**The Robot Operating System (ROS) in Simulation, Control and Navigation of Marine Vehicles**

**Đ. Nađ**, *Faculty of Electrical Engineering and Computing, University of Zagreb*

***A. Vasilijević****, Faculty of Electrical Engineering and Computing, University of Zagreb*

***N. Stilinović****, Faculty of Electrical Engineering and Computing, University of Zagreb*

***N. Mišković****, Faculty of Electrical Engineering and Computing, University of Zagreb*

***Z. Vukić****, Faculty of Electrical Engineering and Computing, University of Zagreb*

**ABSTRACT:**

This paper presents the application of open-source software for simulation, control and navigation of marine vehicles. The Robot Operating System, developed in 2007 at Stanford, has gained increased popularity in mobile robotics. Lately its presence in marine systems is becoming more pronounced. We describe the application of this distributed framework to simulation and real-life operation of the Cooperative Autonomous Robotic Towing System (CART). The combination of the Underwater Simulator and ROS enable visualization and testing options out-of-the-box and reduce required man-effort when developing control and navigation systems.

Further we show the development process and system components of the CART vehicle. The different control approaches are analyzed and it is shown how with cooperative action vehicles can aid in ship-towing operations.

***Keywords:*** *marine systems, ROS, robot operating system, simulation, control, navigation*

Robot Operating System (ROS) u simulaciji, regulaciji i navigaciji pomorskih vozila

***Đ. Nađ****, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu*

***A. Vasilijević****, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu*

***N. Stilinović****, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu*

***N. Mišković****, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu*

***Z. Vukić****, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu*

**SAŽETAK:**

Rad prikazuje primjenu programske podrške otvorenog koda u svrhu simulacije, regulacije i navigacije pomorskih vozila. Robot Operating System, razvijen 2007 na Sveučilištu Stanford, znatno je zastupljen u mobilnoj robotici. U posljednje vrijeme primjena mu se proširila na podvodnu i pomorsku robotiku. Opisujemo primjenu ovog raspodijeljenog sustava na autonomni kooperativni sustav za asistenciju pri vuči brodova. Kombinacija podvodnog simulatora i ROS-a omogućuju vizualizaciju i testiranje sustava uz manje razvojne napore.

Opisujemo komponente kooperativnog autonomnog vozila za asistenciju pri vuči brodova. Obrađuju se iskorišteni pristupi upravljanjua i pokazujemo kako uz kooperaciju vozila mogu asistirati pri vuči brodova.

***Ključne riječi:*** *pomorski sustavi, ROS, robot operating system, simulacije, regulacij, navigacija*

1. **INTRODUCTION**

Marine research is becoming more interesting due to economic, scientific and military factors. With it the utilization of marine robots is increasing. Remotely operated underwater vehicles (ROVs) as well as autonomous underwater vehicles (AUVs) are interesting for their ability to work in depth beyond the reach of human divers. While ROVs are still mostly operated manually, the AUVs have to handle different controllers, behaviours and make decisions to accomplish the requested mission in the unknown underwater environment.

Vehicle autonomy is based on a complex system with many interacting modules. However, testing these modules is expensive and possibly impossible to perform in the sea before the whole system is operational. Hardware in the loop simulations can reduce the development cost. Later on we will analyze the existing tools in mobile robotics that offer easy simulation of underwater vehicles. The tools are developed around a communication infrastructure named the Robot Operating System (ROS).

The Robot Operating System (ROS) predecessor, named Switchyard, was developed by the Stanford AI lab (SAIL) in 2007 as support for the Robot STAIR (1). Continued development at Willow Garage has resulted in the ROS framework that is used today, (2). ROS provides operating system services such as hardware abstraction, low-level device control, message-passing between processes, and package management. ROS by itself is a soft-real time environment however integration with real-time operating systems is available. The framework is well established in land robotics and has a broad range of supported platforms and sensors. Lately, the emerging use of the framework in underwater vehicle competitions like SAUC-E triggered higher usage in underwater robotics. It provides a platform for connecting application (ROS) nodes in a subscribe-publish manner. This inter-process communication infrastructure is well suited for modular development. Parts of the AUV control and navigation architecture are developed as *ROS nodes*, a self standing unit that has predefined inputs and outputs. In a sense a node is a black-box for other nodes, offering only a small interface for interaction. Exactly how nodes connect and interchange data in the background is handled by ROS. We will show the benefits of this approach in the later sections.

