BUILDING A BRIDGE TO ROBOT COSPACE

FATEMEH PAHLEVAN AGHABABA, AMIRREZA KABIRI
In this Demo, the followings will be studied in reasonable details:

- Rescue Virtual robot league
- CoSpace robot league

At first we will have a look at Rescue Virtual robot league then the CoSpace robot league will be studied. Afterwards two approaches are suggested to Filling the Gap between RCJ Cospace and Virtual Rescue Robot competition.

This Demo designed as part of the “Filling the Gap between RCJ Cospace and Virtual Rescue Robot competition“ workshop which is organized by the sponsorship of Robocup international federation and hosted by Cambridge University, London, UK.
Robocup Rescue Simulation

According to event organizers, "The intention of the RoboCup Rescue project is to promote research and development in this socially significant domain at various levels involving:

- multi-agent teamwork coordination
- physical robotic agents for search and rescue
- information infrastructures
- personal digital assistants
- standard simulator and decision support systems
- evaluation benchmarks for rescue strategies and robotic systems

All could be integrated into comprehensive systems in future.
The state of the art in the virtual robot competition is a control system able to control a large team of robots with a single operator.

To make this possible, the robots are able to create a map of the disaster and use this map to make automatic decisions for an optimal search strategy.

Several teams have shown that they are able to detect victims by processing the images from the robot camera, based on features like color, shape and movement. Also other sources like sound are studied.

The Virtual Robots competition simulates small teams of agents with realistic capabilities operating on a city block-sized scenario.
Rescue Simulation Virtual Robot Competition

- First held in 2006.
- The final purpose of this competition is to provide a common benchmark to demonstrate scientific progress in the application of robotics to Urban Search and Rescue (USAR).
- In addition, this league also encourages intuitive operator interfaces and autonomous and semi-autonomous algorithms that can be used to supervising and control multiple heterogeneous robots operating in challenging environments.
Rescue Simulation Virtual Robot Competition

Evaluation process:
• the number of victims,
• the covered area,
• and the elapsed time.

The Virtual Robot Contest 1st Place in the first competition (in 2006) was gone to the Freiburg team, from University of Freiburg, Germany. And the last year’s 1st Place award went to the Cambridge team, from Cambridge University, London, UK.
Research Topics

- Autonomous multi-robot control
- Human multi-robot control interfaces
- Localization, 3D mapping
- Navigation and exploration
- Robust sensory fusion algorithms
- Perception algorithms
- Development of novel mobility modes and sensor processing skills
- Multi-agent cooperation with a controlled real-time factor
- Robust Human Detection based on combination of different features such as voice, temperature
- Distributed planning strategies

Fatemeh Pahlevan Aghababa, Amirreza Kabiri
Research Topics

- Autonomous multi-robot control
- Human multi-robot control interfaces
- Localization, 3D mapping
- Navigation and exploration
- Robust sensory fusion algorithms
- Perception algorithms
- Development of novel mobility modes and sensor processing skills
- Multi-agent cooperation with a controlled real-time factor
- Robust Human Detection based on combination of different features such as voice, temperature
- Distributed planning strategies
Search & Rescue Scenario

- The simulated world includes both indoor and outdoor environment
- The indoor map includes a maze of walls, doors, and elevated floors provide various tests for robot navigation and mapping capabilities. Variable flooring, overturned furniture, and problematic rubble provide obvious physical obstacles.
- The objective for each robot entering the arenas, and the incentive to traverse every corner of each arena, was to find simulated victims.
- Each simulated victim was a clothed mannequin emitting body heat and other signs of life including motion (waving) and sound. This sensor signatures implied the victim’s state: Alive or dead
- Teams will be required to search for victims located in different places in the arena and park their robots “near” each alive victim (somewhere in radius of 1.5 meters around the victim, but Technical Committee has the final word on when a robot is enough “close”)
- Teams can be asked to submit maps generated by their robots to the Technical Committee for scoring, for comparing performance, and for sharing information between teams.

