Integration of an HTN Planning System with an Unmanned Surface Vehicle Simulator

***Abstract*—A current open challenge for Unmanned Surface Vehicles (USVs) is the development of a full capable autonomous guidance systems compliant to international marine rules of the road. This work present the integration of a guidance system compliant to the marine rules integrated to an USV 3D simulator. As proof of concept we present a scenario where the system must guide one simulated USV respecting the international marine rules.**

***Index Terms*—Unmanned Surface Vehicle, Hierarchical Task Network, Guidance System, 3D Simulator.**

I. Introduction

Direct collision between ships represents 60% of the accidents in sea and 56% of the collisions are caused by COLREGs[1] violation [2] [3]. The International Regulations for Preventing Collisions at Sea, 1972 (COLREGs), determine actions that must be followed by maritime pilots for preventing collisions in potential collision scenarios such as crossing, head-on and overtaking.

USVs are autonomous marine vessels that can execute multiple tasks in a variety of cluttered marine environments without human supervision [4][5]. USVs are characterized by small size, high mobility, and good hiding capability; they can be used in many marine applications, including oceanography, remote sensing, environmental monitoring, surveying, weapons delivery, mapping and navigation, along with providing communication support for unmanned underwater vehicles and general robotics research [5][6].

A current open challenge for USVs is the development of reliable, robust and full capable autonomous guidance systems. In general, guidance systems are responsible for both deliberative and reactive motion of a marine vehicle, being implemented respectively by global and local guidance systems. Global guidance systems are responsible for global path planning based on already known maps or nautical charts. Local guidance systems are responsible for local path planner based on real-time information about the environment. In general, local guidance systems actuates on collision avoidance.

Based on the fact that COLREGs currently depends on human interpretation due to its non-objectiveness, it is a huge challenge for an USV system to be totally autonomous and capable of avoiding collisions. This work present the integration of JSHOP2[7] and USV_SIM[8]. JSHOP2 is a Java platform to execute Hierarchical Task Network (HTN) planning and USV_SIM is a simulator of unmanned surface vehicles. The focus of this work is to integrate a planning system for guidance with a USV simulator. So, the idea is to combat the non-objectiveness of the COLREGs modeling the decision-making process of an USV using automated planning.

II. Background

In this section with brief describe key concepts for comprehension of the developed work.

A. HTN Planning and JSHOP2

In Hierarchical Task Network planning, the planning system begins with an initial state-of-the-world and with the objective of creating a plan to perform a set of tasks (abstract representations of things that need to be done). HTN planning is done by problem reduction: the planner recursively decomposes tasks into sub-tasks, stopping when it reaches primitive tasks that can be performed directly by planning operators. In order to tell the planner how to decompose non-primitive tasks into sub-tasks, it needs to have a set of methods, where each method is a schema for decomposing a particular kind of task into a set of sub-tasks (provided that some set of preconditions is satisfied). For each task, there may be more than one applicable method, and thus more than one way to decompose the task into sub-tasks. The planner may have to try several of these alternative decompositions before finding one that is solvable at a lower level. Unlike classical planning, HTN planning is Turing-complete [7].

JSHOP2 is the Java implementation of SHOP2 (Simple Hierarchical Ordered Planner). SHOP2 is a domain-independent automated-planning system. It is based on ordered task decomposition, which is a type of Hierarchical Task Network planning. So, JSHOP2 is a modified version of HTN planning that involves planning for tasks in the same order that they will later be executed. The whole task-network presented in Figure 1 were implemented using JSHOP2.

The main task is the ”Sail” task. From the ”Sail” task three possibilities are tested: 1 - USV has already finished its mission (arrived desired location); 2 - USV has not already finished its mission and there is no possible collision situation detected; and 3 - USV has not already finished its mission and a possible collision situation was detected.

The case where USV has already finished its mission leads to no task. The case where the USV has not already finished its mission and there is no possible collision situation detected leads to: 1 - run the collision detection system. If any possible collision is detected, the USV must let the local guidance system take control; 2 - change current position. If no possible collision is detected the navigation system must keep going toward the target localization. Otherwise, it must let the local guidance system take control; and 3 - Start Sail task again.

The case where USV has not already finished its mission and a possible collision situation was detected leads to: 1 - Identify what type of collision was detected according to the

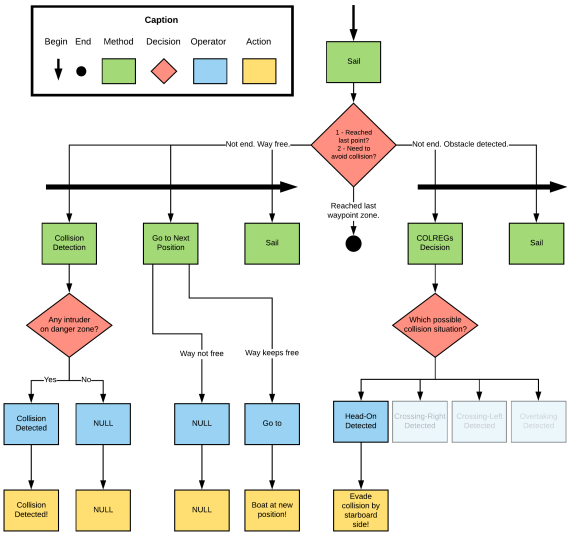

Fig. 1. Task Network Diagram

well-known situations described by the COLREGs and then apply the corresponding action, and 2 - Start "Sail" task again.

