TAU SAIL-IL 2022 RoboBoat competition technical design review

SAIL-IL team, Tel Aviv University, Tel Aviv, Israel

***Abstract*— This report describes the entire strategy, system engineering, design consideration and project life cycle process include design, integration, verification, and validation of the 2021-2022 Tel Aviv University SAIL_IL team, as part of the RoboBoat annual contest**

I. INTRODUCTION

SAIL-IL is the Tel Aviv University's Engineering Department RoboBoat team, competing second year in a row at the RoboBoat competition and the only team from the state of Israel. The RoboBoat program aims to create an environment that allows students to practice research and development in the challenging and evolving world of autonomous vessels. This year the mission was to continue last year work and develop the program further with new tasks.

The objective was to develop a fully autonomous ASV (Autonomous Surface Vehicle) to operate autonomously under various missions including obstacle avoiding route, obstacle detection /classification, platform control and localization. development process was based on agile development due to the tight constraints of the project and uncertainties. It required an on going planning and execution method and integration between multidisciplinary teams and various sub-systems.

II. COMPETITION STRATEGY

This year we focused on improving and further developing of the system based on conclusions from last year work. The beginning of the development process focused on system engineering process, understanding the requirements from the operational concept as well as internal requirements related to last year conclusions and allocating them to the various sub-systems as well as to the related teams. The team focused on the ability to add more capabilities to the current system to be adaptable to new missions and requirements. We prioritized the mandatory navigation task and the navigation channel as they require similar system capabilities, and later added the speed gate. As the water and ball cannons missions were defined at a late stage, we chose to prepare only for the water blast task. The docking task was not attempted due to lack of resources.

All tasks are based on recognizing and classifying objects, on the fly path planning and have similar manoeuvrability requirements. The entire software architecture is based on ROS (Robot Operating System) framework[1]. Add to this that the architecture is loosely coupled between the various sub-systems allow parallel and scalable development (for example obstacle avoidance using Computer Vision and LIDAR in case one was not sufficient) as well as sub-system pairs integrations. Another advantage of ROS was noticed when the shore station was connected to the ASV via wireless network thus eliminating the need to write Server-Client infrastructure from scratch. To lower risks and mitigate the development all off-the-shelf components integrated in the system are compatible with ROS and has built-in SDK's. It allowed the teams to focus on algorithm and system software development.

A. Mandatory Navigation Channel

To demonstrate basic autonomous capabilities, the navigation task is mandatory and must be completed first. To accomplish this task the ASV should identify both gates and navigate through them, keeping linear course. Preparing for this task allowed us to develop and improve all the basic capabilities such as object detection, localization, control, and path planning. In addition, it allowed us to test the stability and reliability of the trimaran hull.

[1] See reference 1

B. *Obstacle Channel*

The obstacle channel requires the ASV to identify the next gate and pass through it without hitting obstacle and gate buoys. This task requires updating the path continually while keeping the ASV in a curvy lane. Using stereo lab camera with 120˚ FOV, the buoys are detected and classified. Once the nearest set of buoys detected, the ASV will navigate to the average point between them. Practically the course is divided to segments which makes the path planning easier, as only one point is targeted and less obstacles should be identified at once, and allows better error correction.

C. *Water Blast*

This task requires identifying target and filling a tank until the ball reaches to the marked line. It demonstrates precise control, aiming and hull stability as the ASV should relatively stay in place. Although this mission was published in a later stage, we decided to attempt it as we had dedicated team for it. Furthermore, we already had the propulsion system controller which made easier integration of the cannon. A cannon was built and positioned on deck.

D. *Snack Run!*

The Snack Run task demonstrates hull form efficiency coupled with its propulsion system, and the resulting maneuverability. The ASV should enter through gate buoys, go around the mark buoy and exits through the same gate buoys as quickly as possible. This year we decided on a trimaran design that allowed as a bi-directional movement and better stability.

III. DESIGN CREATIVITY

Since it is yet fairly new program, we raised funds for the project throughout the year. We performed a cost benefit analysis on each design component to best utilize our limited budget. For example, we decide to manufacture most our hull in-house via CNC and manual sheet metal bending machines and 3D printers.