Fatemeh Pahlevan Aghababa, Amirreza Kabiri
At first USARSim has been chosen as simulation infrastructure for a virtual robot competition and it was used until 2015.
At 2016, Robocup Virtual Robot Committees decided to change the server to Gazebo/ROS
In the following we will discuss ROS & Gazebo.
Robot Operating System (ROS)

- supported by the Open Source Robotics Foundation (OSRF),
  - in 2007 with the name Switchyard
    - Willow Garage

A meta operating system

- performing many functions of an operating system but it requires a computer's operating system such as Linux
- Provides communication between the user, the computer's operating system, and equipment external to the computer including sensors, cameras, as well as robots
- and the ability to control a robot without the user having to know all of the details of the robot

Other robot frameworks are such as Player, YARP, Orocots, CARMEN, Orca, MOOS, and Microsoft Robotics Studio.
Gazebo

- Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments.
- While similar to game engines, Gazebo offers physics simulation at a much higher degree of fidelity, a suite of sensors, and interfaces for both users and programs.
- Typical uses of Gazebo include:
  - Testing robotics algorithms
  - Designing robots
  - Performing regression testing with realistic scenarios
- A few key features of Gazebo include:
  - multiple physics engines
  - a rich library of robot models and environments
  - a wide variety of sensors
  - convenient programmatic and graphical interfaces
ROS Gazebo

- Gazebo is an independent project like boost, ogre or any other project used by ROS.
- Usually, the latest major version of gazebo available at the beginning of every ROS release cycle (for example gazebo7 for ROS Kinetic) is selected as the official one to be fully integrated and supported and will be kept during the whole life of the ROS distribution.
- Gazebo development is not synced with ROS, so each new major version of Gazebo must be released before being used in a ROS distribution. But Gazebo website cover how to use ROS with different versions of Gazebo.
- The set of ROS packages for interfacing with Gazebo are contained within a new meta package named gazebo_ros_pkgs.
ROS Gazebo (gazebo_ros_pkgs)

- gazebo_ros_pkgs is a set of ROS packages that provide the necessary interfaces to simulate a robot in the Gazebo 3D rigid body simulator for robots.
- gazebo_ros_pkgs integrates with ROS using ROS messages, services and dynamic reconfigure.
- Some of the features of gazebo_ros_pkgs include:
  - Supports the latest stand alone system dependency of Gazebo, that has no ROS bindings on its own
  - Builds with catkin
  - Treats URDF and SDF as equally as possible
  - Reduces code duplication with Gazebo
  - Improves out of the box support for controllers using ros_control
  - Integrates real time controller efficiency improvements from the DARPA Robotics Challenge
Gazebo Plugins

- A plugin is a chunk of code that is compiled as a shared library and inserted into the simulation.
- The plugin has direct access to all the functionality of Gazebo through the standard C++ classes.
- Plugins are useful because they:
  - let developers control almost any aspect of Gazebo
  - are self-contained routines that are easily shared
  - can be inserted and removed from a running system
- Previous versions of Gazebo utilized controllers. These behaved in much the same way as plugins, but were statically compiled into Gazebo.
- Plugins are more flexible, and allow users to pick and choose what functionality to include in their simulations.

Fatemeh Pahlevan Aghababa, Amirreza Kabiri
Gazebo Plugins

- You should use a plugin when:
  - you want to programmatically alter a simulation (move models, respond to events, insert new models given a set of preconditions)
  - you want a fast interface to gazebo, without the overhead of the transport layer (No serialization and deserialization of messages.)
  - you have some code that could benefit others and want to share it
- There are currently 6 types of plugins
  - World
  - Model
  - Sensor
  - System
  - Visual
  - GUI
What is CoSpace Robot?

The CoSpace Robot, developed by the Advanced Robotics & Intelligent Control Centre (ARICC) of Singapore Polytechnic, consists of real robot, virtual robot, and simulator.
What is Cospace Robot?

The CoSpace Robot has the following features:

✔ Customisable 3D virtual environments with simulated real-world physics
✔ Easy-to-learn and interactive graphical interface
✔ Support high level programming language coding for the advanced users
✔ Both virtual and real robot runs on same codes
✔ Safe and conducive environment for users to experiment
✔ FREE CoSpace Virtual Simulator for students
Cospace Robot league

- Soccer
- Dance
- Rescue
Building a bridge to robot cospace Target

Cospace rescue robot league contains two competitions (Primary and Secondary). Junior cospace rescue robot teams start their journey from primary competition and continue their work on secondary.

