Master Systèmes Dynamiques et Signaux

Mémoire

Marine Research Platform: Autonomous Sailbot ASPire

Author :
M. Yacine Benhnini

Supervisor :
Ms. Anna Friebe

Jury :
Pr. L. Hardouin
Pr. L. Jaulin

Dated
August 27, 2018

Acknowledgement

I would like to thank all the Åland Sailing Robots for the work during this summer, and the good working environment and team spirit.
I thank Anna Friebe for having made this internship possible and being available for when help was needed.

Contents

List of Figures

List of acronyms

CAN	*Controller Area Network*
AIS	*Automatic Identification System*

Introduction

As the project was reaching its end, my internship as a test engineer mainly focused on having the boat ready before its evaluation as it will be evaluated for marine sensor measurements and harbour porpoise detection. It included a lot of work for uniting features that were partially merged or not merged with the whole system, improving some of them, and make sure everything is tested to delete as much bugs or undefined behaviour in the system as possible.
The two features that will be discussed in this report are the camera processing for obstacle avoidance, using a thermal camera, and a navigation system based on a voter system.

The last part will discuss the building of a new sailboat for competing in the Worlds Robotic Sailing Championship.

Chapter 1

Image processing

1.1 Image processing

Image processing is a decisive part for obstacle avoidance and is a challenging issue [4]. Using AIS only to detect the presence of boats around the autonomous sailboat will not be enough in a lot of situations, especially when sailing close to a coast. It is very likely in these area to encounter vessels which are not equipped with an AIS, thus loosing the conveniency of having an easy access to informations like the position, speed and course of other vessels.

The ASPire is equipped with a thermal imaging camera. One of the advantage of a thermal camera over a regular camera is its robustness against some meteorological conditions like fog, which can be very present during some seasons in the area of the Åland Islands. Another good point is that the seawater will generally be noticeably colder than every other objects in the thermal camera vicinity. It makes a proper ground for using simpler detection algorithms with less computation time in the embedded system as we are quite limited in this field, and want to save power to keep the boat self-sustainable. Last interesting observation, seawater temperature on the surface is quite homogeneous, with very low amount of disparities in the vision field of the camera.
The waves are an issue, appearing most of the time as bright small lines scattered in the image (fig. 1.1).

The processing algorithm was first mainly focused on edge detection techniques such as Canny filtering, using erosion and Gaussian blur in the preprocessing part (fig. 1.2).

The inconvenience here is that it is very hard to keep as much of the environment contours as possible, especially keeping the horizon line visible in the output, while maintaining a very low amount of waves appearing.

Figure 1.1: View of a shore from the thermal camera.

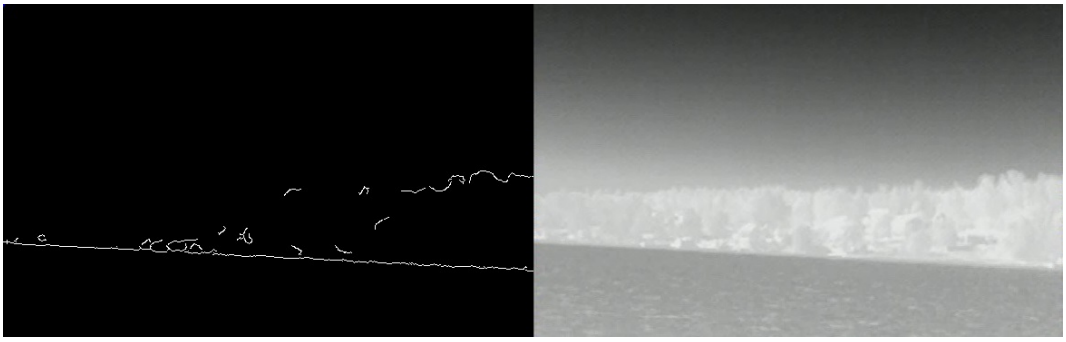

Figure 1.2: First edge detection algorithm applied.

It is very problematic considering the way we want to detect an obstacle. The goal is to keep the horizon line visible in the edge detection, and an obstacle in our vicinity would appear lower than the horizon line depending on how big and close it is to the camera. The lower part of the output is then considered as the amount of free space we assume having in front of the boat (fig. 1.3).

Figure 1.3: Algorithm output with a boat in front.

In this case, letting waves appear in the output will make the algorithm interpret them as obstacles, and removing them may lead to some obstacle be invisible or partly invisible.

