon, A. Viguria; FADA Center for Advanced Aerospace Technologies; Seville, Spain; G. Ferri, F. Ferreira; NATO STO Centre for Maritime Research and Experimentation; La Spezia, Italy; Xingcun Liu, Y. Petillot; School of Eng. & Physical Sciences, Herriot-Watt University; Edinburgh, UK; D. Sosa; University of Las Palmas de Gran Canaria; Las Palmas de Gran Canaria, Spain

Abstract

ERL Emergency is an outdoor multi-domain robotic competition inspired by the 2011 Fukushima accident. The ERL Emergency Challenge requires teams of land, underwater and flying robots to work together to survey the scene, collect environmental data, and identify critical hazards. To prepare teams for this multidisciplinary task a series of summer schools and workshops have been arranged. In this paper the challenges and hands-on results of bringing students and researchers collaborating successfully in unknown environments and in new research areas are explained. As a case study results from the euRathlon/SHERPA workshop 2015 in Oulu are given.

Introduction

Collaboration between robots of different domains is necessary in many disaster scenarios, for example by utilizing unmanned aerial vehicles (UAVs) for coarse area mapping and using unmanned ground and surface vehicles (UGVs and USVs) to perform environment manipulation.

The European Robotics League (ERL) Emergency [40], **Error! Reference source not found.** competitions are aimed at advancing the field of multi-domain robotics by bringing robot teams from different backgrounds together to identify and solve issues in cooperation. In addition, a series of summer schools and workshops aimed at integration of teams from multidisciplinary fields of studies for diversifying the field of multi-domain robotics applicable in disaster scenarios, have been organised.

In this paper, the challenges and hands-on experiences of the 2015 summer school are presented. The summer school was jointly organized by University of Oulu [30] and SHERPA [31] from the 1st to the 5th June 2015 in Oulu, Finland, and a total of 42 students from ten different countries were present. This paper also explains the organizational aspects and pedagogical goals of the event and, by discussing the success and failures as a case study, aims to provide some guidance on running a technical summer school for young adults.

Motivation for ERL Emergency and SHERPA

There are numerous robotics competitions, ranging from those of mainly educational purpose (e.g. FIRST Robotics Competition [11], World Robot Olympiad [12], BEST [13]) to those whose goal is to inspire and promote new cutting-edge research with significant prizes (e.g. DARPA Robotics Challenge [14]) with numerous competitions being some mixture of the these two goals (e.g. NASA The Centennial Challenges[15], Intelligent Ground Vehicle Competition[16],International Autonomous Robot Racing Challenge [17], RoboRAVE [18], RoboGames [19], RoboCup [20],VEX Robotics Competition [21], RoboSub [22], MATE [23], SAUC-E [24], Maritime RobotX Challenge [25], RoboBoat [26], International Aerial Robotics Competition [27], Student Unmanned Air System [28], UAV Outback Challenge [29]). ERL Emergency is a competition of this mixed category and its participants range from university students to experienced academic and industry professionals. Amongst all the listed competitions, ERL Emergency is unique in its incorporation of all the three main robotics domains of air, land and water. In ERL Emergency, successful teams must be able to set up and use highly heterogeneous and interconnected robots to complete highly complex search-and-rescue (SAR) and other emergency related tasks in varied environments. In short, ERL Emergency tests the capabilities of multi-robot systems (MRS) in SAR and other disaster scenarios. The purpose of the summer school was to prepare students for employing such multi-domain robot systems in the ERL Emergency competition scenarios.

SHERPA, or “Smart collaboration between Humans and ground-aErial Robots for imProving rescuing activities in Alpine environments”, is a research project focusing on SAR operations in alpine environments utilizing advanced human-robot interaction (HRI) where one human operator controls multiple heterogeneous and semi-autonomous UGVs and UAVs. SHERPA shares ERL Emergency’s concept of utilization of MRSs and as such was a very suitable partner in the organization and running the Summer School in both academic and practical respects.

The motivation behind ERL Emergency’s and SHERPA’s focus on heterogeneous robot systems is that in many real-life disaster scenarios collaborating robots of different domains are more robust and flexible than any single multi-purpose robot [1]. The heterogeneous robot teams could be used to assist in the rescue work by, for example, mapping structurally unstable environments, finding survivors or locating radiation sources in a non-intrusive way or from locations that are inaccessible to human rescue workers. For example in the case of avalanche or other wide-area disaster UAVs could perform area mapping and call tool-carrying UGVs to a location of a survivor or other point of interest. Compared to a single robot assisting in the rescue work, heterogeneous robot teams are more robust due to redundancy, more flexible as one robot can be designed to do one task very well (e.g. snake-robot that can survey collapsed buildings) and be used during the scenario when the capabilities of the other robots are lacking, i.e. the robots can perform as a team where each member has its own strengths and weaknesses. The heterogeneous robot teams could also be a lower cost option than using high-performing and multipurpose (e.g. humanoid) robot as such excellent multipurpose robot might be very difficult to design and/or prohibitively expensive to purchase and maintain. In search and rescue robotics, UAVs and UGVs have been researched in collaborative scenarios for mapping, localization and task planning in multiple cases [5], [6], [8], [10].

Multi-domain Robot Cooperation

Cooperation between multi-domain robots has been researched widely, especially between UAVs and UGVs. Systems have been researched with varying degrees of autonomy, from fully integrated multi-robot systems with centralized robot control to separate heterogeneous autonomous robots with decentralized control and only sharing common communication interfaces [4], [8]. In the case of multi-domain robots usable in disaster scenarios, distributed decentralized methods are the most preferable due to having greater robustness against failures in the system. However, integration of heterogeneous robots is a very difficult challenge and the complexity can be overwhelming when separate robot teams need to work together to accomplish tasks in multi-domain situations. There are currently very few formal case studies on multi-domain robot teams with different backgrounds working together. Also, there currently exists no known examples where robots from multiple domains have been utilized alongside search and rescue (SAR) teams immediately after a disaster event [6].