B. USV_SIM Simulator

The USV_SIM simulator is being developed by the Autonomous System Lab. from PUCRS[1], Brazil. The simulator uses a combination of multiple physics packages to build a test environment for Unmanned Surface Vehicle. It contains multiple robot models such as propelled boats(rudder boat, differential boat, air-boat) and sailboat. Boats are affected by waves, wind and water currents. All those features allow disturbing the movement of boats in a realistic way. Figure 2 shows the cited four boats on the USV_SIM simulator.

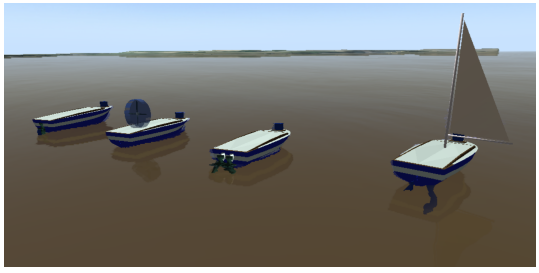

Fig. 2. USV_SIM Boats

III. Integrated System

In this section, we present the architecture of the fully integrated system. Figure 3 shows the main components of the integrated system.

USVs simulated on USV_SIM can be controlled using Robot Operating System (ROS) [9].

From one side, the USV_GUIDER [10] is responsible for interpreting the plan generated by the JSHOP2 and translate

[1]Pontifical Catholic University of Rio Grande do Sul

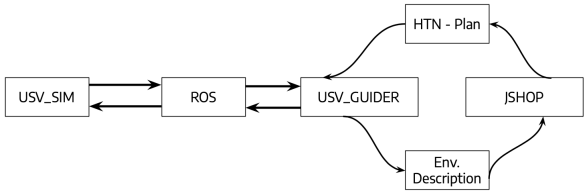

Fig. 3. Integration Arch

the defined actions into ROS commands. In the scope of this work, the ROS commands are linked to one boat. From the other side, the USV_GUIDER is responsible for translate conditions of the simulated environment to a description compliant to the JSHOP2 acceptable syntax. The conditions are captured through ROS topics. The USV_Guider was implemented in python using ROSlib library.

The JSHOP2 is originally developed to user interaction through Graphical User Interface (GUI). For this integration, we have done small changes to allow JSHOP2 to be executed from command line, this way our system is capable of performing systems call to run JSHOP2 when needed. Every execution of JSHOP2 reinterprets the generated environment description and generates a new plan.

IV. Experiment

For validation of the integrated system was performed the simulation of a situation where one air-boat encounter another anchored air-boat in a head-on situation. A boat following the COLREGs must avoid the possible collision changing direction to its starboard side. Figure 4 illustrates the head-on situation. Figure 5 The initial state of the experiment. In this experiment is expected that the controlled boat start sailing ahead and change its direction when a danger head-on situation is detected, as shown in the Figure 6. After evade the danger situation the controlled boat returns to its ahead sailing mode.

Fig. 4. Experiment - Head-on

A. Experiment Replication - General Instructions

For replication of the presented experiment 3 main repository are required: JSHOP2 fork [11], USV_SIM [12], and USV_GUIDER [10]. It is required to install the USV_SIM repository as a catkin package and run the repository as follow:

```
~/catkin_ws$ catkin_make_isolated --install
~/catkin_ws$ source
install_isolated/setup.bash
~/catkin_ws$ roslaunch usv_sim
airboat_scenario4_intruder.launch
parse:=true
~/catkin_ws$ roslaunch
```