My work on this part was mainly to improve the processing while keeping it light enough for real time use and power usage. As a first shot I assumed that the waves will most of the time appear as horizontal lines. Even if the boat is rolling a lot, a rotation is applied to the image using the data given by the compass to always have a view as if it were 0° of rolling. With this assumption we can change the shape of the erosion kernel to a more vertical one. As the waves lines are really thin, they are more likely to disappear than any other obstacle which occupy more pixels on the vertical (fig. 1.4).

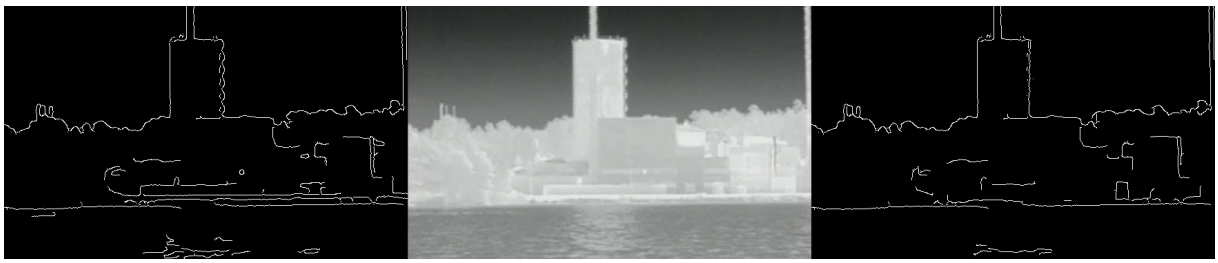

Figure 1.4: Square kernel on the left against vertical kernel on the right.

The second big assumption is that the seawater will always be homogeneous, and colder than anything else in the visual field, at least in spring/summer time. So I applied the meanshift [1] segmentation algorithm, this algorithm can be interpreted as looking for homogeneous part and grouping them in a single cluster. The first implementation was a version without clustering. I applied the algorithm and then I kept using an edge detection algorithm so I could keep the same kind of output, thus avoiding to change the whole obstacle avoidance functions in the software (fig. 1.5).

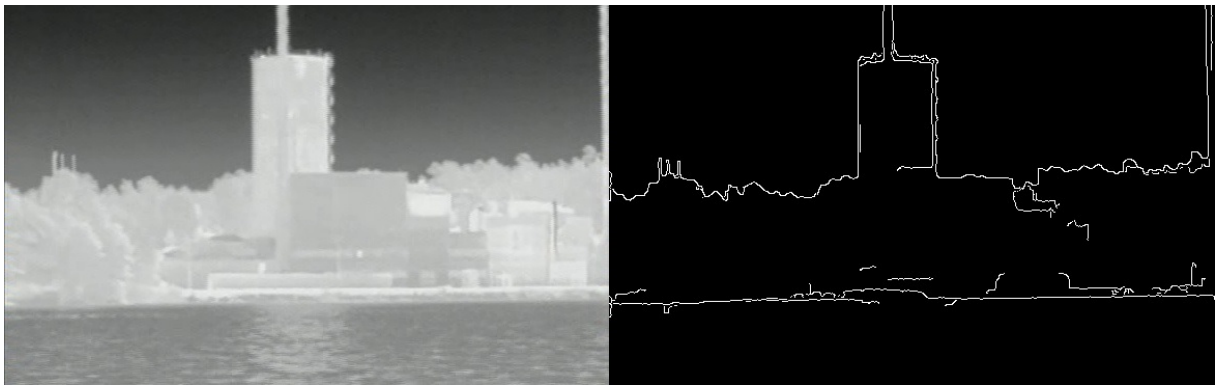

Figure 1.5: Filtering stage of meanshift segmentation.

The results are really satisfying, it is way easier to keep a lot more details on the overall environment while having a very low amount of wave lines appearing. The computation