² See reference 2

A. *Hull & Propulsion Design*

Last year's platform was a catamaran. Although it was fairly stable platform, the stern sank and tilted. Furthermore, a lot of water reaches the deck mostly in reverse drifting. Moreover, we decided to prioritize manoeuvrability and platform stability over speed. That led us to the trimaran design with two azimuth stern thrusters at 90˚ and one bow thruster embedded into the center hull. This allows the "crab-like" sailing which could be useful for avoiding obstacles and maneuvering at low speed. As most of the center hull is placed underwater, most of the weight is centralized thus allows less pitching. Furthermore, stability is gained also from submerging the side hulls and creating a predictable increase in righting moment. To meet this requirement, we chose to design a symmetrical vessel.

The hulls are made from foamed polystyrene, which was milled with a CNC machine, like surfboards production. Foamed polystyrene was chosen due to its lightweight and ability to bear strikes. After producing the desired shape of the hull, it was coated with fiberglass, carbon fiber, and epoxy to keep it sealed and stiff. The deck of the ASV is made from layers of birch wood. To ensure the electronic systems on the ASV are cooled properly, the electronic box placed on deck is cooled by using a heat sink which are placed under the wooden deck and create and ideal convection.

B. *Computer Vision*

1. *AI Interface Algorithm*

For object detection we decided to use the real-time object detection algorithm model YOLOv4-tiny[2], which offers a good balance between speed and accuracy. The model was trained and deployed via darknet framework, which enables integration of the model with the rest of the system through ROS[3].

2. *AI Training*

To create our custom trained model, we assembled a dataset of the required objects as

³ See appendix B.1

detailed in the competition rules. The dataset was composed of pictures from a variety of sources, including previous competitions and pictures we staged ourselves during test runs in the field. The dataset was then expanded by creating augmentation and different variations to simulate variance in angle, exposure, saturation, brightness, noise, etc. The dataset was split into three different sets: training set (70%), validation set (20%) and testing set (10%).

3. Deployment

After the model was trained, it was then integrated into the system such that it would process live images from the camera and augment them with bounding boxes. This data stream is fused with the camera's pointcloud to create a marker array detailing type, relative location, and certainty of detected objects.[4]

C. Range Estimation

As part of last year's conclusions, we planned to fuse the data from the stereo camera and the LiDAR system. The stereo camera is used to classify obstacles in the close surrounding and the LiDAR is used for both detecting obstacles from further range and improving the accuracy of the object range estimation. Throughout the work, we understood that we can get even more information by separating the two sensors' data collecting and processing. Hence, we decided to use the LiDAR to generate Laser Scan data for the navigation process and SLAM[5] (Hector)[6].

Innoviz joined as sponsor and offered InnovizOne Lidar which is a solid-state sensor with 115˚x25˚ FOV and up to 250m Detection Range.

D. Localization

For localization a 6 axis IMU and GPS were used. For sensor fusion and minimizing error Kalman Filter was used through Robot Localization ROS package[7]. This package integrates inputs from IMU and GPS and uses Kalman filter to evaluate the relative position of the boat. The Kalman filter algorithm receives as input a series of noisy measurements of different localization values representing the state of a system at a given time, and estimates the system location, based on the result of the filter in the past[8].

E. Path Planning

Path planner relies solely on information received from the sensors in real time, i.e. localization given by the odometry (GPS, IMU), the buoys identified by the camera and obstacle detection conducted by LiDAR sensor. Two planners are used to get a robust path planning:

Global Planner – algorithm based on Dijkstra's algorithm[9] was coded to create a path around obstacles known to the robot. While A* is known to be the better algorithm in term of computational power, the Dijkstra's algorithm performed better through experiments.

Local Planner – we chose DWA (Dynamic Window Approach) Planner[10] to produce obstacle avoidance functionality. In general, DWA is an online collision avoidance approach in robotics that takes under consideration the constraints imposed by limited velocities and accelerations of the robot. DWA ROS package is used to follow the global plan and to consider new obstacles in the way. The Local Planner sends cmd_vel which is velocity commands to the controllers.