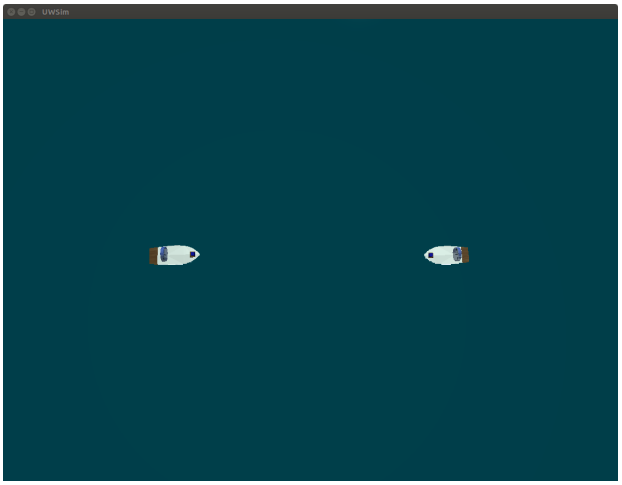

Fig. 5. Experiment - Head-on

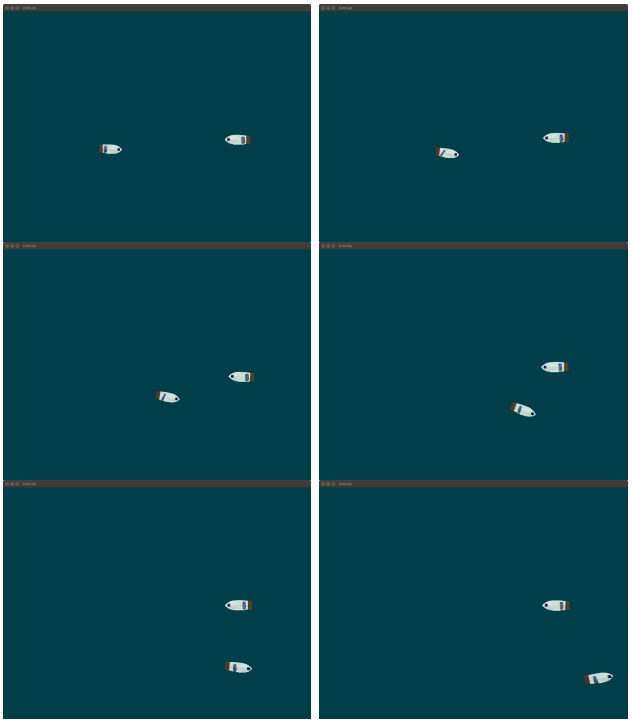

Fig. 6. USV motion from initial position to desired final location, avoiding collision with another ship

```
usv_sim airboat_scenario4_intruder.launch
parse:=false gui:=true
```

After simulation is opened its necessary to click start in the gazebo simulator.

For the USV_GUIDER execution is required to run in the top folder of the installation:

```
/USV_GuidanceSystem$ ./run.sh
```

Is expected that after the execution of the command above, the controlled boat start to move, as presented in the Figure 6.

V. Conclusion and Future Work

This study present a Hierarchical Task Network planning system integrated with a USV simulator. The experiment shows that we achieved a correct behavior treating a head-on dangerous situation. The simulated USV avoided the collision respecting requirements defined on Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs). Some of the limitation of the present work are:

1) The current local guidance system presented is capable of treat only one COLREGs situation. Future work must expand it to treat at least the four most common collision situations: crossing by right, crossing by left, overtaking, and encounter between different types of ships and USVs.
2) The developed simulation is capable of control the motion of only one ship at a time. Future work could expand it to control several ships at same time.
3) The current sail task is capable of only sailing the USV ahead, according to its own orientation. Future work could expand this functionality to be capable of more complex movement in the world such as follow waypoints.

Future work could also consider goal recognition to identify the actions of other ships.

References

[1] IMO, "Convention on the international regulations for preventing collisions at sea, 1972 (colregs)."

[2] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: An overview of developments and challenges," 2016.

[3] S. Campbell, W. Naeem, and G. W. Irwin, "A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres," *Annual Reviews in Control*, vol. 36, no. 2, pp. 267–283, 2012.

[4] J. E. Manley, "Unmanned surface vehicles, 15 years of development," in *OCEANS 2008*, vol. 2008-Supplement, pp. 1–4, Sept 2008.

[5] F. Thompson and D. Guihen, "Review of mission planning for autonomous marine vehicle fleets," *Journal of Field Robotics*, no. December 2017, 2018.

[6] W. Naeem, G. W. Irwin, and A. Yang, "Colregs-based collision avoidance strategies for unmanned surface vehicles," *Mechatronics*, vol. 22, no. 6, pp. 669 – 678, 2012. Special Issue on Intelligent Mechatronics (LSMS2010 & ICSEE2010).

[7] D. Nau, T. C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "SHOP2: An HTN planning system," *Journal of Artificial Intelligence Research*, 2003.

[8] A. Amory, D. Henrique, L. M. Gonalves, M. Paravisi, and V. A. M. Jorge, "Simulated environment for unmanned surface vehicles." https://github.com/disaster-robotics-proalertas/usv_sim_lsa, 2018.

[9] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[10] D. A. Jurak., "Python usv_guider." https://github.com/Unmanned-Surface-Vehicle/USV_GuidanceSystem, 2018.

[11] D. A. Jurak., "Jshop2 fork." https://github.com/Automated-Planning/jshop2, 2018.

[12] D. A. Jurak., "Usv_sim fork." https://github.com/Unmanned-Surface-Vehicle/usv_sim_lsa, 2018.