time is unfortunately increasing a lot when using the best sets of parameters, reaching sometimes one or two seconds for a single frame. Considering the boat speed it is not really a big deal, but on the other hand the amount of computational resources used was too much on the boat hardware. It would be risky to freeze the tasks of the navigation program when having a heavy image processing occurring.
The last idea is to take advantage of the fact that the meanshift segmentation algorithm can directly output the clusters it found. Then making use of a small calibration it is easy to detect which clusters are seawater and which ones are not. As the boat is equipped with a temperature sensor, among other sensors, for marine data gathering, we can relate the clusters output with the temperature measured. A simpler way, in case we don't have any sensor, is to manually check the first frame and set a given cluster to be the seawater. As the seawater temperature will not noticeably change in an area during a mission, we can confront each clusters to this sort of threshold, and define them as seawater or not. To keep track of temperature change during long missions, we could recalibrate this threshold with the closest value each given time, as the closest value will most likely be seawater. The seawater cluster will also always be one of the biggest, and located in the lower part of the image.
Considering the processing speed, doing a simple resize of the image before processing was perfect. Unfortunately time was missing for implementing the whole clustering process.
Now we have the output of the meanshift filtering and edge detection, the simple interpretation of the output to detect an obstacle is subject to some quite dangerous situations where an obstacle will be interpreted as a free space (fig. 1.6).

Figure 1.6: Failing situation when the bottom of the image is not water.

If the obstacle cover one side of the field of view to the very last pixel down, then the first edge detected will be the top of the object and not its bottom. Very simple case: when launching the boat from the dock, the camera will most likely have a wall appearing on one side, which is going to be interpreted as free space. One way to give some more robustness is to do some comparison between mean,min,max,and percentiles of the height

distribution in the processing output. This way it is possible to do a distinction between some situations.
Depending on the horizontal size of the obstacle, as it is appearing in the field of view, the interpretation will switch depending on which percentiles are used as a threshold: if we use the first and third quartiles on the previously seen wall, it will be interpreted as an obstacle as soon as it takes half of the horizontal field of view. Unfortunately this still doesn't make possible the distinction between a wall on a side and a boat taking nearly the whole horizontal field of view.

1.2 Obstacle avoidance

The obstacle avoidance is implemented in the voter system through the proximity voter and the midrange voter, using AIS data and the video stream from the camera (described in the next section). One serious issue with the camera avoidance is its very narrow field of view: only 25 degrees wide for its horizontal span. The output of the image processing algorithm is cut into 25 columns, each columns corresponding to a 1 degree span. With such a narrow field of view the boat will not see the obstacle after having done a slight turn, then it might turn back toward it right after if for example the waypoint is in this direction. To avoid this, when an obstacle is seen, it is kept in memory for a short time, so we have less chance of turning back toward an obstacle.

Chapter 2

Navigation system

The Navigation system has two algorithms implemented. One is a line-follow algorithm based on the following publication [2] and the other is a voter based system [3] .
The idea is to keep algorithms simple enough to keep a low computation power requirement, thus decreasing the energy consumption by using less powerful hardware. The previously mentioned algorithms fit perfectly in this mindset.

2.1 Voter based navigation

The main part of my work on the navigation system, was to improve the current voter based system and test it to find and solve undefined or unwanted behaviour.
The main idea behind this navigation algorithm is to reduce the complexity of the algorithm while taking into account as much parameters as possible, the voter based system rely on smaller entities that tend to take care of one thing only. We can list the different voters as following: the course voter, waypoint voter, wind voter, channel voter, midrange voter and proximity voter. Each voter is going to put votes on bearings it wants to reach, each votes are summed into a global ballot, taking into account the weight given to each voters. In the end the bearing holding the maximum number of votes will be the chosen target. The names are pretty much self-explanatory on the different voters purpose: - the course voter votes for the current heading of the boat. It is useful for lowering the amount of energy spent by flipping the actuators all the time when the global target course is varying by a few degrees. It is also used to compete against the channel voter, for adjusting the width of the tacking patterns when a tack is needed. - the waypoint voter will vote for the bearing corresponding to the angle between the next waypoint location and the boat heading. - the wind voter checks the wind direction and if the boat enters a beating mode, this voter votes for the closest tacking angle corresponding to the current heading. - the channel voter checks the distance between the boat and the line joining the previous waypoint and the

next waypoint, adding more votes toward the line the furthest it is to it. - the midrange voter uses the AIS signals and check other boats position and course. A risk factor is computed and the voter puts negative votes depending on this risk. It is triggered when a vessel is detected between 100 to 200 meters. - the proximity voter uses informations from the AIS and the camera. The AIS part is used only if a vessel is detected under 100 meters, the camera is used all the time. A risk factor is also used to level the votes.