F. Control

Nvidia Jetson Nano is used as a controller, that receives cmd_vel commands from the path planner which provides angular and linear velocity. IMU and magnetometer data are used by a feedback loop to compare the wanted and practical velocities. Linear velocity is extracted from IMU data by integration of the accelerations in different axes. In addition, the magnetometer and IMU provide measurements of the changes in angles in two axes, which enables the calculation of the angular

[4] See appendix B.2
[5] See reference 3
[6] See appendix B.3
[7] See reference 4
[8] See appendix B.4
[9] See reference 5
[10] See reference 6

velocity. For the thrust configuration, BlueRobotics data sheet was used[11] to establish the momentum generated by the engines in respect to pulse width modulation (PWM) and voltage[12]. PID controller is used in which the required velocity by the algorithm is being transformed into momentum needed and in turn changes the PWM signal that is being sent to the engines.

G. Water Cannon

A cannon was designed based on reverse engineering of an RC jet-ski waterjet system. The following design[13] enables to deliver large amounts of water. The cannon has two main stages, called First Stage and Second Stage, where propellers are set in a water leak proof enclosure and driven by a shaft connected to a brushless motor. This brushless motor can rotate at very high RPM (Max. 40,560RPM) at voltage of 12V. It is located on the bow, above the camera and LiDAR on a variable degree adapter.

To obtain the task, the target is identified via the computer vison system described above. By conducting several experiments, we were able to determine the angle of the cannon[14], the right R.P.M and the distance from the target to fill the tank in the most efficient way. The cannon shoots in specific time intervals and readjusts the boat's location and the cannon's R.P.M according to the target's updated location. This strategy was chosen to avoid the use of complex computer vision and delicate angle changes.

H. Autonomy

The autonomous function of the ASV is governed by a state machine[15], managing the flow and transition between competition tasks, and monitoring the various onboard systems to detect and handle hazards.

Provided that all start-up checks have passed, and we have successfully reached our desired starting point (e.g., 6 ft from the mandatory navigation channel), the ASV transitions to autonomous mode. In this mode, the state reflects the current task, with the basic flow of: (a) locate task starting point (for example, gate buoys). (b) perform task; More complex tasks, such as the Water Blast, are divided into subtasks, each represented by its own state. (c) Move to next task according to our competition strategy; location is predefined using the provided course configuration. In addition to the regular flow, there are flows dedicated to hazard handling and recovery, for example in the case of a malfunctioning sensor. Finally, there is a general "return to home" state.

For implementing the state machine, we looked for a framework that natively supports integration with ROS, since the state machine should communicate with the other ASV components over a distributed ROS network. As the state machine is a new addition to the ASV, we did not have a previous implementation to build on, and it was important for us to focus our efforts on developing the state machine itself rather than on drivers, APIs, or possible compatibility issues. Additionally, we preferred tools that allow for testing and simulation, in both isolated and real-world environments, with relative simplicity.

Following initial difficulties with Gazebo, we decided to use Stateflow and Simulink from MathWorks, who also kindly offered us guidance through the first steps of the project. Stateflow supports many features for state machine design, including global variables, code and graphical functions, timeout-based state transitions and more, all in a graphical interface. This enabled us to focus on the underlying logic and design a more robust state machine, that maintains autonomous control for as long as possible. Moreover, the MathWorks ecosystem includes a

[11] See appendix B.5
[12] See appendix B.12
[13] See appendix B.6
[14] See appendix B.7
[15] See appendix B.8

complete ROS toolbox that can generate C++ code for a ROS node running our state machine. Besides significantly reducing coding time, running on C++ is of critical importance, since our state machine is intended to be run on an Nvidia Jetson Xavier NX and is required to operate in real-time.

I. Communication

For the physical network, we used a communication switchboard, which all the ASV computers are connected via Ethernet cables (802.3) and the shore station's computer is connected via a closed WIFI 2.4Ghz (802.11) network[16].

