Model-based path planning and obstacle avoidance architecture for a twin jet Unmanned Surface Vessel

Jose Villa
Mechatronics Research Group
Tampere University (TUNI)
Tampere, Finland
jose.villaescusol@tuni.fi

Jussi Aaltonen
Mechatronics Research Group
Tampere University (TUNI)
Tampere, Finland
jussi.aaltonen@tuni.fi

Kari T. Koskinen
Mechatronics Research Group
Tampere University (TUNI)
Tampere, Finland
kari.koskinen@tuni.fi

***Abstract*—The aim of this study is to implement a model-based control architecture for a twin jet Unmanned Surface Vessel (USV) using the line-of-sight (LOS) path algorithm for straight-line. The modeling and design of the motion control system are studied using MATLAB-Simulink and the real-time implementation is done by using Robot Operating System (ROS). To simplify the real-time implementation and thus, programming files, the necessary code generation is done by generating a Standalone ROS Node from Simulink software, making the process easier for researchers with non-advanced programming skills. Simulation and field test results are shown to validate the model-based control architecture.**

***Index Terms*—model-based, path following, simulation, implementation, USV, MATLAB-Simulink, ROS**

I. Introduction

In recent years, the research involving autonomous systems has increased in many different scenarios, including the unmanned surface vessels (USV) in lakes or sea areas. Research on USV has included trajectory tracking and path following control problems. Furthermore, several studies include different guidance laws for path following and collision avoidance [1], [2]. Those studies require an extensive code programming process which is time-consuming and involves many steps before implementing the desired algorithm successfully. There are some software tools which allows the generation of C and C++ code, such as MATLAB/Simulink. The modeling and design of the motion control system can be studied using MATLAB/Simulink, and the real-time implementation can be done by using Robot Operating System (ROS) [3].

In this work, a model-based control architecture for a USV is studied following the USV design used in a twin jet research vessel. Furthermore, a simple path following algorithm is described. In addition to that, all schematics for simulation and implementation are shown in MATLAB/Simulink to create the necessary model subsystems in order to generate and build a standalone ROS node for real-time operation of the USV.

II. USV Control architecture

A. Description of the USV platform

The Research vessel used in this study is an aluminum hull with thrust vectoring twin water-jet. It uses two marine diesel engine with 170 kW of rated power and an intelligent operation (IO) control system (AJ IO [4]). Furthermore, by using a twin jet configuration, the USV has an ultimate maneuvering accuracy as it can move in all directions without bow thrusters.

The Research Vessel used in this study contains a Linux CPU as a ROS Master computer, which is connected to the rest of instrumentation (including GPS Compass and IO system). The necessary control commands are sent by a computer running MATLAB/Simulink which is a ROS node in the system.

B. Path following algorithm

The aim of a path following algorithm is to reach every point of a predefined path independent of time. Line-of-sight (LOS) guidance is used in this study [5]. A LOS vector from the surface vessel to the next way-point or a point on the path between two way-points can be used for heading control. There are no temporal constraints such as the representation of obstacles and other positional constraints in this study. Furthermore, the switching mechanism is defined as a circle of acceptance for surface vessels [5].

C. Control system

The complete control system is based on two different PID controllers for the surge and heading control, and their parameters are obtained by using Rapid Control Prototyping (RCP) during the field tests. Surge control keeps the surface vessel in a predefined constant speed. LOS algorithm sends heading commands to the autopilot to accomplish the predefined path. Furthermore, the feedback loop includes a cascaded Low-Pass and Notch Filtering to reduce motions induced by waves [5]. By using this cascade filter, small waves have been suppressed getting constant outputs (surge and yaw) in the controller.

D. Block diagram (Simulation)

The block diagram for Simulation is based on four main subsystems (Fig. 1). Navigation subsystem contains all LOS path following controller equations. The control block contains both PID controller and cascade filter. The boat model has joystick parameters as input. The conversion from forces to joystick commands is included using tables provided by the jet manufacturer. The boat model subsystem contains all dynamic equations to get the surge and heading for the simulator.

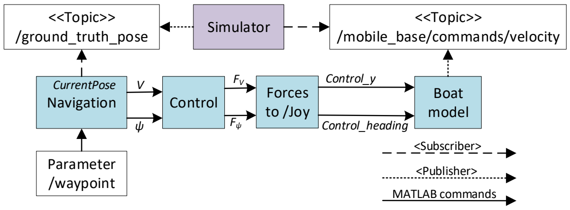


Fig. 1. ROS schematic for Navigation and Control (Simulation)

E. Block diagram (Implementation)

The implementation schematic can be seen in Fig. 2, showing the Simulink model with respective subsystems and both GPS Compass and IO system from the USV. The sensor data is obtained from the GPS Compass, which uses WGS84 (Latitude, Longitude) instead of a projected coordinate system, such as UTM (Universal Transverse Mercator) or ETRS-TM35FIN coordinate systems. Conversion from WGS84 to ETRS-TM35FIN system uses an already defined function [6].