2.1.1 Modifications and improvements

Having multiple small entities makes things relatively easier to understand what the overall system is supposed to do, but the increasing number of parameters makes it sometimes very difficult to catch where the wrong behaviour comes from. So one of my first goal was to make sure the voters are as independent as possible, and none of them ends up doing something that is already done by another one.
It could be simple things as the proximity voter preferring to give more importance to the camera field of view, thus giving more votes to bearings in front of the camera when avoiding. This way the boat is more likely to turn towards a bearing where it knows nothing is there, than turning in a direction where it was blind before, with no information on it. But considering the very low field of view of the camera (24 degrees), this ended up doing the same votes than the course voters when the field of view is clear of obstacles. Then having a higher weight on the avoidance, makes the boat keeps its heading too much, leading to larger tacking patterns for example.

2.1.2 Wind voter

The wind voter has been modified to not vote for some courses, but to put veto instead. The idea was to delete as much negative votes as possible. Before the wind voter would lower the global votes corresponding to the wind direction +/- 45 degrees, and checks for the current heading of the boat to add votes to the closest tacking angle. The issue is it does not actually forbid the boat to go towards the wind in some very few cases, making it fails to reach a waypoint.
Switching to vetos, and eventually even deleting the extra vote on the tacking angle, it makes the boat actually avoid a chosen zone. It is just applied to the global ballot as a mask. In a tacking situation, the waypoint voter will be adding votes on the side of the line as well as the channel voter so the boat will eventually have the good pattern, so no need for the extra votes from the wind voter.

2.1.3 Course voters

In situations where strong gusts and drifts happen, the boat could do a full turn on itself while trying to enter a tacking pattern. It is due to the course voter having more power when the bearing to the next waypoint is within 45 degrees to the true wind direction. The veto applied by the wind voter makes the waypoint voter less powerful, and when the boat is reaching the tacking angle, it will keep turning with the momentum and the gusts. The course voter having more weight in this case because it is not masked by the wind voter, the boat will have an excessing bearing for a few seconds and the channel voter will put it back to the good heading, making the boat do a full turn in some cases.
To prevent the effects induced by a lot of variations of the wind direction and speed, I implemented another voter that memorizes the previous target and votes for this one, called last-course voter. By putting just a bit more weight to this voter than the course voter, it will sort of overwrite its small variations and drifts.
These voters follow a simple linear formula:

$$votes[target] = maxVotes; \tag{2.1}$$
$$\text{For } i \in [1, 10] : \tag{2.2}$$
$$votes[target + i] = maxVotes - (i/90) * maxVotes; \tag{2.3}$$
$$votes[target - i] = maxVotes - (i/90) * maxVotes; \tag{2.4}$$

2.1.4 Waypoint voter

The waypoint voter has been modified to have a larger range of headings where it adds votes, and an increased minimum of votes. This modification was following the wind voter usage of vetos. Because the vetos cover 90 degrees centered on the true wind direction, in the upwind and downwind directions because we use a wingsail, the waypoint voter would be entirely masked by the wind voter if the bearing to waypoint is in the masked area. To solve this, the voter follow this formula:

$$votes[target] = maxVotes; \tag{2.5}$$
$$\text{For } i \in [1, 90] : \tag{2.6}$$
$$votes[target + i] = maxVotes - 0.4 * ((i/90) * maxVotes); \tag{2.7}$$
$$votes[target - i] = maxVotes - 0.4 * ((i/90) * maxVotes); \tag{2.8}$$

We use a linear decrease of number of votes from 100% to 60% in a 90 degree span around the bearing to waypoint. With this larger span and a minimum of 60% of maxVotes, we keep the influence of the voter even if the next waypoint is upwind or downwind.

2.1.5 Channel voter

The channel voter is monitoring the distance between the boat and the line joining the previous waypoint reached to the next waypoint. The idea is to have it keeping the boat within a given distance to the line. It is done simply by adding vote in the half side pointing towards the line. It gives the same amount of votes for each headings on the good side. This way it is just acting as an offset, without giving a preference to any bearings as long as they point towards the side where the line is.

2.1.6 Proximity voter

The proximity voter takes the image processing output and AIS informations. It uses AIS informations only if a boat is between 50 to 100 meters away. The output of the image processing is cut into 25 columns, to match the 25 degrees of horizontal span of the thermal camera. Then the interpretation described in the previous chapter is implemented, the voter will check the percentage of white pixels in the column and set its vote to the same percentage of its maximum votes. Unfortunately it has not actually been tested properly on the boat, and the simulator tests have been proven later to be unreliable for this test.

2.1.7 Simulation tests of the voters

The tuning of the voters have been mostly done through the simulation, in the following figure we can see the result with the last chosen set of weights. (fig. 2.1). The set of weights is:

Course	LastChoice	Waypoint	Channel
0.6	0.7	1.0	1.0

Table 2.1: Set of weights.