In robot cooperation, the communication interface should be commonly agreed and only critical information should be exchanged. In a USV — UGV case presented by Weaver et al. [7], during joint actions only GPS information and mission status update information was necessary to be exchanged between platforms. All the processing was done locally as much as possible within individual robot platforms in the scope of the role taken during a joint action. When performing joint actions, it is important to define the roles and behaviors of participating robots during the encounter beforehand to minimize the required communication, as was done in the case of collaborative landing and take-off scenario between a USV and a UAV [7]. This is also most likely the best approach to use in the case of competitive scenarios within the ERL Emergency framework.

Summer School inspired by the ERL Emergency Competitions

The motivation behind organizing a multi-domain oriented summer school was to see how to get people coming from different backgrounds more involved to achieve goals as pursued in the ERL Emergency multi-domain oriented competitions. One of the main issues is to establish co-operation between teams coming from different domains to work together in disaster scenarios, aid each other in complex situations and to execute joint missions. This involves establishing joint interfaces in communication and interfacing between teams in a way that can be generally used in collaboration between different robot teams that may not have had any contact between each other previously. The summer school was also used to see what it is like to organize collaboration between robots in a multi-domain scenario when separate teams needed to work together.

The summer school organized at the University of Oulu consisted of air, water and ground domain robots, each of which required very different skillsets. Because of this, it was decided early on that each student would focus on learning about single domain of their choice in order to keep the required amount of work reasonable. Teaching every student about all the three domains in four and half days would have been both overwhelming for the students and unfeasible for the organizers to organize. The target in the end was to form teams capable of working in a simulated disaster scenario that required robots of all the three domains. As the summer school progressed it became evident that this division was mainly, but not completely, successful. Subsequent sections will describe the teaching and its successes and failures in more detail.

Figure 1. Participants and some of the organizers who were present at the very start of the summer school.

Organization of the Exercises

During the summer school, the students developed algorithms for controlling land, marine and aerial robots in several hands-on experience sessions. At the start of the summer school, each student chose which of three domains' practical sessions he or she would like to attend. Due to evenly distributed interest in the three domains, all of the students were able to follow the practical session line of their choice. The land, marine and aerial topic groups were further divided to subgroups, consisting of 3 – 4 students each. Each subgroup within a domain was given the same tasks under the instructions and guidance from the trained staff of the university of Oulu and SHERPA. The results of each group's work were presented in a challenge scenario which required cooperation between subgroups of different domains.

Design of the Challenge Scenario

The planned challenge scenario required the UGV, USV and UAV to be used simultaneously to minimize the time it takes to cover the area. The exercises preparing for the scenario were aimed to implement control algorithms for the robots in order to react to collaborative events where the robots were required to perform joint actions at selected points of interest by exchanging information with each other via a communications server. The area used was approximately 350 * 350 meters, consisting mainly of a water area and of a grassy field with no trees. The area was chosen so that the aerial drone Wi-Fi could easily cover the testing area, and so that the aerial drone could visit the land and marine robots without having to worry about collisions with tall objects.

The main common scenario objective was for all of the robots to survey the areas they can travel on, as shown in Figure 2. In addition to this, there were specific events that triggered joint actions between robots. Due to technical limitations, mainly of being afraid of driving in to deep water by accident, joint actions were not possible between the UGV and the USV. Therefore, the UAV was the most active counterpart in joint actions. The joint actions planned were for the UAV to check locations designated by the UGV and to perform a simulated sample hand-off with the USV. Basically, either the USV or UGV indicated first an event, deliver simulated sample from USV to UGV or check out location, where the UAV would then break-off from its own mission to perform the joint action. Whenever a USV or UGV initiated a joint action, they would prepare themselves for the encounter with the

UAV and wait until the joint action was completed. To avoid deadlock, such as could result from the communications server failing, the robots generally also need to have a maximum time window assigned for performing the joint action. If a joint action is unsuccessful, the robots would simply abandon the joint action and continue on their own missions.

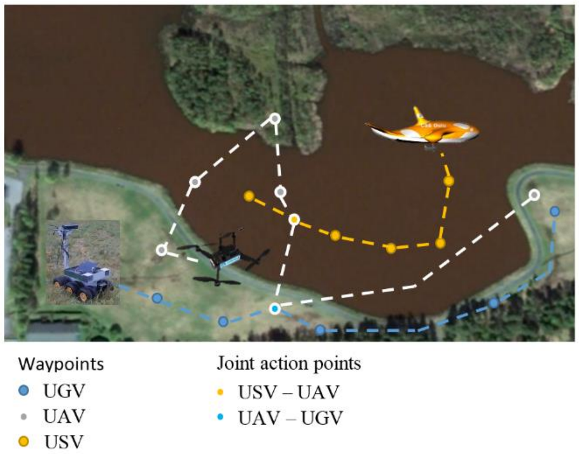


Figure 2. The scenario was designed so that each robot had their own autonomous mission following their own waypoints. There were also a couple of events designed for requiring the UAV to interact with the USV and the UGV at the joint action points.

Establishing cooperation between multi domain teams

Cooperation between robots operating in different domains is a difficult subject, especially when different teams with different backgrounds come together to achieve common goals and mission objectives. An event detection based system is most likely the easiest to implement, which is why it was chosen as the approach to use in tasks requiring joint operations to be executed between teams. In this approach, every team only needed to consider how their robot would behave during performing a joint action. Every joint action was initiated by one robot indicating an event they have detected that need a coordinated action to be performed with another robot. The event itself was also always clearly defined and simple, for example handing off a sample by one robot to another robot taken during a survey mission.

