Development of a navigation system with high function and high reusability for Maritime RobotX Challenge 2018.

Shuhei Horiguchi ,Osaka Univ, Team Leader , Software Development
Ryodo Tanaka , Univ. of Tsukuba, Sub Team Leader, Software Development
Kazuki Yamada, Toyama College of Technology, Software Development
Ryo Nakabayashi, Univ of Tokyo, Software Development
Masaya Kataoka, Tier IV. Inc., Software Development
Hiroyuki Harada, Osaka Univ, Hardware Development
Ojiro Matsumoto, Osaka Univ, Hardware Development
Manami Maekawa, Osaka Pref Univ, Hardware Development
Kana Miyazawa, Osaka Pref Univ, Hardware Development
Suisei Wada, Yanmar Co. Ltd, Electrical Circit Development
Taisei Nishishita, Osaka Univ, Electrical Circit Development
Natsuki Akazawa, Toyama College of Technology, Project Management
Tamaki Kumauchi, Osaka Univ, Project Management

2018/11/11

Abstract - OUXT Polaris has been developing of an autonomous navigation system through participating in Maritime RobotX Challenge 2014 and 2016.
In this document, we describe the advantages and disadvantages of the previous system and introduce the new system for 2018 designed based on further review.

1 Evaluation of the previous navigation systems

In 2014 we achieved fifth place in the competition. In this system, the Intel NUC and laptop computers were used as calculators. For sensors, we used the SC-30 lent by our sponsor's Furuno Electric and Hokuyo Automatic YVT-35LX. The network equipment malfunctioned due to thermal runaway during usage, and development of more easy-to-troubleshoot system seemed to be necessary.

In 2016, we redesigned the computer system, which improved the computer performance and software development efficiency by using a desktop computer. However, the power consumption was extreme, and advanced knowledge and experience was required for the power supply design. So, we added more equipment, inverters, and increased the weight of the hull. Also, since there was only one computer, we had to procure replacement parts at the site in case of failures. There were still other problems. Lidar used for obstacle detection revealed serious lack of the amount of point cloud data that can be detected and its accuracy. And, we discovered that the hull could be swept by very strong waves at Hawaii.

Considering these reviews, this year we have been working on the autonomous navigation system with high component reusability, while keeping high calculation performance and advanced controllability.

2 Development policy of a navigation system for Maritime RobotX Challenge 2018

The hardware for Maritime RobotX Challenge 2018 was reformed under the broad policy below.

- Construction of distributed control system using calculators of the same type.

- Elimination of inverters by the unification of the power system.
- Sophisticated recognition of surroundings using additional sensors.
- Improvement of stability with low center of gravity by additional floats.
- Reconsideration of sensors arrangement and type.
- Realization of the lateral movement by sophisticated driving mechanism.

In addition, we reformed the software based on the following policy.

- System development by the gazebo cooperation and the simulator.
- Open-source system development with ROS which takes reusability into account.
- Increased calculation speed by the sensor fusion and adoption of deep learning.

3 Development of electrical hardware

3.1 Construction of distributed control system by calculators of the same type

Jetson TX2 was adopted as a hull based on review of the last competition. Although this built-in computer is the size of a business card, it is possible to run Tiny yolo with nvidia tegra GPU at about 10 FPS.

Since power consumption is also extremely small, it contributed greatly to power saving and the lighter weight of the hull. By harmonizing all hull calculators with Jetson TX 2, it became possible to exchange immediately when a breakdown occurred, and we made effective use of valuable working time. With the adoption of the distributed control system, we decided to further utilize the ROS that we are using from the last competition.

3.2 Elimination of inverters by the unification of the electrical system

Computers and sensors adopted for control of the hull are unified with those of 12V and 5V, with only the voltage of the driving system at 24V. Therefore, there are three power systems in the hull. Figure.1 shows the connection diagram of the circuits, and there are three power systems that mentioned above.