Cooperative Autonomous Robotic Towing System (CART) concept is based on the idea of using cooperative unmanned to improve the safety and feasibility of salvage operations by increasing the distance between the towing tugboat, or salvage vessel, and the distressed ship. This can be done by combining the use of a pick-up buoy that is able to maneuver in order to keep a safe position during the operations, deployed at sea by the crew of the distressed ship, with an unmanned semi-submersible vehicle, remotely controlled or supervised, performing the recovery task. The semi-submersible vehicle, which is connected to the tugboat with a cable, recovers the pick-up buoy by knotting the cable around it with a suitable maneuver (see Figure 1).

|  |
| --- |
| cart_op_procedures.png |
| **Figure 1. CART operating procedure** |

The final system consisted of both semi-submersible vehicles (B-ART and C-ART) that communicated over a common remote control station (RCS). Both vehicles where equipped with a GPS and reported their position the RCS. Although the communication is established through the RCS, the control and navigation algorithms where running independently on the vehicles.

The article will present the used and developed tools for the simulation, integration and testing of the described CART system. The next section will describe in detail the derived dynamic model for simulation and the implementation of the simulator in the ROS framework. Implementation of the control and navigation system is described in section 3. Finally we present the real system and field results from harbor and sea testing performed during several occasions.

1. **SIMULATION**

Ability to simulate the system before real-life application is a prerequisite for fast and safe development of control and navigation algorithms. It offers the possibility to test algorithms and remove theoretical and/or implementation errors. We will show that by separating the control and navigation system into modules and decoupling them from vehicle definitions we allow the use of same controller and navigation algorithms for the simulator and real-vehicle. This extends the overlap between simulation and real-life and reduces redundancy. This approach is commonly known as Hardware-In-the-Loop (HIL) simulation.

* 1. **Notation**

We start by introducing the notation that will be used through-out the rest of the article. It is known that position and orientation of rigid bodies are determined by six independent coordinates. We define these six coordinates in table 1. For motion analysis in six degrees of freedom (DoF) we also define linear and angular velocities.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DoF** | **Description** | **Forces and moments** | **Linear and angular velocities** | **Positions and angles** |
| 1 | surge  (forward motion) | X | *u* | *x* |
| 2 | sway  (lateral motion) | Y | *v* | *y* |
| 3 | heave  (vertical motion) | Z | *w* | *z* |
| 4 | roll  (rotation about forward axis) | K | *p* | *Φ* |
| 5 | pitch  (rotation about lateral axis | M | *q* | *θ* |
| 6 | yaw  (rotation about vertical axis) | N | *r* | *ψ* |
|  |  |  |  |  |
| **Table 1: Rigid body degrees of freedom and their SNAME notation** (3) | | | | |

Based on the table above we define three vectors **η** = [x y z φ θ ψ], **ν** = [u v w p q r], **τ** = [X Y Z K M N] as vehicle position and orientation state vector in the Earth-fixed frame, the state vector of linear and angular velocities in the body-fixed frame and the forces and moments acting on the vehicle in the body-fixed frame, respectively. The Earth-fixed frame is considered as a North-East-Down frame of reference, while the body fixed frame is equivalently a forward-lateral-down frame.

The defined vectors and values fully describe the general rigid body 6 degree of freedoms in a 3D space. Underwater vehicles that are used by the CART project can be considered as rigid bodies if we avoid the influences of the cable. Although the cable influence is present in higher sea-state we can model the cable as a complex disturbance separately and not directly as part of the vehicle.

* 1. **Rigid-body model**

Vehicle movement in six DoF can be expressed with the following equation of motion, (4):

(1)

(2)

where , and are defined in the last section. External disturbances and forces are represented with . represents the rigid-body system inertia matrix and is the added mass term introduced by hydrodynamic effects acting on the vehicle. The vehicle Coriolis and centripetal forces are denoted by while the effects due to added mass terms are noted . Linear and quadratic damping terms are defined in , while the vector represents resulting restoring forces in presence of gravity and buoyancy.