Establishing communication link between teams

In disaster situations time is of the essence and wasted time can results in significant loss of life and environmental hazards. Therefore, establishing information sharing communications needed to perform coordinated actions between all robots needs to be ensured via an easily accessible medium. For the summer school scenario, HTTP communication protocol was chosen for joint action planning out of the available web-based protocols due to being the most popular, having easy accessibility and being simple, and for those reasons being a realistic option of communication in real disaster scenarios. A simple HTTP server can be run on virtually any platform, requiring very little additional work for implementing communications. In order to minimize the communication overhead, robots need to function as autonomously as possible, only communicating the most vital information to indicate and execute significant joint efforts in the field. Similar minimal communication methods are also a realistic option in real-life disaster scenarios where available and reliable bandwidth might be a scarce resource.

For our exercises, a very simple communications server was made using the Python programming language's SimpleHTTPServer module. During operation, the programs used to control the robots could connect to the server via a TCP/IP connection. Via the server, robots were able to inform other robots of their status by updating mission related information via HTTP POST and GET messages in a common dictionary. One critical area for implementing communication between robots of different domains and teams was to very simply and clearly define the cases where the communication is needed and what should happen when a joint event has been triggered.

The reason why inter-robot communications were avoided and centralized approach was used was the added complexity of purely ad-hoc system. Also, the planned scenarios did not require a lot of data to be shared between robots. Because all of the different domains used their own systems and ways to represent data, implementing a joint interface for combining also the data representation between teams would have required unreasonable amount of work. Our summer school was in that sense a test what actually happens when teams with completely different robots come together without any previous contact with each other.

Communication protocol

The communication events were designed to be as simple as possible so that the students would not need to use too much time figuring out how to use the communications server. Data was exchanged with the communications server in JSON (JavaScript Object Notation) message format. The communication flow was kept very simple; one robot would initiate a joint action by broadcasting event information on a joint HTTP communications server by using the HTTP POST method. The message contents are then stored onto a commonly agreed data structure on the server. The downside of using HTTP is that the robot control software connected to the server needs to poll the server data using, for example, the GET method.

In JSON message format, the GET message would return the following when queried from the communications server: {"latitude": Number, "longitude": Number, "samplingLocationReached": Boolean, "waterSampleRetrieved": Boolean,}. Also, singular values could be queried using the POST message without a value. For setting new values, POST method was used with the content type being set to application/json. An example of the message contents of a POST method is the following: {"waterSampleRetrieved": true}.

The basic exchange flow of messages during performing joint action was planned to be the following; a robot would find itself in a situation where a joint action needs to be performed, the robot then updates this event to the communications server via a POST message. Other robot capable of assisting then notices this from the server by reading the information via the GET method. The joint action is agreed by both robots, for example, by using POST method to set each other as the robot they are currently helping. When the joint action is completed, the aiding robot uses POST to tell on the server that it has finished the joint action. After this, both robots are free to continue on the mission they were on previously.

Domain specific exercise implementations

As there was no centralized control structure guiding all the robots, the domain specific robot teams could concentrate towards getting the best out of the robots working in their own domains first. This most likely reflects best the situation where multiple domain specific teams with different backgrounds have assembled their systems and then need to quickly establish collaborative actions in an unforeseen situation, which in our case was running through the scenario at the end of the summer school.

On the utilization of simulators

Using simulated robots during exercises have many advantages over working with just the real robots ones. Simulated robots are cheaper, more numerous, safer, more varied sensor wise and in general easier to set up [3]. Nevertheless, experimenting with real robots in an authentic environment is, and will remain for the foreseeable future, essential as simulations can never model reality perfectly. Therefore, it was decided that the summer school will use both simulated and real robots. The students used simulators to develop control algorithms and to test their sanity before being used on the actual robots. This use of simulators aided greatly in the teaching of the summer school and allowed greater flexibility in the timing and organization of the exercises. Additionally, weather is always a factor when outdoor exercises are in question. It is good to have simulator environment available also in case the weather prohibits working with the real robots and more advanced simulators may also allow to test and validate algorithms on different weather conditions.

The simulations' proper level of abstraction should be carefully considered. The more realistic the simulation the longer it takes to set it up and the slower it will run, so in practice either the available time or available hardware will set the limit for fidelity of the simulation. From the point of efficiency, it would be best to match real sensor data, such as GPS coordinates, between the simulated and real environment and run all the robots in a single, shared, simulation environment. However during the summer school, this proved to be infeasible within the available time-frame because robots operating in the three different domains used very different simulators and tools that could not be integrated into a one simulator without excessive amount of additional work.

Land Robot

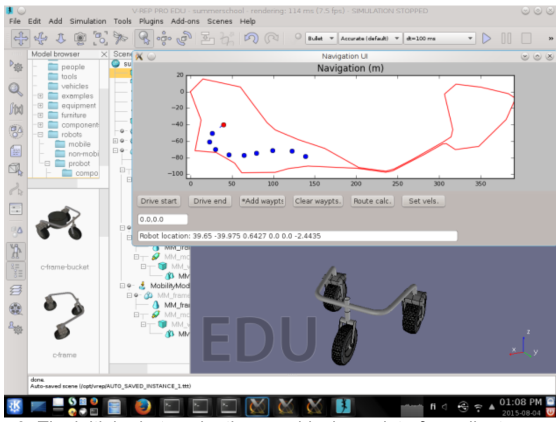
When organizing exercises, the choice of operating system is critical when the code needs to be easily portable between target systems. Porteus Linux (based on Slackware) distribution was chosen because it was fast to boot up and could be run straight from a USB stick. It is also quite compatible with different computer configurations and the included drives, although basic in their functionality, were sufficient for the purpose. The V-REP simulator [33] and any required Python packages were easily able to run on Porteus [34]. The simulator was primarily used for testing the basic functionality of the UGV and the control algorithms designed for it.