The Target of this joint project is to build a bridge from cospace rescue robot to major rescue simulation league virtual robot competition.
Cospace Rescue Virtual Robot model

Led

Left Ultrasonic Sensor

Front Ultrasonic Sensor

Right Ultrasonic Sensor

Back Ultrasonic Sensor

Compass Sensor

Wheels

Left Colour Sensor

Right Colour Sensor
Cospace Rescue Virtual Robot model

The robot has 3 outputs:
- 2 independently controlled motors at the back of the robot
- 1 LED which can be set to steady on or flashing

The robots also have a range of sensors:
- 4 ultrasound sensors that measure the distance of objects in front of the sensor in centimetres. These are fitted to the front, left, right and back of the robot. These can be used to detect walls and boundaries.
- 2 colour sensors used to detect the colour of the field below the sensor in order to detect the different coloured objects. They return a red, green, blue values ranging between 0-255
- 1 Compass sensor at the top of the robot, which measures the angle in degrees where 0° is facing the scoring board. The angle is compass bearing is measured anticlockwise.

Fatemeh Pahlevan Aghababa, Amirreza Kabiri
Cospace Rescue Real Robot model
Cospace Rescue World model

Objects to be collected
Blue special zone (double points!)
Blue trap and yellow warning area
Pink line leading to deposition area
Obstacle
Orange deposition area
Cospace Rescue Scoreboard

Each Robot can collect at most 6 object. Objects are defined using colors and different colors have different score value.
The Cospace rescue robot have a control panel which is used for both Primary and Secondary competitions.

This control panel have the following functionalities:

1. Robot sensor's viewer
2. Robot manual controller with joystick, mouse and keyboard
3. Change robot position in the world model
4. AI programming panel which used event driven programming approach. This panel contains a powerful graphical interface which is more popular along primary teams.
Cospace Rescue Control Panel

5. C/CS compiler and debugger  
6. Competition simulator  
7. Loading the programs as a dll file to robots

It should be mentioned that most of the cospace rescue primary teams use the control panel’s graphical user interface (GUI) to develop their programs which will be translated to ‘C’ code by the interface. This happens while most of the secondary teams develop their strategies using ‘C’ language directly.

In the following slides, first we will discuss about AI development using the GUI and then writing the code directly in ‘C’. Our aim is to find a way in which the junior team can use their old developed programs in the new server and make their migrations as easy as possible.
Cospace Rescue Control Panel

- **Robot selection** – this allows the user to select which robots’ sensor values to view; the live sensor values for the Blue Virtual Robot or the Red Virtual Robot are displayed according to team’s selection.

- **Real-time sensor feedback** – this allows the user to view the sensors readings as the robot moves through the virtual environment. This is useful for obtaining readings from the colour sensors.

- **Manual Robot control** – by moving the round joystick controller, the direction and speed of the robot can be controlled manually. Teams can use this to position their robots over a particular color and obtain the corresponding sensor values.
Cospace Rescue Control Panel

Competition tab is the most important tab, and allows the teams to program their robot and also load and run their programs.

It also helps teams to relocate their robots, changing the stages (world1, world2) and develop their code using the AI GUI.
Cospace Rescue Control Panel

8 minute countdown timer

If one of the robots gets stuck in a loop or stuck on an obstacle you can click on the relocate button which will move the robot slightly.

Click on this to award a penalty, a dialog box pops up asking to which robot to give the penalty to.

If one of the teams want to quit on the game, they can click on these coloured buttons.

Scoring for the two teams, automatically updates

To load a program into either the blue or red robot, click on the respectively coloured robot icon and select the program location when the dialog box opens.

Launches the AI development panel for programming.

Launches the help system

Please note parts of this are out of date

Start – starts a competition
Pause – pauses the competition
Stage 1 End – allows you to move immediately on to stage 2 when testing programs
Reset – resets the game, you will need to reload the programs
AI development using Cospace Rescue GUI
AI development using Cospace Rescue GUI
Cospace Rescue GUI- Defining Variables

All variables that are created are integer variables.

There are some system variables that have already been created by the software.
Cospace Rescue GUI - Adding statement

- Statement specifies action.
- A program is formed by one or more statements in sequence.
- Each statement will have a condition and an action associated with it.
- The statement type, conditions and actions associated with the new statement need to be specified.

Fatemeh Pahlevan Aghababa, Amirreza Kabiri
Cospace Rescue GUI - Adding statement

To add a new statement:

1. Select the statement/subroutine to which you want the statement to be added to and click the right mouse button and select ‘add a new statement within the bundle’ if this is the first statement within the bundle. Once is a statement in the bundle you can select ‘add a new statement’
2. Give the statement a meaningful name after which it will be added to the statement tree
Cospace Rescue GUI - Adding subroutines

Teams can add subroutines (also known as procedure or subprogram) which are referred to as ‘bundles’ of statements. They are a portion of code in the program which performs a specific task and can be used to decompose the program into smaller chunks. Using subroutines can allow team’s program to be less mess and more easy to debug and diagnose.
**Cospace Rescue GUI - Adding subroutines**

To add a subroutine:

- Select the previous statement/subroutine to which teams want to place the subroutine after and right click this statement/subroutine. Select add ‘add a new bundle within the bundle’ or ‘add a new bundle’ as appropriate.