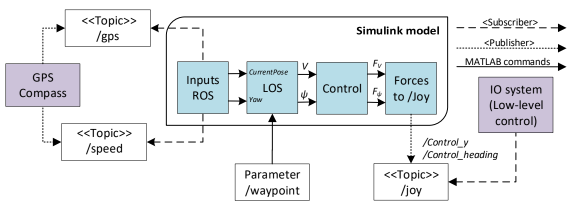


Fig. 2. Simulink schematic for Navigation and Control (Implementation)

The test model is based on the Robotic System Toolbox from MATLAB/Simulink [7]. In order to define the necessary ROS topics, a standalone ROS node is generated and built from the Simulink model [8], avoiding all significant programming issues and the coding time that every ROS package involves.

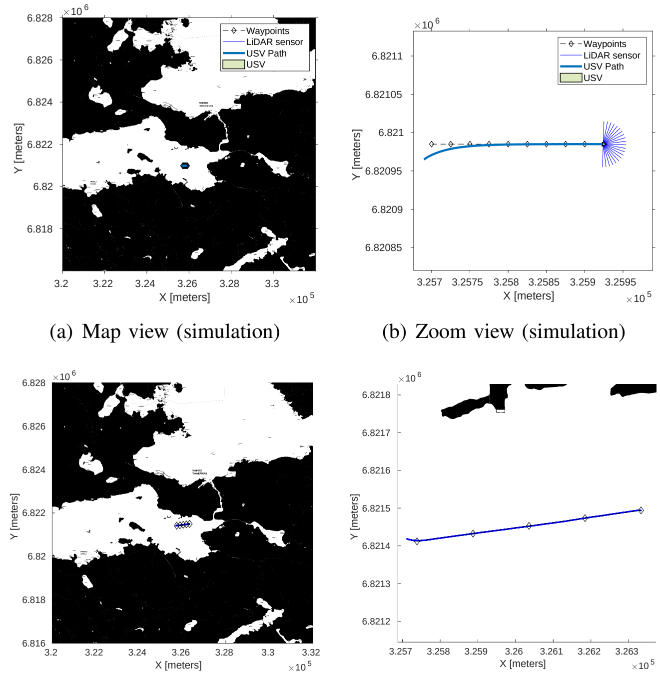
III. Simulation and Field-Test Results

Using the Simulink model shown in subsection II-D, Fig. 3(b) shows the results of the path-following controller. Plots are simulated using ETRS-TM35FIN coordinate system. The simulation has been done adding an offset of (-10,-20) meters from the initial way-point as a starting point of the simulation.

The field test results from the block diagram described in subsection II-E are shown in Fig. 3(d). The blue line shows the GPS data recorded during the test and the black dotted line is the predefined path. The error between this predefined and the USV path is less than 1 meter (measured by the tangential distance to the path). This difference is due to environmental elements such as wind or wave drift forces, which compensation is not considered in this study.

IV. Conclusions

In conclusion, a model-based control architecture for a twin jet Unmanned Surface Vessel is implemented for real-time implementation. Due to the real-time implementation, it is not possible to use the same software for simulation and implementation (MATLAB/Simulink) and a second software



(a) Map view (simulation)

(b) Zoom view (simulation)

(c) Map view (field-test)

(d) Zoom view (field-test)

Fig. 3. Simulink-ROS: Simulation and field-test results

is included (ROS). To avoid programming issues and the related time consuming, this paper describes the schematic and connections to generate and built a standalone ROS node. This ROS node includes a simple straight-line path following algorithm (LOS controller). Both simulation and field test results are shown to confirm the control architecture.

Acknowledgment

This paper is based on the aColor project funded by Technology Industries of Finland Centennial and Jane & Aatos Erkko Foundations. The authors gratefully acknowledge the contributions of Sauli Virta from Alamarin-Jet Oy.

References

[1] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," *IFAC Proceedings Volumes*, vol. 36, no. 21, pp. 211–216, 2003.

[2] S. Moe and K. Y. Pettersen, "Set-based line-of-sight (los) path following with collision avoidance for underactuated unmanned surface vessel," in *Control and Automation (MED), 2016 24th Mediterranean Conference on*. IEEE, 2016, pp. 402–409.

[3] M. Quigley, K. Conley, B. 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*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[4] Alamarin-Jet Oy. (2018) Intelligent operation - AJ IO. [Online]. Available: https://www.alamarinjet.com/products-intelligent-operation

[5] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[6] O. Lammi. (2014) Coordinate system functions. Github Repository.

[7] The MathWorks, Inc. (2018) Design and test algorithms for robotics applications. [Online]. Available: https://se.mathworks.com/products/robotics.html

[8] ——. (2018) Generate a standalone ROS node from simulink. [Online]. Available: https://se.mathworks.com/help/robotics/examples/generate-a-standalone-ros-node-in-simulink.html