For the land robot exercises, the students implemented a control software where a very common client–server topology was utilized. In the exercises, the students worked on implementing both the robot client running on Ground Control Station (GCS) and a robot server running onboard the robot. The provided robot server for the simulator environment had readymade implementations for using some of the simulated robot's peripherals, such as motors. The task of the students was to make the robot client and server code to work together to accomplish the objectives in the assigned missions. The purpose of the robot server was to implement a common interface to the robot that would act the same regardless whether a real robot or a simulated robot was being controlled by the robot client. The robot client connected to the robot server via an IP-address, while the robot server was either connected to the simulator or running on board the real robot controlling the robot's peripherals and motors.

The exercises for the UGV group started with simulator exercises, where the basics of the used Coppelia Robotics V-REP simulator were covered with assistance of the invited main developer of V-REP. The next task was to implement the robot control interface to the simulated land robot shown in Figure 3, where the graphical user interface (GUI) and server side communications were implemented using Python 2.7.5 programming language. The goal for the land robot team was to implement the robot control using the GUI for robot navigation planning and to implement the robot motion control to the server side running on board the robot. The edge coordinates of the movement area in the outside environment were visualized in the GUI and the robot location was read from simulated and real GPS data. Finally, the produced implementations were tested outside in the testing area.

For final field testing, two real robots were prepared available for outdoor testing: a C-frame robot, consisting of modular building blocks, with four cameras and, mainly as a back-up, a six wheeled Mörri robot with GPS localization. The robots, shown in Figure 4, had been developed and constructed in close collaboration with Probot Ltd [35]. The control interface for the Mörri and C-frame robots were kept the same, so that the developed software used with the simulator could be utilized in the control of either of the robots with minimum effort. Primarily, the intention was to use the C-frame also in the outdoor exercises, which was also used in the simulator environment, but due to technical difficulties the Mörri robot was used.

The idea in the UGV hands-on sessions was to give the students hands-on experience by allowing them to come up with their own low- and high-level control solutions under the guidance of trained university staff. However, the development of these control solutions took more time than expected and less time was spent on testing and evaluating them. Also, because the robot platform had to be switched due to breakdown of the C-frame robot during the summer school, a complete control implementation was not readily available for the robot server side running on the Mörri robot. Therefore, the land robot teams did not have time to implement the joint mission with the UAV for the final scenario. Generally, the UGV exercises were considered being too difficult given the timeframe for the exercises, which was also reflected on the returned feedback surveys at the end of the summer school. It is therefore important to have some backup system for the exercises in case of technical difficulties.



0.0,0.0

Robot location: 39.65 -39.975 0.6427 0.0 0.0 -2.4435

```
done.
Auto-saved scene (/opt/vrep/AUTO_SAVED_INSTANCE_1.ttt)
```

Figure 3. The initial robot navigation graphical user interface client program and the V-REP simulator running the simulated C-Frame robot used in the

land robot group exercises. The students were given the real GPS boundaries of the final exercise area to make sure their robots will navigate correctly in the allowed movement areas.

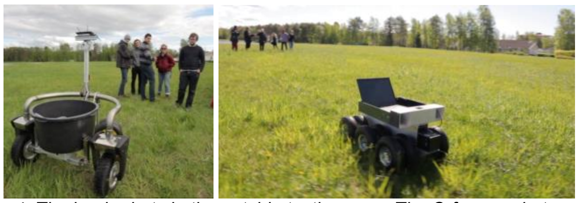


Figure 4. The land robots in the outside testing area; The C-frame robot platform on the left and the Mörri robot on the right.

Aerial Robot

The quadrotor aerial drone exercises consisted of implementing control algorithms for the drone, shown in Figure 5, which was operated through software running on a ground station PC. The control code was implemented in C# programming language, using predefined primitives, such as Goto, RotateYaw, TakePicture, etc. The ground station communicated with the aerial drone via a Wi-Fi link, using MAVlink protocol. The students also had access to high level information of the quadcopter, such as drone altitude, position in the NED (North-East-Down) frame, status of the flight battery and more. The HTTP communications server was primarily used for information sharing of the locations of the other robots and of simulated tasks, such as virtual sample collection. The goal of the students was to use the information available of the quadrotor and the other robots to build a high-level controller (state machine, trajectory generator, etc.) to handle the mission. In particular the quadrotor had to take-off after a flag raised by the other robots, reach the target robot at known coordinates with fly-to commands, grab a picture when the attitude was suitably flat, return home and land. In landing, students had to implement trajectory generation to achieve a smooth path and to detect when the quadrotor was landed to shut down the motors.

The UAV communicated via Wi-Fi with the GCS by means of MAVlink protocol over UDP, as shown in Figure 6. The high level software was arranged to offer a set of high level commands that were triggered by MAVlink messages from the GCS.

The UAV was a custom quadrotor designed by University of Bologna [36] and ASLAtech company [37].It features a compact design suitable to fly in harsh condition such as rain and snow. Its high power to weight ratio makes it suitable to fly in high wind speed conditions. The main hardware consist of an ODROID board running Linux and ROS for the high level control, a pixhawk [38] board that implements the low level controller and the actuation of the drone via the motors, a Wi-Fi antenna for communication with the ground station and a camera to grab photos and video.