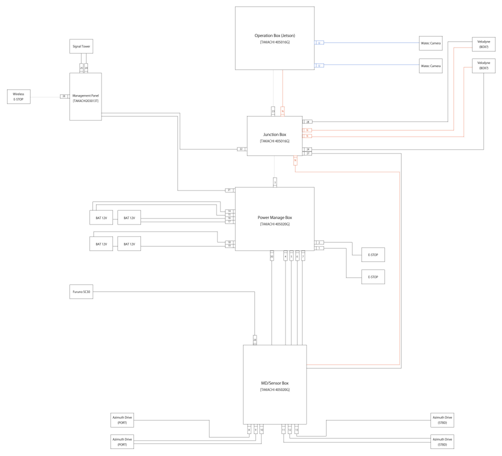

Figure 1: The connection diagram

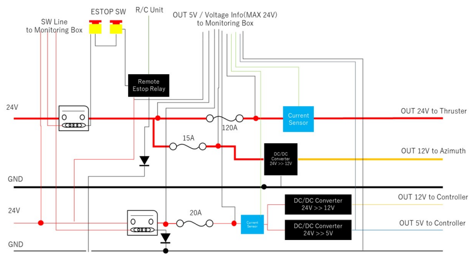

Figure 2: The circuit connection

We separated electronic components into three segments for power supplying, computing, and motor driving. We developed five main boxes which are connected by cables. All boxes are waterproof, and they can be separated. This makes it easy to build up on the vessel, and they can also be easily removed to fix failed components. Next we will introduce some significant features of some boxes. These boxes have three main functions: voltage converting, switching for safety, and distributing. The circuits for the motor and the others are completely separated for noise reduction. This time, the control panel includes switches and indicators that are separated from Power Management Box, so we can put switches mostly anywhere on the vessel. Also, the other devices like sensors are not supplied power from this box directly, but via a distributor. This design helps to reduce the number of waterproof connectors on this box and makes the wiring simpler. Figure.2 shows the circuit connection.

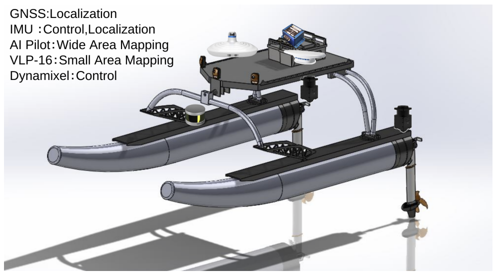
GNSS:Localization
IMU :Control,Localization
AI Pilot:Wide Area Mapping
VLP-16:Small Area Mapping
Dynamixel:Control

Figure 3: Sensor assembly

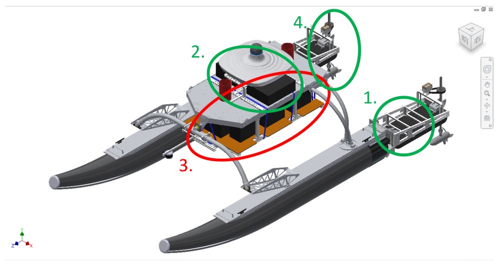

Figure 4: Hardware assembly

3.3 Sophisticated recognition of surroundings using additional sensors

We realize that we could not get enough information due to lack of sensors to recognize the surroundings.

Therefore, we extended the number of sensors largely for this competition that shown on Figure.3. Aipilot has a built-in LiDAR and a camera module which are necessary for running the autoware and the nvidia Drive PX2. Autoware is an all-in-one automatic operation module developed by TierIV.inc. We modified the specification of the AI Pilot and have been operating it as a module equipped with a Jetson TX2 which processes data of camera and VLP-16. We borrowed the AI Pilot from our sponsor Tier IV.inc. The VLP-16 installed in AIPilot is mainly used for detecting obstacles that are distant and to the stern. Another VLP-16 attached to the front of the hull is used for obstacle detection near the hull and ahead.

The IMU, Inertial Measurement Unit, is a sensor with gyro sensors, acceleration sensors and angular velocity sensors used for localization and speed control.

We borrowed the SC - 30 satellite compass from our sponsor Furuno Electric. Data of hull speed and localization output from the SC-30, and data output from the IMU are subjected to sensor fusion by a particle filter and self-position estimation is performed.

In order to deal with a task of fixed point keeping, we use servo motors for robot, ROBOTIS Dynamixel, so that we can change the angle of the thruster.

This improvement made the control system complex, but we could make hull move to just beside.

4 Development of the hardware mechanism

4.1 Improvement of stability by additional floats

We mount floats which make additional buoyant on the back of the full (No.1 part in Figure.4) to decrease the possibility of the full downfall. The structure materials of aluminum support the buoyant which occurs when the Styrofoam sinks in the sea. We also use GFRP(Glass Fiber-Reinforced-Plastics) not to damage the Styrofoam when it contacts the obstacles.

It makes sailing steady because it produce big buoyant though in spite of its lightness. In addition, we put the mounter of battery on the full body and improve the sailing stability than last competition because the center of gravity was too high.