For software implementation rosmaster package[17] was used. With rosmaster, one computer is configured as the master in all the computers that are connected to the network, thus all ROS nodes and topics are shared between all on deck and shore computers[18]. Since most of the modules on the ASV already use ROS as their platform to publish information, the connection and integration were immediate.

A communication package was coded that allows recording sensors' data, present real-time data of system health in shore station and perform emergency shut down.

IV. EXPERIMENTAL RESULTS

Before all water experiment multiple lab experiments were conducted. Although our lab time was greater than water time, we learned that it makes experiments more efficient.

A. Object Detection

During both lab and field experiments photos of all obstacles were taken in different environment conditions to expand database. The system generates a consistent live data stream of 7 marker types, with stable speeds averaging at 24 FPS, and mean average precision of 86.29%. This is an improvement of previous year's CV team, which achieved identification of 4 obstacle types at mAP of 68%, at higher speeds of 30 FPS.

B. Path Planner

To verify the correctness of the algorithm simulations in RVIZ[19] and Gazebo were used. Using simulation allowed parallel development of the path planner, independent from the platform and control development. The characteristics of the platform and the missions' elements was injected to the simulator. This allowed easier debugging.

C. Floating Platform

As the hull was custom made an experiment to ensure buoyancy and stability was conducted. The hull was tilted and loaded with weights. We found that the hull is very much stable in all directions and can carry up to 30 kg.

D. Communication

As there are multiple computers on deck we used the rosmaster protocol. In the experiment we published multiple messages to ensure that all the computers are subscribed to the relevant topics and identify the master computer. we found that all CPUs can communicate and there is neglected latency in retrieving data. This allows better control and easy data transferring.

V. ACKNOWLEDGMENTS

We would like to thank Tel Aviv University's Faculty of Engineering and to Dean Prof. Yossi Rosenwaks allowing us to participate in this unique project, and for offering us support and resources.
Special thanks to Danny Berko and Jacob Fainguelernt, our academic directors, and Tel Aviv University faculty members, who believed in the project and helped make all our ideas come true.
Thanks to all our professional mentors for offering a lot of knowledge, experience, and support: Dr. Hanan Spond, Roi Raich, Bental Tavor, Hanoch Aharon, Amitai Peleg and Omer Karakukli.
Thanks to all our gracious sponsors for funding and accompanying the team: Rafael Advanced Defence Systems, Valens, Systematics Ltd., Innoviz Technologies and Lake TLV.

[16] See appendix B.9
[17] See reference 7
[18] See appendix B.10
[19] See appendix B.11

VI. RFERENCES

1. Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng, "ROS: an open-source Robot Operating System", ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.
2. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", arXiv:2004.10934, 2020.
3. Robot Mapping course [online], 2013.
4. robot_localization Package Summary, ROS Wiki.
5. Huijuan Wang, Yuan Yu, Quanbo Yuan, "Application of Dijkstra algorithm in robot path-planning", 2011 Second International Conference on Mechanic Automation and Control Engineering, 2011.
6. dwa_local_planner Package Summary, ROS Wiki.
7. rosmaster Package Summary, ROS Wiki.
8. BlueRobotics T200 specs [online].