These elements in equation 1 define the basic relation between the forces and speeds in the body fixed frame. Vehicle kinematics in equation 2 describes further the relation between body fixed speeds and the inertial reference frame. We define the Jacobian transformation as:

(3)

where and are the rotation matrix for linear and angular velocities, respectively. The matrices are defined in (4), and depend on the current orientation in the inertial frame.

* 1. **Simulator node**

The rigid-body model, defined in the last section, is used in the simulator implementation inside ROS with model parameters either identified on the real vehicle using Identification by self-oscillations (5). The low-level controllers output the vector which is the input of the ROS simulation node. Using equation 1-3 the vehicle model is propagated at a sampling frequency 10 times higher than the main control loop sampling frequency. The usual sampling frequency used for controlling the CART vehicles is 10Hz. The simulation model outputs noisy measurements to simulate sensors measurements like GPS, IMU, etc. However, to provide better sensor simulation we separate the sensor simulations into individual ROS nodes. This approach offers the ability to interchange different simulated devices without changing the simulation node implementation. This allows us to use extended sensor simulation and visualization nodes that are part of the UWSim suite, described in the next section. Later on, during in-field experimentation the distributed approach allowed us to run simplified simulation on the CART computer that is not capable of running the heavy UWSim simulation suite.

**2.3 Visualization**

Visualization is an important aspect for testing as provides feedback about the system for users that are unfamiliar with the inner workings, parameters and state transitions. It allows supplying the software to the end user for testing before the real-life implementation exists. We chose the visualization that is part of the UWSim system. The suite offers modeling of 3D environments and objects. Based on the interaction between models and the 3D environment in sensors are simulated that can be used by the underlying simulation software. Meanwhile, everything is displayed in a 3D rendered scene as shown in figure 2.

|  |  |
| --- | --- |
| uwsim_cart.png | uwsim_circle.png |
| **Figure 2. Circle mode with simulated measurement noise** | |

1. **CONTROL AND NAVIGATION**

Equations 1-3 describe a full six DoF coupled rigid body model. However, our vehicles are semi-submersible, meaning they operate on the surface or near the surface depending on the sea-state. Due to this we control only the surge and yaw degree of freedom. Heave is not controlled and due to under actuation we cannot control sway, roll or pitch. We can additionally simplify the model by uncoupling the remaining surge and yaw degrees of freedom.

The vehicle model described in section 2 is non-linear and coupled, which makes it impractical for control design. We can simplify the model by making the following assumptions:

• Vessel dynamics is uncoupled, i.e. coupled added mass terms are negligible, the center of gravity and inertia are the same, roll and pitch motion are negligible. This assumption is valid for small remotely operated vehicles. As a consequence of these simplifications the total Coriolis and centripetal matrix vanishes for equation 1 and restoring forces influence only the heave degree of freedom.

• Drag matrix is diagonal and each term can be approximated with a first order speed dependant term.

These simplifications lead to one, generalized, uncoupled, nonlinear dynamics equation that describes all degrees of freedom separately and it is given by:

(4)

where is a single degree of freedom, and other elements represent the diagonal element of matrices defined in section 2. Restoring forces are included in when is the heave degree of freedom.

**3.1 Control layout**

|  |
| --- |
| cart_control.png |
| **Figure 3. The control layout of the C-ART vehicle** |

The control layout is structure into a cascade of speed controllers and higher-level controllers as shown in Figure 3. The speed controllers control the vehicle dynamics making the higher-level controllers independent of the vehicle parameters. The speed controllers are auto-tuned PI controllers with a feed-forward action:

(5)

where we can take the feed forward gain to be . Using the reference speed is more robust since measurement noise or estimation errors are not additionally introduced via the feed forward. More details about this structure can be found in (6).

The higher level controllers needed for the maneuvers are the line-following (course) controller for B-ART maneuvers and the virtual-target path following controller for C-ART maneuvers.