4.2 Review of the sensors arrangement and type

The upper part of the hull is used for detecting the location information of the objects around it. In order to secure an adequate visual field, we arranged the sensor at height. We placed a stand to arrange Ai pilot upon the MD/Sensor Box and the Jetson Box (No.2 part in Figure.4) so that the hull won't be detected by the lider. In addition, we selected the dual lidar structure, which has another lider in front of the hull. As a result, we can get information about view of both wide field and detailed forword.

4.3 Stabilized navigation by lowering the center of gravity

At the last competition, our machine had a big problem with stability due to the high center of gravity caused by arranging batteries on the deck. To deal with the problem, we expanded the space for batteries and Power Management Box (No.3 part in Figure.4) and made center of gravity lower. This improvement helped our machine to run stably.

4.4 Realization of lateral movement by sophisticated driving mechanism

In order to increase the flexibility of movement of the hull, two propulsion mechanisms with a degree of freedom control in the yaw axis were mounted (No.4 part in Figure.4). An outboard electric motor (LACOMETA) with 86 pounds of thrust turns into the direction of the yaw axis using servo motors(XM540-W270, Dynamixel) and timing belt pulleys. With these additions, the hull can also move sideways to the hull.

5 Improvements to the Software System

The intent of OUXT Polaris from the beginning was to upload the source code of the system onto GitHub and then participate in the RobotX competition. The primary objective of open-sourcing the source code online was to reduce the barriers of such an undertaking for people who have interest in the Maritime RobotX Challenge and in the autonomous robotics community in general. This system was designed and implemented using the Robot Operating System (ROS), serving as a framework for robot software development.

5.1 System Development simulations through cooperation with gazebo

There are multiple simulators that can link with the ROS, but in our case, the link with the ROS was crucial, leading to the usage of gazebo. Originally, we developed a hull simulator named "ros_ship_packages" [1], but with the advent of the official simulation called VMRC, we extracted some functions from the package and moved to the current "robotx_packages"[2]. Inside of "robotx_packages", we built a simulator based on the VMRC source code. Figure.5 shows the demonstration of navigation. And videos were uploaded to the YouTube channel[3].

[1]https://github.com/OUXT-Polaris/ros_ship_packages

[2]https://github.com/OUXT-Polaris/robotx_packages

[3]https://www.youtube.com/watch?v=i68BraNP7Zo&amp;t=3s

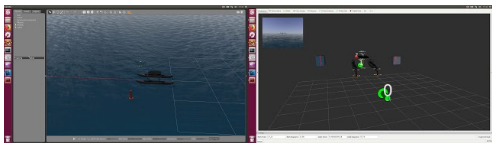

Figure 5: Navigation demonstration

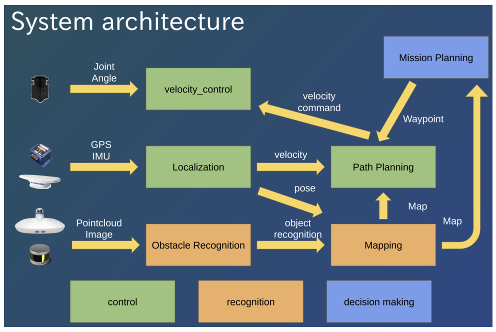

Figure 6: System architecture

5.2 Open Source Development with awareness to ROS reusability

The ROS has a feature that allows easy utilization of the software assets. It can easily divide the system using interprocess communication with TCP / IP communication, or use a different programming language (C ++, Python, Lisp, Rust, etc.) to exchange data with each other, When interprocess communication is performed, data is exchanged based on the message type. This message type can be expanded by its own definition, enabling extremely flexible operation. Furthermore, the robot simulators compatibility with gazebo is simple and easy , making it possible to speed up the development. Since all the packages developed this time are released as open source software, we believe that other teams can utilize OUXT Polaris's software assets and contribute to the sophistication of the competition. allowing easier reuse of the system.

This system is designed as a collection of modules as shown on Figure.6 Each of these modules cooperate to allow complicated autonomous navigation.

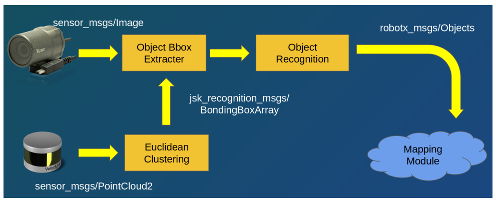

Figure 7: Obstacle recognition architecture.