Appendix A - Component Specifications

Component	Vendor	Model/Type	Specs	Custom/ Purchased	Cost $	Year of Purchase
ASV Hull Form/Platform	nanofiber	custom made	foamed polystyrene covered fiberglass, carbon fiber, and epoxy.	custom	2,507	2022
Waterproof Connectors	MPH Electrical Engineering	EKPK 180 G 2538055 outdoor junction boxes		purchased	111	2022
Propulsion	BlueRobotics	T200	T200 specs	purchased	210	2022
Power System	Fullymax	LIPO 14.8V 5000mAh		purchased	144	2022
Power System	Fullymax	LIPO 14.8V 7500mAh		purchased	212	2022
Power System	Fullymax	LIPO 18.5V 5750mAh		purchased	206	2022
Power System	Fullymax	LIPO 7.4V 7500mAh		purchased	123	2021
CPU	Nvidia	Jetson Xavier NX	Jetson Xavier NX specs	sponsored		2021
CPU	Nvidia	Jetson Nano	Jetson Nano specs	sponsored		2021
Motor Controls	SeeedStudio	PCA9685	PCA specs	purchased	20	2022
Teleoperation	Ubiquti	Bullet M5	Bullet M5 specs	purchased	115	2021
Compass	Yiqigou	GY-271 QMC5883L	Magnetometer specs	purchased	11	2022
GPS	U-blox	c94-m8p	GPS specs	purchased		2021
IMU	HiLetgo	GY-521 MPU-6050 MPU6050	IMU specs	purchased	10	2022
Camera	StereoLabs	ZED2	Camera specs	purchased	500	2021
LiDAR	Innoviz	InnovizOne	LiDAR specs	sponsored		2022
Water Cannon		Custom self production				
Algorithms						
Vision		YOLOv4 and darknet				2022
Localization and Mapping		Custom based on Dijkstra, DMW ROS package and robot_localization package		custom		2022
Autonomy	MathWorks	Simulink				2022
Communication		Custom based on rosmaster package				2022
Open-Source Software		ROS				2022

Appendix B

1.

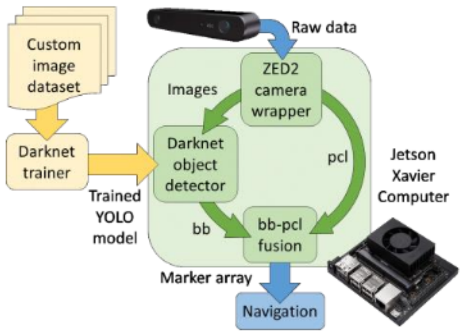

Figure 1 : CV interface architecture

2.

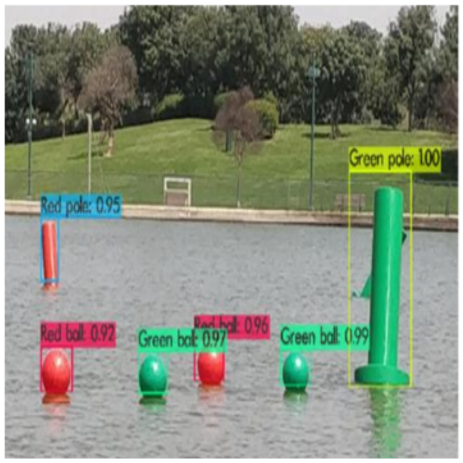

Figure 2 : bounding boxes and accuracy in real time

3.

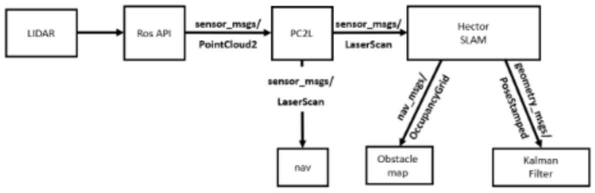

Figure 3 : LiDAR interface architecture

Pc2l – gets as input the sensor point cloud. converting it to 2D, filtering the relevant FOV (angle and height, mainly filtering out the water), and publishing laser scan message.

Hector SLAM – gets as input the laser scan and uses it to create an obstacle map (Occupancy Grid) of the environment and to estimate the boat's movement and its position on the map.

4.

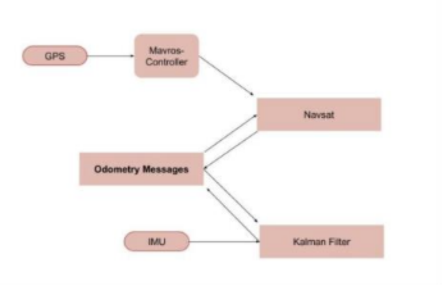

Figure 4 : Localization interface architecture

5.