- Assigning a meaningful name and then the subroutine is created and can be populated with more subroutines or statements.
Cospace Rescue GUI- Managing statement and subroutines

The order of statements and bundles on the tree is very important as the higher statements are up on the tree the higher their priority. To delete, move up or move down a statement right click on the statement or subroutine and choose from the appropriate actions:
Cospace Rescue GUI- Statement types

When a statement is added, the type of statement needs to be set.

The three type of statements that teams can use for different requirements are a super statement, non-interrupt statement or a default statement.

Fatemeh Pahlevan Aghababa, Amirreza Kabiri
Cospace Rescue GUI- Statement types

Default action – default statements have the lowest priority. Most statements in a project will be default statements.

Non-interrupt action – these statements have the same priority as default statements however when they are expected they will not be interrupted or terminated unless:

- The exit action condition is fulfilled
- The super action statement is executed
When the non-interrupt action is specified, it is necessary to define an exit condition for this action. Only when this specified exit condition is true will the statement be terminated. Projects can contain multiple non-interrupt action statements.

Super action – these statements have the highest priority. Once the condition for the statement is true it will be execute immediately, it will not wait for an action to complete, but will interrupt the action. A project can contain many super action statements.
To set more complicated conditions you can use to the advanced condition box. To do this, scroll to the bottom of the conditions section and select the ‘write code’ button.
Cospace Rescue GUI- Complicated conditions

This will launch the code editor. You can then enter advanced conditions using C code such as:
Cospace Rescue GUI - Controlling the motors

The two wheels can be controlled independently. Their direction and speed can be controlled. Each wheel can be set to a value between -5 and 5. Where +5 is maximum speed forward, -5 is maximum speed backwards and 0 is stop. The duration for the motor movement can be set. The minimum amount of time that the wheel action can be set for is 60ms.
Cospace Rescue GUI - LEDs

The LEDs can be set by setting the LED value to different values:

<table>
<thead>
<tr>
<th>LED value</th>
<th>LED status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>off</td>
</tr>
<tr>
<td>1</td>
<td>blinks</td>
</tr>
<tr>
<td>2</td>
<td>Is steady on</td>
</tr>
</tbody>
</table>
Cospace Rescue GUI - Key actions

For certain tasks, a key action must be specified by choosing a the action from a drop down menu:

- **FindObject** – select this when an object is detected
- **Deposit** – when the deposition area is found
- **Teleport** – select this to teleport from world 1 to world 2
- **GameEnd** – select this to end the game
Cospace Rescue GUI - MyState

The ‘mystate’ value determines whether you are in World 1 or World 2.
Setting this allows you to teleport from World 1 to World 2.
As it was mentioned before the output of the program written by GUI will be the C code which can be edited using C programming language. The primary teams really like to develop using the GUI and prefer to use this way while most of secondary teams are directly writing their codes in C and think that they GUI is not satisfiable for their strategies.

So it would be essential to support both types in order to satisfy both groups.

In the following slide, first we will discuss about compiling the code to dll and load it on robot and then we will study the code structure and it’s connections to server in details.
Cospace Rescue- Compiling C-Code

The final C code should be compiled to ‘.dll’ file using the GUI compile button.
Cospace Rescue- Loading C-Code on Virtual Robot

Fatemeh Pahlevan Aghababa, Amirreza Kabiri
Cospace Rescue- Loading C-Code on Real Robot

Team can choose to upload your own C-code to upload to the real robot.
Studying the Cospace Rescue Server

The ‘C’ code has connections to server using the ‘\_declspec(dllexport)’ which roles in exporting from a DLL. The following functions are exported from the compiled dll.

- **SetGameID**: This function set the ‘CurGame’ variable value. This variable is used to specify which world is currently running (world1 or world2).
- **GetGameID**: This function returns ‘CurGame’ variable value.
- **OnTimer**: The ‘OnTimer’ function is called once on each cycle and will call Game0() or Game1() function based on the ‘CurGame’ variable value.