Then we can see the difference when enabling the midrange voter and adding two vessels nearby (fig. 2.2).

With a weight of 1, the minimum distance has reached 26 meters which is really dangerous. The only visible influence of the midrange voter is a very small deviation before its last turn, whereas in the other setting with weight=3, the boat enters a big loop before going back to its pattern without avoidance.
Lastly when adding the proximity voter (fig. 2.4), the boat have avoided the other vessels

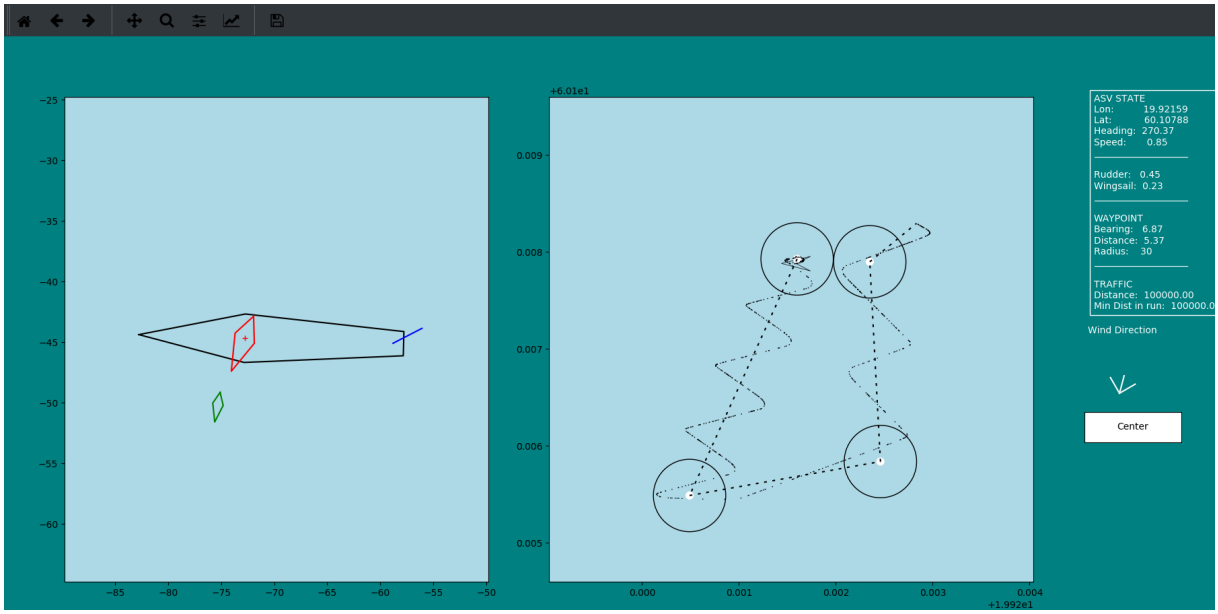

Figure 2.1: Simulation of the voter system without avoidance

but we can see that the tacking patterns have been narrowed a lot. It is due to the proximity voter having the same votes as the course voter when there is no obstacle in front of the camera. Because of this it can interfere in the course vs channel balance, thus reducing the channel size.

2.1.8 Testing the navigation system

In the figure (fig. 2.5) are the logs of a sailing test near the university.

The wind was coming from the south during this day. It is the same mission as the one used in the simulator so we could compare the results more easily. We can see that both navigation algorithms are having similar trajectories as expected, with the voter system being sometimes more unstable with little unnecessary turns when reaching third waypoint in this particular case.
When this test has been done, the last-course voter was not implemented, thus the voter system was really sensitive to drifts and winds variation. It was unfortunately something not visible in the simulation and there was too little time to improve the simulator while doing the other tasks.
After multiple tests during the summer, we had a really hard time having the boat sailing in wind conditions not close to perfect, either due to the wingsail command or mechanical issues.