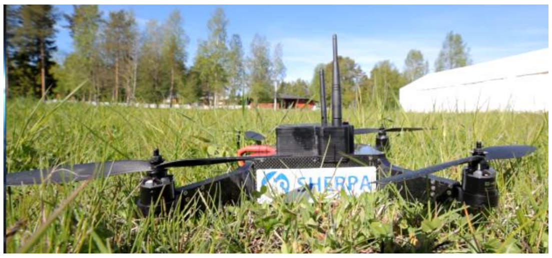


Figure 5. *The UAV provided by SHERPA.*

Although the designed goals for the UAV were quite complex, the students were given a high-level programming environment with a lot of ready made implementations for the C# environment, shown in Figure 7. Unfortunately, no simulator environment was made available for UAV exercises which limited the testing opportunities for the developed algorithms. This partly contributed in simplifying the structure of the planned exercises, which in turn resulted in many of the more advanced PhD students considering the UAV exercises to be too easy according to the feedback given by the students. On the other hand, only one out of five groups were able to completely implement the tasks required in the final scenario which also indicates that there was not enough time for all the groups to test their implementations on the real UAV. Unfortunately, the organization of the UAV exercises suffered from the lack of allocated teaching resources for implementing them. Most of the preparation work for the exercises was done by one person and working in a complex situation where much work needs to be done between robot teams from multiple domains, that is not enough. Therefore, there was less flexibility in how the students could implement their own systems and the exercises may have also been designed to be too simple from the organizational standpoint. It should be ensured that there is a right balance between the readily given implementations and what the students can make themselves.

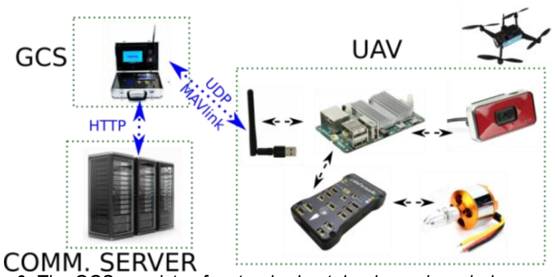
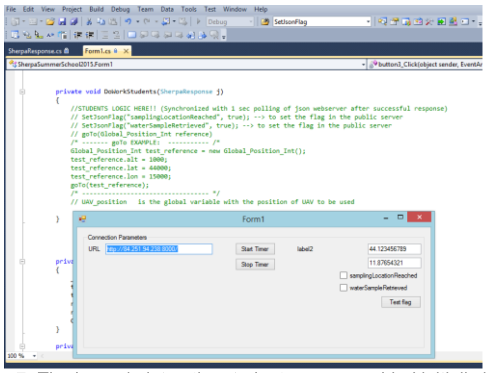


Figure 6. The GCS consists of a standard notebook running windows, mission planner and visual studio.



```
private void DoWorkStudents(SherpaResponse j)
{
    //STUDENTS LOGIC HERE!! (Synchronized with 1 sec polling of json webserver after successful response)
    // SetJsonFlag("samplingLocationReached", true); --> to set the flag in the public server
    // SetJsonFlag("waterSampleRetrieved", true); --> to set the flag in the public server
    // goTo(Global_Position_Int reference)
    /* ------- goTo EXAMPLE:  ----------- /*
    Global_Position_Int test_reference = new Global_Position_Int();
    test_reference.alt = 1000;
    test_reference.lat = 44000;
    test_reference.lon = 15000;
    goTo(test_reference);
    /* --------------------------------- */
    // UAV_position   is the global variable with the position of UAV to be used

}
```

Figure 7. The bare skeleton the students were provided initially in the UAV exercise groups.

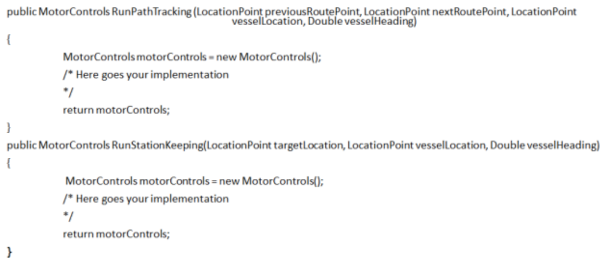
Marine Robot

In the aquatic exercises, four teams were instructed to design path tracking and station-keeping algorithms, for which a C# template is shown in Figure 8, for an unmanned surface vehicle (USV). The challenges for the USV control algorithms followed from an unknown control model and varying environmental conditions, from which wind and waves were the most significant.

After some literature review to the state-of-the-art control algorithms, students implemented and tested their algorithms in a PC class room with simulator environment. In simulations, students were using Aquamarine Robot's [39] GUI to follow how their algorithms perform with different wind conditions. The physics simulation for testing was running within the same framework used for implementing the robot control. After a group's implementation was accepted in the simulator environment, each team tested their implementation with the actual USV.

Aquamarine Robot's USV, the Dolphin (Figure 9), has length a of 3.2 meters and a width of approximately 1.5 meters, and was originally designed for a variety of water quality survey scenarios, including lake-floor mapping and water sampling from different depths. With Aquamarine Robot's GUI, shown in Figure 10 and Figure 11, user can program different sensing routes together with route point actions, and monitor task execution in real-time. The communication is based on 3G/4G connection, and for the locomotion and navigation Dolphin's on-board computer utilizes GPS, gyro-compass and two electric boat engines.