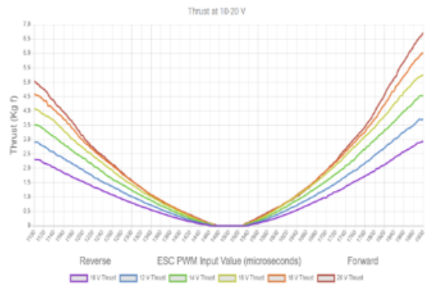

Figure 5 : momentum vs. PWM[20]

6.

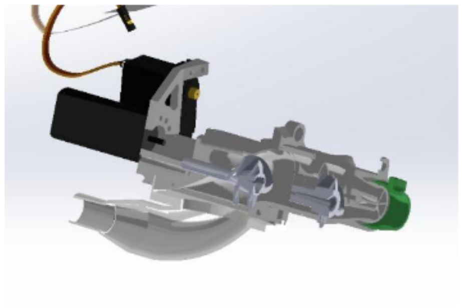

Figure 6 : Water cannon design

[20] See reference 8

7.

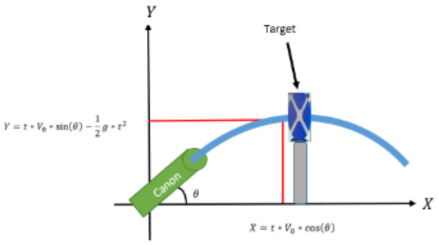
Y
Target
$Y = t * V_0 * \sin(\theta) - \frac{1}{2}g * t^2$
Canon
θ
X
$X = t * V_0 * \cos(\theta)$

Figure 7 : Water cannon angle calculation

8.

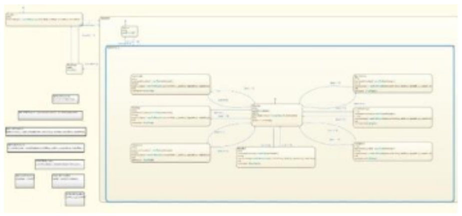

Figure 8 : state machine

9.

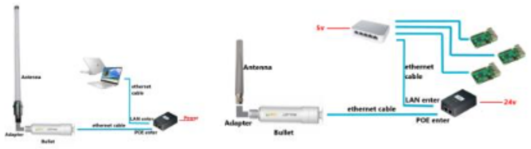

Figure 9 : Communication interface architecture

10.

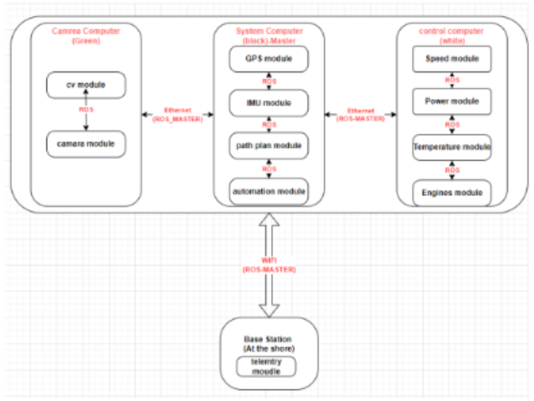

Figure 10 : System deployment

11.

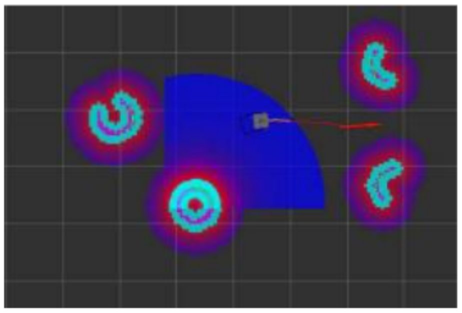

Figure 11 : RVIZ simulation

12.