5.3 Faster computational speeds concerning the adoption of deep learning and sensor fusion

In this system, deep learning is used for object recognition from images. Deep learning has been applied to a wide range of fields such as image recognition, translation, robot control and the likes. It has made remarkable results in recent years, especially in the field of image recognition. Through various competitions, it has been shown that the results from deep learning are far better than the recognition methods based on designs by conventional humans. Furthermore, in the past, it was mainly a method of recognizing objects appearing in images such as ImageNet, but recently, starting with Fast RCN, including Faster RCNN, Yolo by SSD and Joseph, and Faster-RCNN by Shaoqing. These new methods of object recognition utilizing deep learning are also appearing.

However, when designing an environmental object recognition system to be installed in this autonomous navigation system, it is necessary to measure the type of the object and the relative position of the hull. Since the computer to be used is a built in computer, calculation reduction of is required for quick recognition and positioning. Therefore, in the proposed environmental object recognition system, recognition of environmental objects is performed by combining 3D Point cloud processing and image processing by deep learning. The layout of the object recognition system is shown on Figure. 7

Euclidean clustering is performed on the data acquired from the 3D LiDAR. For the 3D LiDAR sensor, two VLP-16 [4] made by the Velodyne Lidar company were used. As part of the pre-processing of the Euclidean clustering, integration of point clouds acquired from the two VLP-16s are taken and the removal of point clouds obtained from the reflection on the ship are performed. The point cloud before pretreatment is shown in the Figure. 8, and the point cloud after pretreatment is shown in the Figure.

[4]https://velodynelidar.com/vlp-16.html

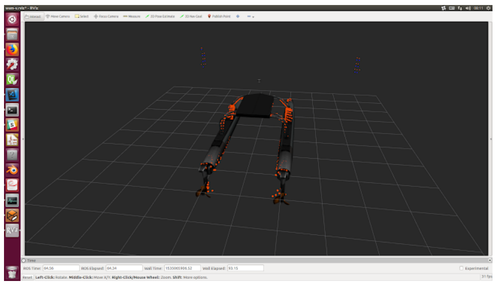

Figure 8: point cloud before pretreatment

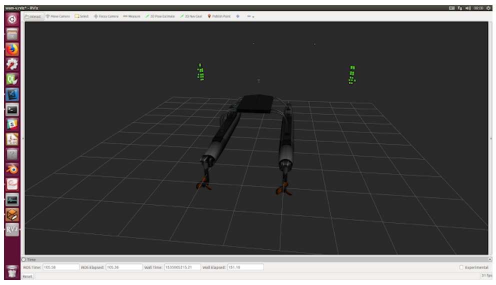

Figure 9: Point cloud after pretreatment

9.

The blue dot group in the Figure.8 is a point cloud obtained from a VLP-16 attached to the upper part of the hull, and the orange color group is a point cloud obtained from a VLP-16 attached near the surface of the water.

After eliminating the point cloud of the ship, Euclidean clustering is performed to detect surrounding objects. Euclidean clustering is an algorithm that classifies a point cloud into multiple clusters based on Euclidean distance. The Euclidean clustering algorithm of this system is based on PCL's Euclidean Cluster Extraction[5].

The result of Euclidean clustering is shown in the Figure. 10.

Furthermore, this object region extraction result is projected on the image plane of the camera to perform object recognition through deep learning. The image data obtained from the camera and the result of Euclidean clustering are combined to extract the object region from the camera image. First, the result of Euclidean clustering is transformed into the coordinate system as viewed from the camera. For the coordinate transformation, we

[5]http://pointclouds.org/documentation/tutorials/cluster_extraction.php

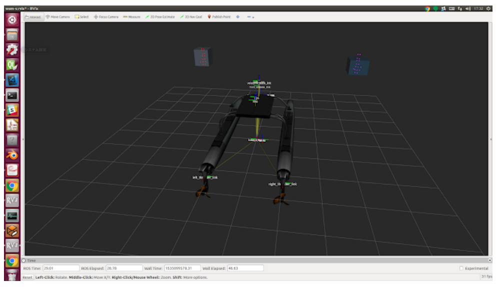

Figure 10: Euclidean Clustering result

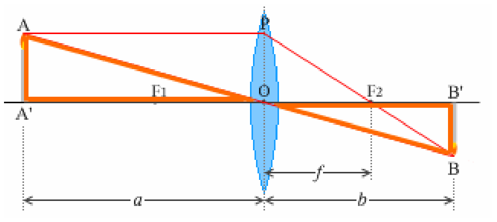

Figure 11: Pin hole model

used a coordinate transformation library frequently used in ROS called tf [6]. By using tf, it is possible to easily manage the coordinate system in the distributed control system. Concerning specific usage, implementation, etc.[7], this material explains in detail. Thereafter, the pinhole model is applied to the camera, and the position of a certain point in three dimensional space and the position of the point on the image are associated by the following algorithm. The pinhole model is a model of the simplest camera, as shown in the Figure. 11.