The USV exercises were the most successful having the right amount of high- and low-level work for the students. In low level, given the basics for the aquatic robots, students needed to understand and approximate how the control change in differential drivable motors change the course of the robot, and based on these findings, design basic principles for the control system. In high level, given compass heading and coordinates of the robot and route points, students needed to understand how to design a path tracking algorithm that is able to converge towards the correct path, and depending on the wind conditions, ensure robustness for the system. In addition, students needed to understand and parametrize their system, taking into account that when going to test their design with the actual robot, the behavior might differ a lot compared to the simulations due to the complex hydrodynamics. The students were given free hands to implement their algorithms to the given template and were encouraged to use the state-of-the-art from the literature. One student group's implementation actually surpassed the USV's original control algorithm on how well it was able to adapt to different wind conditions. In total, three out of four teams were able to conduct successful path tracking trials, and one of these algorithms was used in the final demonstration where the USV cooperated with the UAV.



```
public MotorControls RunPathTracking (LocationPoint previousRoutePoint, LocationPoint nextRoutePoint, LocationPoint
                              vesselLocation, Double vesselHeading)
{
            MotorControls motorControls = new MotorControls();
            /* Here goes your implementation
            */
            return motorControls;
}
public MotorControls RunStationKeeping(LocationPoint targetLocation, LocationPoint vesselLocation, Double vesselHeading)
{
            MotorControls motorControls = new MotorControls();
            /* Here goes your implementation
            */
            return motorControls;
}
```

Figure 8. Templates for path tracking and station keeping implementations.

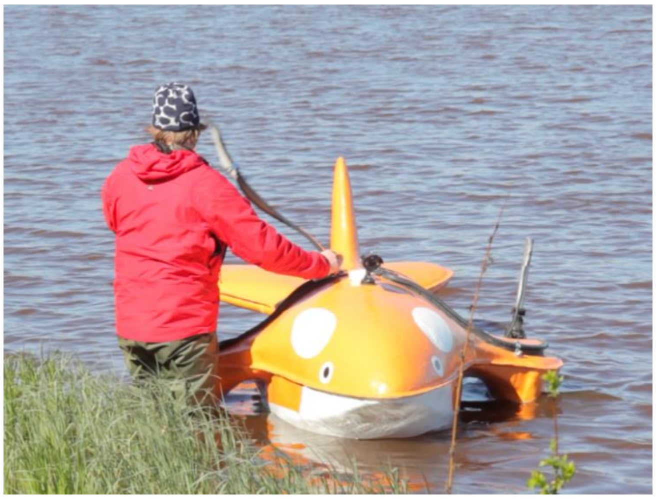


Figure 9. USV Dolphin.

Figure 10. Ground control station with the web-interface to the USV.

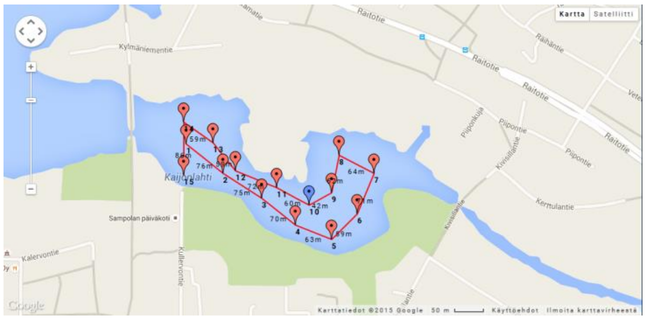


Figure 11. Waypoint route assigned to the USV. Waypoints could be programmed so that special operations could be performed at them, which was utilized to do joint actions with the UAV.

Challenge Demonstration

The original goal was to have teams to compete against each other in performing this final grand challenge, but as the end of summer school approached it became apparent that some teams' code would not be ready enough for it. Therefore, the organizers decided that in the final day the students would further test their algorithms in a real outdoor environment with real robots and only the best solutions would be used to do a demo run of the grand challenge. In the end, one cooperative mission between the UAV and USV was performed. The two robots were controlled with code made by students and they successfully used the communications server to execute a joint mission, where the marine and aerial robots transferred information via the communication server and performed a simulated sample transfer from start to end (Figure 12 and Figure 13).

In the final demonstration, after arriving to a sample point, USV sent its coordinates and instruction to the UAV to 'pick up sample' via the HTTP communications server. This was responded by the UAV by approaching the USV, taking a picture of it to indicate pick up of the simulated sample and then returning back to its initial location. After the successful sample pick up, the USV resumed going through the other waypoints in its mission.

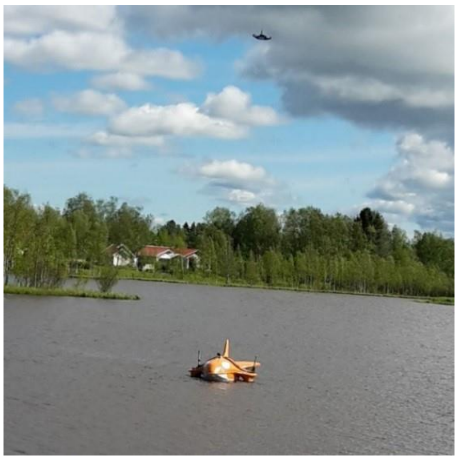


Figure 12. UAV hovering over the USV during the final challenge demonstration.

Figure 13. The picture the UAV took to simulate the sample pickup.

Organizational Aspects

This section briefly summarizes some of the practical issues, some of which may also seem quite obvious and trivial, that should be taken in account when organizing an international robotics event such as a summer school. Even small errors can cost if not money, then at least significant amounts of time to correct. The main advice is that, as with everything, experience helps. So if at all possible, recruit or ask advice from a person / persons who have experience in organizing similar events. Secure the support of your organization's financial staff. Especially with fiscal issues, small matters can lead to tedious bureaucratic situations. Try to have somebody with actual pedagogical experience planning and running the exercises as expertize with a given field does not always translate to teaching prowess, especially so if the students are not familiar with the subject matter.