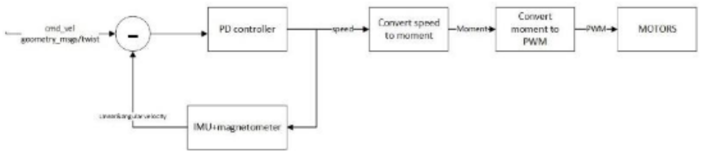

Figure 12 Control interface architecture

Appendix C – System Architecture

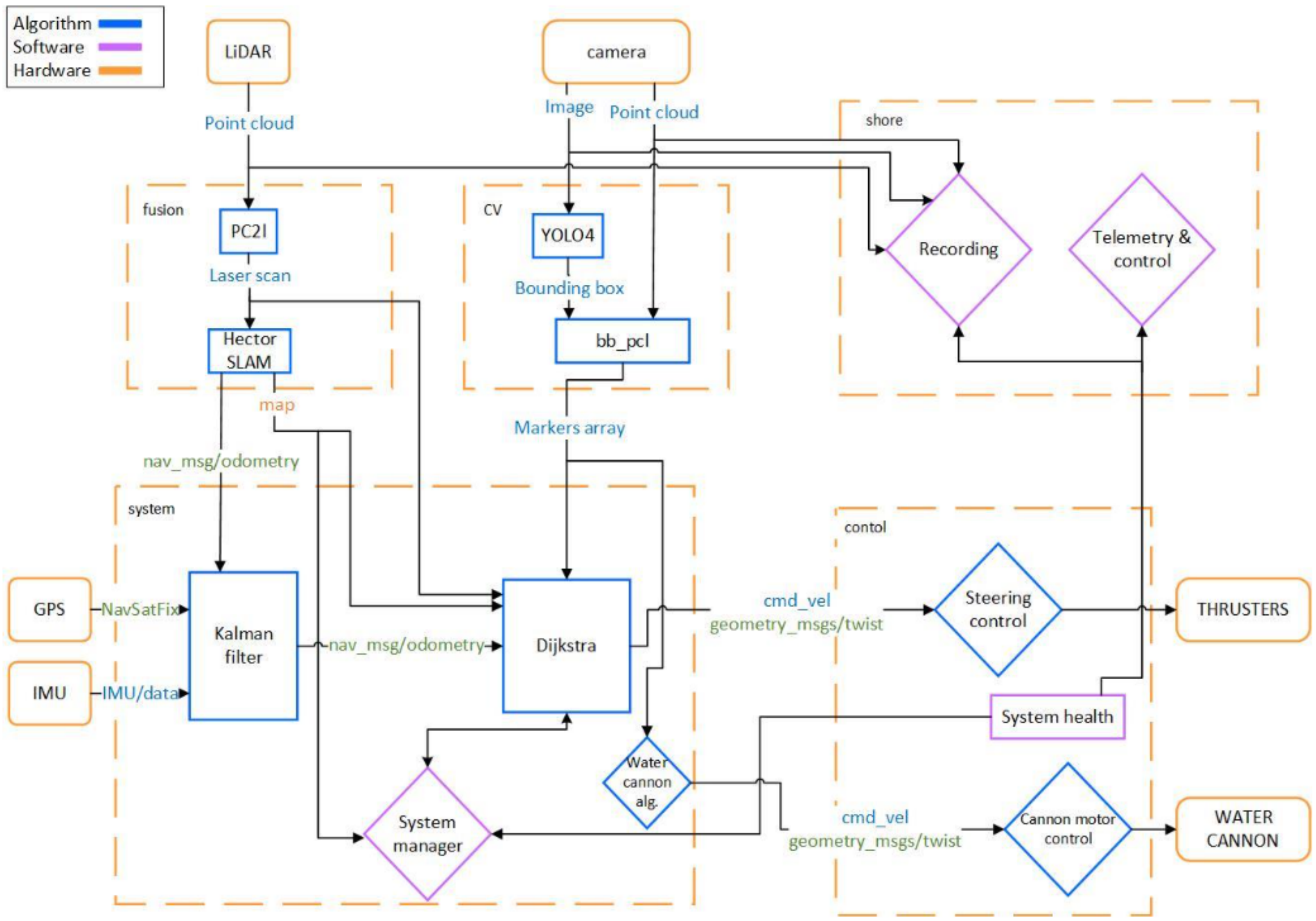

Figure 13 System architecture