The line-following (course) controller is an extended PD controller described in (7). It uses the current and desired position to establish the desired course to follow. The desired forward speed is constant and only the yaw-rate is commanded to decrease the lateral error introduced by currents or wind. This is used by the B-ART vehicle to keep the desired course away from the distressed ship in bad weather, as shown in figure 4.

|  |  |
| --- | --- |
| bart_launch.png | rope_tying.png |
| **Figure 4. The B-ART (left) and C-ART (right) maneuvers** | |

The knotting is done by performing a circular motion around the B-ART target. For that we generate a circular path around the known or estimated B-ART location which is possibly moving. The path is tracked by the path following algorithm that uses the virtual-target approach as described in (8). This is a Lyapunov-based controller that generates the commanded yaw rate as:

(6)

where is the curvature constant of the circle. The gain is selected accordingly to the desired performance. The variable is the speed of the virtual target along the path. It depends on the vehicle forward speed and approach angle. Finally, the approach angle is modeled as continuous function of lateral distance between the vehicle and the virtual target in the virtual target reference frame, often called the *Serret-Frenet* frame.

**3.2 Navigation layout**

The navigation layout consists of two parts: a) the GPS+Imu hardware and b) the EKF filter with the dynamic model. The separately developed filtering of the Inertial Measurement Unit in combination with the GPS measurements allowed the in-house creation of a configurable navigation sensor. Cheaper of the shelf sensors could be use and integrated in hardware to provide a quality positional and inertial sensor.

The Kalman Filter was implemented on the PIC microcontroller that fused the measurements via a kinematic model. The design purposely excluded knowledge about the dynamic model of the vehicle to allow using the sensor on vehicles whose parameters are not know or are hard to identify.

Output of this navigation module is the input into the ROS software navigation module based around the Extended Kalman Filter. The ROS navigation node uses the thrust inputs and the dynamic model described in section 2 to estimate vehicle speeds and positions. The estimated speeds are used by the controllers. The benefit of this filter is that we are able to use our cascade control layout even in cases where speeds are not directly measured. The C-ART and B-ART systems are examples of such systems that have no sensor to measure surge. The EKF filter is designed to enable incorporation of additional asynchronous measurements from speed sensors without changing structure. This allows automatically fusing all available knowledge about the vehicle state to increase the final navigation precision without setting the requirement on the sensor suite.

The control and navigation performance was evaluated in real-life condition during the system validation in harbor and on the sea. The system validation was carried out in Tallinn, Estonia by the end-user PKL.

1. **CONCLUSION**

The paper has offered an overview of the simulation infrastructure available in the Robot Operating System framework. The UWSim is predominately envisioned as an underwater vehicle simulator; however, mobile robot simulators exist as well but where not covered due to size limitations. The reader is advised to look-up Gazebo and MORSE for mobile and aerial simulators.

We have also described components of the C-ART (autonomous semi-submersible vehicle) and B-ART (autonomous semi-submersible buoy) control and navigation. Since all algorithms were wrapped in ROS nodes they can exists individually and allow a puzzle like assembly of alternative control and navigation architectures for future vehicles. This aspect is very useful since new development does not start with new monolithic structures but rather subtracts or adds new nodes to the already existing and testing core system. The concept was already proven by extending the architecture to a diver tracking surface platform were the navigation and control algorithms of the C-ART system were fully reused.

##### **REFERENCES**

1. **Quigley, Morgan, Berger, Eric and Ng, Andrew Y.** *STAIR: Hardware and Software Architecture.* 2007.

2. *ROS: an open-source Robot Operating System.* **Quigley, Morgan, et al.** 2009, ICRA Workshop on Open Source Software.

3. **Thompson, R. Lee and al., et.** *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid.* s.l. : The Society of Naval Architects and Marine Engineers, 1950.

4. **Fossen, Thor I.** *Guidance and Control of Ocean Vehicles.* s.l. : John Wiley & Sons, 1994.

5. *Fast In-field Identification of Unmanned Marine Vehicles.* **Mišković, Nikola, et al.** 1, s.l. : John Wiley and Sons Ltd., 2011, Vol. 28.

6. *Guidance and control of an overactuated autonomous surface platform for diver tracking.* **Misković, N., et al.** 2013. Control & Automation (MED), 2013 21st Mediterranean Conference on.

7. *3D Line Following for Unmanned Underwater Vehicles.* **Mišković, Nikola, Nađ, Đula and Vukić, Zoran.** 2, s.l. : Brodogradnja : Äasopis brodogradnje i brodograÄ‘evne industrije, 2010, Vol. 61.

8. *Virtual target algorithm in cooperative control of marine vessels.* **Triska, Z., et al.** 2012. MIPRO, 2012 Proceedings of the 35th International Convention.

##### 