Usually only a limited amount of time and effort can be given to any project and the limited resources need to be prioritized. In the case of summer schools, it is the recommendation of the authors to ensure the basic utilities such as food, transportation and accommodation are especially well organized, even at the cost of teaching, as tired and hungry students will not much care about the quality of teaching. Also note that a major part, perhaps the main part in some cases, of any summer school is the networking and informal exchange of ideas. Opportunities for this should be provided, for example by poster sessions, social events and group assignments. The overall schedule and preparations should be flexible and simple enough to allow small adjustments due to unforeseen or overlooked issues.

Call and Registration

The call for students should be sent well ahead of time and advertised in as many places as possible (e.g. mailing lists, homepages of affiliated organizations, social media). The deadline for registration need to be set according to the deadlines agreed upon with the utility providers (e.g. hotels, bus companies, cafeterias), that need the final number of participants. Do try to stick to the deadlines and especially to the maximum number of students you have set up as last minute deviations from them will usually cause disproportionate amounts of extra work, especially if contracts with utility providers need to be updated. If changes to registration deadline seem necessary due to lack of participants, do it only very close to the original deadline as usually the highest amount of students enroll very close to the deadline.

The call for students and registration instructions should be as clear and complete as possible. Even one or two small questions received from each interested candidate or selected student, due to a lack of information provided in the instructions, can quickly cause days of work in replying emails and changing instructions may lead to additional confusion. Obviously, the returned registrations forms should provide all the basic information from the students such as names, contact information, dietary and accessibility issues but also information about the relevant technical experience so that the scope of the exercises can be fine-tuned and balanced student teams formed.

Close to the beginning of the summer school, the students should receive an information-package containing at least payment instructions, relevant addresses, event kick-off time and explicit instructions on how to locate kick-off location.

Preparation at the Venue

Carefully choose the type of exercises as they significantly complicate the organization and execution of any summer school, especially so when done with actual machines that have a chance of breaking down. If some of the exercises are planned to be outside, have also some alternative plan if there is any chance that the weather might be unsuitable.

Integrating heterogeneous systems can take significant amount of time and effort from the organizers. Therefore, to minimize effort and unforeseen time-consuming problems, preferably only one whole system provided by one organizing partner should be used throughout the summer school. It is vital to begin system integration as early as possible, especially if the MRS is composed of robots from multiple organizing partners. In the case of the summer school presented in this paper, the organizers underestimated the required time which was the main cause of the problems encountered during the hands-on exercises. If possible, it is a good idea to give the simulator environment and some preliminary exercises to the students a few weeks before the event. This way minimum amount of time is spent on the basics which many students are probably well familiar with.

Fine-tune things with the utility providers, for example cooperate with the cafeteria staff so that students have clear instructions on how to operate at the meal lines. Do bargain with the utility providers. Summer school usually involves tens of participants, and thus significant amount of money, so the utility providers are likely to be more flexible with their terms and conditions.

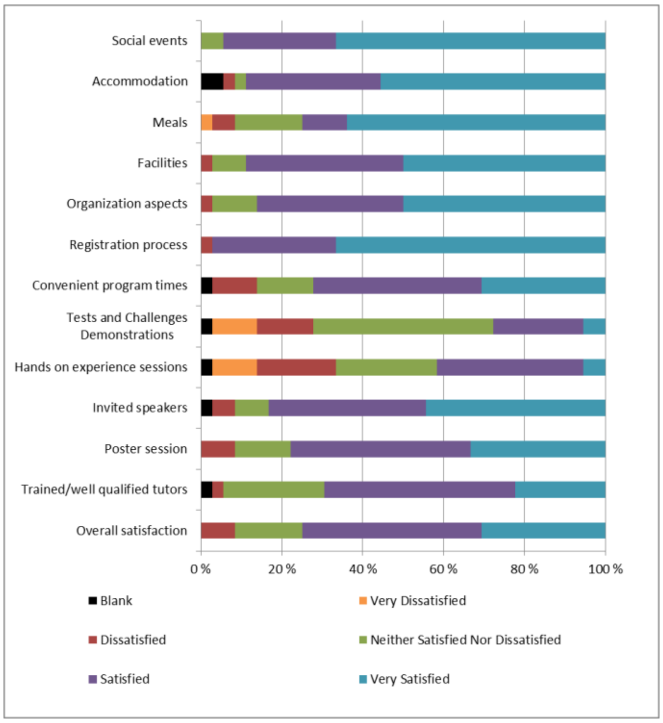
Operation Guidelines During the Event

No matter how well a given event is prepared, there will always be some unexpected, overlooked and/or just unlucky issues that will require action on the fly and at least one organizer should be reserved for dealing with those issues. If required, do not be afraid to command the students. They will understand that smooth operation requires willing cooperation. Furthermore, to keep the schedule, treat the students as the adults they are, e.g. if someone misses the pre-arranged bus journey then it's up to that person to go to where the others are going. You can of course try to help people also individually but do not delay the organized event unnecessarily because of few individuals.

Final Thoughts

Four and a half days is a very short time to teach and deciding on the correct scope of teaching is challenging, especially so when the topic is advanced and students have wildly different backgrounds. The exercises should be challenging enough to keep the students engaged. On the other hand, there is only very limited time to teach in which the tasks must be achievable. Overly hard exercises are frustrating and can be a pointless test of patience.

In hindsight, the original goal of meaningful co-operation between different domains was too demanding given the time reserved for event organization and for the time reserved for practical hands-on sessions. The lack of time was also reflected in the feedback of some of the students who needed more time especially for testing their implementations on the actual real robots Nevertheless, despite the setbacks, the students focusing on different domains successfully integrated a highly heterogeneous systems of different domains in the very limited time available and the majority of the students were overall satisfied with the summer school (Figure 14).