Fatemeh Pahlevan Aghababa, Amirreza Kabiri
Studying the Cospace Rescue Server

- **GetDebugInfo**: This is used for debugging purposes by server and it will return all the defined variables between server and client which is changing by team’s codes.
- **IsGameEnd**: This function returns ‘bGameEnd’ boolean value which shows whether the game is ended or not.
- **GetTeamName**: It will return char* which is the team’s name.
- **GetCurAction**: This function returns the ‘CurAction’ variable value which is the number correspond to the current action.
- **GetTeleport**: It will return ‘Teleport’ variable value which is used for moving from first world to the second.
Studying the Cospace Rescue Server

- **SetSuperObj**: If the robot load a specific combination of object’s color in world1, it will receive super objects which has a high score in world2. This function will set ‘SuperObj_X’, ‘SuperObj_Y’, and ‘SuperObj_Num’ variable values which shows the location of super objects and the number of super objects in that location.

- **GetSuperObj**: This function returns ‘SuperObj_X’, ‘SuperObj_Y’, and ‘SuperObj_Num’ variable values.

- **GetCommand**: This is used for sending commands to server. A command contains ‘WheelLeft’, ‘WheelRight’, ‘LED_1’ and ‘MyState’ variable values.
Studying the Cospace Rescue Server

- **SetDataAI**: This function receives the following variable values from the server:
  - **US_Left**: Left Ultrasonic sensor
  - **US_Right**: Right Ultrasonic sensor
  - **US_Front**: Front Ultrasonic sensor
  - **CSLeft_R**: Color sensor left red value
  - **CSLeft_G**: Color sensor left green value
  - **CSLeft_B**: Color sensor left blue value
  - **CSRight_R**: Color sensor right red value
  - **CSRight_G**: Color sensor right green value
Studying the Cospace Rescue Server

- **CSRight_B**: Color sensor right blue value
- **PositionX**: Position X of robot in cartesian coordinate (only available in world2)
- **PositionY**: Position Y of robot in cartesian coordinate (only available in world2)
- **Compass**: Compass value of robot which is between 0 to 360 degree.
- **Time**: Which shows the passed time from the start of the simulation.

These variables will be set in team’s code using both direct C code programming or using an AI development GUI for creating states.
Studying the Cospace Rescue Server

In order to change the current cospace server to new server which is combination of ROS and Gazebo, we will need to prepare such functionalities. The new server should access these functions from `.dll`. Remaining on this `.dll` style have the following advantages:

- Teams can still use their previous code. If everything changes in one year, cospace may lose lots of teams.
- The final `.dll` would be compatible with the current available cospace rescue robot model.
- Rewriting a new stable GUI is time consuming and also children don't like changes that much.
The old cospace teams are familiar with ‘C’ programming code and they can improve their knowledge instead of learning another programming language.

Virtual Robot Committee can provide a direct interface for ros programming lator and secondary teams can directly write their codes in ROS.
Cospace Rescue New Server Suggestions

We suggest two different approaches for replacing the current connections to old server with connections to new server. The first suggestion is based on combination of ROS and Gazebo while the second suggestion only contains of Gazebo.
Cospace Rescue New Server Suggestion 1

- Creating the cospace robot and world model in Gazebo
- Creating a ROS repository which contains:
  - A package that connects to `.dll` file of the robot and set/get variable’s values and shows the scoreboard.
  - Packages like `robot_description` and `setup` from RoboCupap2018RVRL_Demo for spawning the robots and running the world model.
  - A launch file which should be `roslaunch` by clicking on the control panel start button.
Cospace Rescue New Server Suggestion 1
(recommendations)

1. Install ROS on Windows
   b. [https://janbernloehr.de/2017/06/10/ros-windows](https://janbernloehr.de/2017/06/10/ros-windows)

2. Install Gazebo on Windows

1. Porting Windows Dynamic Link Libraries to Linux
   a. [https://github.com/taviso/loadlibrary](https://github.com/taviso/loadlibrary)
Creating the cospace robot and world model in Gazebo

The ‘C’/‘C++’ program that connects to `.dll` file of the robot and set/get variable’s values and write on Gazebo socket.

Writing the Gazebo plugin that receives values and control the robot
This approach only need Gazebo installation on windows and knowledge about it’s C++ API and Plugin.

1. Install Gazebo on Windows

2. Gazebo C++ API
   a. http://gazebosim.org/api

3. Gazebo plugin
   b. http://gazebosim.org/tutorials/?tut=plugins_hello_world
Thanks for your attention!