In the Figure.11, f is the focal length, a is the distance from the viewpoint to the lens, and b is the distance between the lens and the object. Apply the following algorithm to project the coordinates P (unit: meters) you want to project onto the point $P_i = [p_{xi}, p_{yi}]$ (unit: pixels) on the image plane.

1. Parameter f_h (viewing angle in the height direction), f_w (viewing angle in the lateral width direction) are defined, each unit is rad
2. Define the coordinates $P = [p_x, p_y, p_z]$ to be projected on the image plane
3. Define the coordinates $P_0 = [p_{x0}, p_{y0}, p_{z0}]$ where the camera exists

[6] `http://wiki.ros.org/en/tf`

[7] `https://www.slideshare.net/kojiterada5/tftf2`

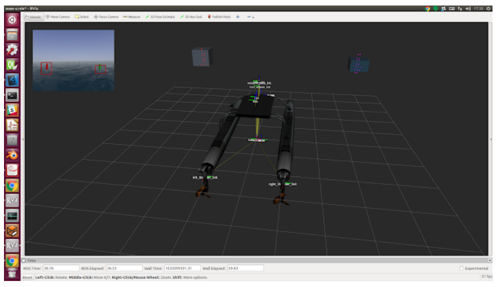

Figure 12: Result of the projection

4. Calculate a vector $v_0 = [v_{x0}, v_{y0}, v_{z0}]$ extended from P_0 toward the center of the image plane
5. Calculate the vector $v = [v_x, v_y, v_z]$ when looking at the coordinates P from the coordinates P_0
6. $pitch = atan2(v_z - v_{z0}, v_x - v_{x0})$
7. $yaw = atan2(v_y - v_{y0}, v_x - v_{x0})$
8. Define the number of pixels in the image height direction as h and the number of pixels in the horizontal width direction as w
9. $p_{xi} = w/2 - yaw * f_w * w$
10. $p_{yi} = h/2 - pitch * f_h * h$

At this time, the point P to be projected is selected from six rectangular planes making up the bounding box (rectangular parallelepiped) of the Euclidean clustering result, closest to the camera with respect to the normal distance as a reference, and the coordinates of the four end points constituting the plane. Based on the four points projected on the image plane thus obtained, an object region on the image is calculated as shown in the Figure.12.

Object recognition is executed for each object region obtained in this manner. Convolutional Neural Network (CNN) was used for this task. Figure.13 shows the procedure.

Neural nets with this structure can be represented by Keras in the following notation.

1. model = Sequential()
2. model.add(Conv2D(32, kernelsize=(3, 3), activation='relu', inputshape=inputshape))
3. model.add(Conv2D(64, (3, 3), activation='relu'))
4. model.add(MaxPooling2D(poolsize=(2, 2)))
5. model.add(Dropout(0.25))

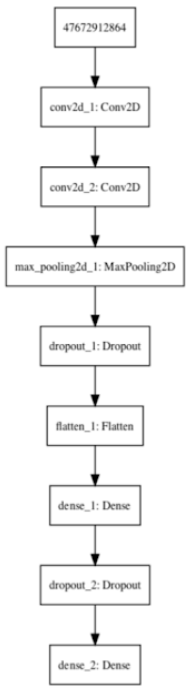

Figure 13: Procedure of the CNN

6. model.add(Flatten())
7. model.add(Dense(128, activation='relu'))
8. model.add(Dropout(0.5))
9. model.add(Dense(4, activaton='softmax'))

In addition, this system, using the tensorrt library for inference, can infer about 30 regions of interest per second using the NVidia Jetson TX 2. In the training step of deep learning, a recognition rate of 90% or more was achieved from a sample size as few as 200 sheets by turning over the image, blurring, etc. to increase the number of samples through variation.

6 Comclusion

Based on the results of the 2016 competition, OUXT Polaris rebuilt the system and developed an improved system for the Maritime RobotX Challenge 2018. We succeeded in constructing a highly reusable system by designing systems with high independence as parts, in addition to high computing capacity and environmental recognition capability. We also focused on a lower center of gravity and increased navigational stability. On the software side, deep learning was utilized for high accuracy in object recognition, developed as open source. We hope these significant upgrades will produce positive results in the competition.