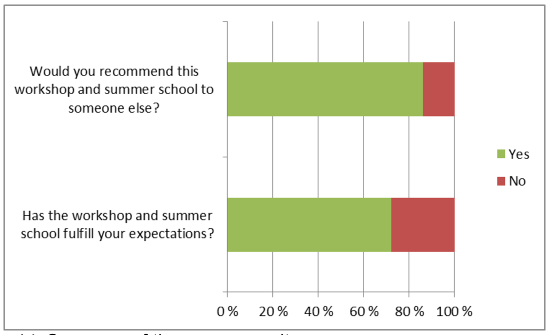


Figure 14. Summary of the survey results.

For future years, one way of advancing the scope of the summer schools, and the ERL Emergency competitions, is to involve more numerous cooperating robots. This would help to push the practical benefit of robots in SAR missions. At the moment, real-life SAR operations have only a few, if any, robots assisting the rescue workers so there is significant potential for more extensive MRS utilization, especially for the critical first 48 hours after which the likelihood of finding survivors drops rapidly in disaster scenarios. Currently, there are on average two human operators per one robot in a typical SAR mission, but simulation tests have implied that up to eight to twelve robots can be operated by a single operator with good efficiency [2]. One of SHERPA's goals is indeed to increase this ratio and ERL Emergency could push the usage of MRS in actual SAR operations by somewhat shifting its focus to allow and encourage more robots per competition team. Other sources for inspiration could be other EU research programs such as DARIUS, ICARUS, TIRAMISU and TRADR.

As far as the authors are aware, at the moment there are no summer schools where students could control multiple SAR – capable robots at once (either directly or as a swarm). Of course, the practicalities of such large scale summer school, or competition, could be exceedingly complex and expensive as even using a few robots can put significant strain on the summer school organizers. However, at least in the case of summer schools, the cost and organization issues could be significantly mitigated by using simulators and Virtual Reality more extensively.

References

[1] A. Khamis, A. Hussein and A. M.Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art,"Cooperative Robots and Sensor Networks, Springer International Publishing, pp.31-51, 2015.

[2] P. Velagapudi, P. Scerri, K. Sycara, H. Wang, M. Lewis and J. Wang "Scaling effects in multi-robot control, "International Conference on Intelligent Robots and systems (IROS'08), 2008.

[3] R. Tellez, "A Thousand Robots for Each Student: Using Cloud Robot Simulations to Teach Robotics," Robotics in Education, vol. 457, pp.143-155, 2016.

[4] Z. Yan, N. Jouandeau and A. Ali Cherif, "A Survey and Analysis of Multi-Robot Coordination," International Journal of Advanced Robotic Systems, vol.10, 2013.

[5] D. Serrano, G. De Cubber, G. Leventakis and S. Govindaraj, "ICARUS and DARIUS approaches towards interoperability,"IARP RISE Workshop, Lisbon, Portugal, 2015.

[6] F.E. Schneider and D. Wildermuth, "Assessing the search and rescue domain as an applied and realistic benchmark for robotic systems," 17th International Carpathian Control Conference (ICCC), Slovakia, 2016.

[7] J. N. Weaver, D. Z.Frank, E. M. Schwartz and A. A. Arroyo, "UAV Performing Autonomous Landing on USV Utilizing the Robot Operating System," ASME District F - Early Career Technical Conference, Birmingham, Alabama, 2013.

[8] Z. Beck, L. Teacy, A. Rogers and N. R. Jennings, "Online Planning for Collaborative Search and Rescue by Heterogeneous Robot Teams," Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, 2016.

[9] Y. Cao, W. Yu, W. Ren and G. Chen, "An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination," IEEE Transactions on Industrial Informatics, vol. 9, no. 1, pp.427-438, 2013 http://www.igvc.org/

[10] G. Christie, A. Shoemaker, K. Kochersberger, P. Tokekar, L. McLean and A. Leonessa, "Radiation Search Operations using Scene Understanding with Autonomous UAV and UGV," arXiv:1609.00017, 2016.

[11] http://www.firstinspires.org/robotics/frc

[12] http://www.wroboto.org/

[13] http://www.bestinc.org/

[14] http://www.darpa.mil/program/darpa-robotics-challenge

[15] https://www.nasa.gov/directorates/spacetech/centennial_challenges/index.html

[16] http://www.igvc.org/

[17] https://robotracing.wordpress.com/

[18] http://roborave.org/

[19] http://robogames.net/index.php

[20] http://www.robocup.org/

[21] http://www.vexrobotics.com/vexedr/competition/

[22] http://www.robonation.org/competition/robosub

[23] http://www.marinetech.org/international-competition/

[24] http://www.sauc-europe.org/

[25] http://www.robotx.org/

[26] http://www.robonation.org/competition/roboboat

[27] http://www.aerialroboticscompetition.org/

[28] http://www.auvsi-suas.org/

[29] https://uavchallenge.org/

[30] http://www.oulu.fi/university/

[31] http://www.sherpa-project.eu/sherpa/

[32] https://eu-robotics.net/robotics_league/

[33] http://www.coppeliarobotics.com/index.html

[34] http://www.porteus.org/

[35] http://probot.fi/

[36] http://www.unibo.it/en/homepage

[37] http://www.aslatech.com/

[38] https://pixhawk.org/

[39] http://www.aquamarinerobots.com/

[40] A. F. Winfield, M. P. Franco, B. Brueggemann, A. Castro, M. C. Limon, G. Ferri, F. Ferreira, X. Liu, Y. Petillot, J. Roning, F. Schneider, E. Stengler, D. Sosa and A. Viguria, "euRathlon 2015: A multi-domain multi-robot grand challenge for search and rescue robots," Alboul L., Damian D., Aitken J. (eds) Towards Autonomous Robotic Systems. TAROS 2016. Lecture Notes in Computer Science, vol. 9716 , pp. 351-363, 2016.