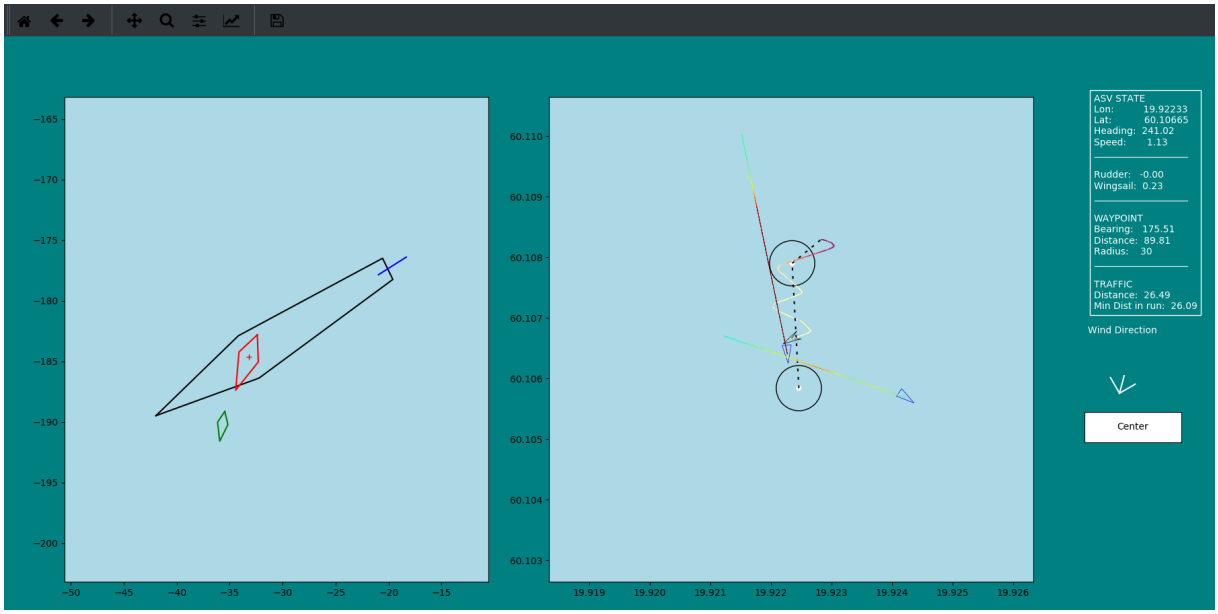

Figure 2.2: Midrange voter weigth=1, a small deviation is visible

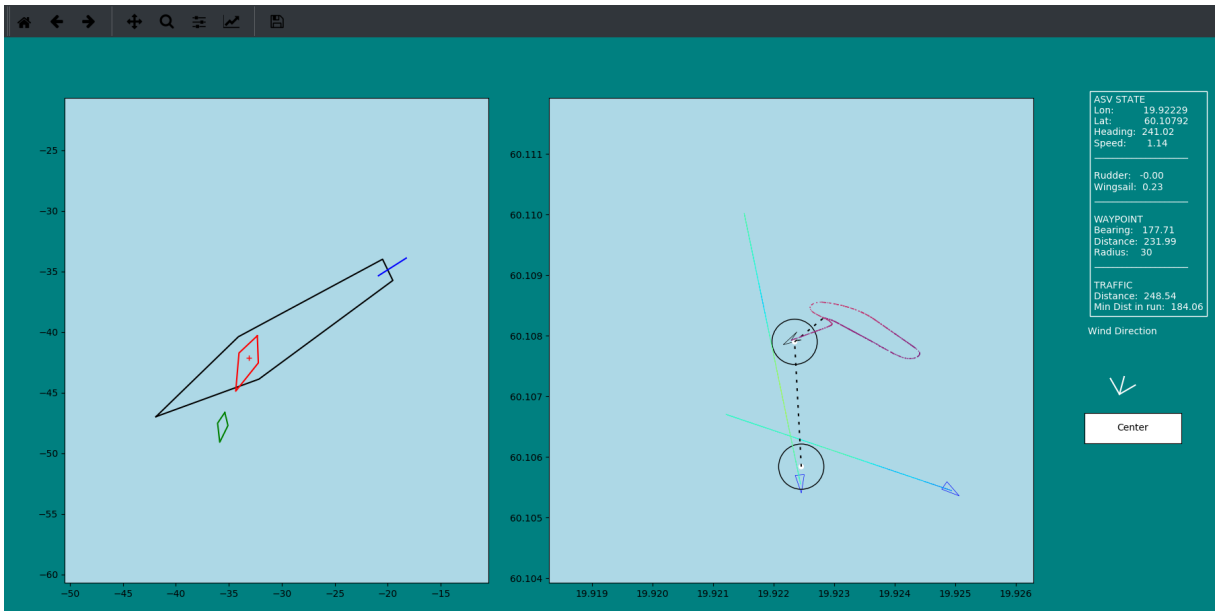

Figure 2.3: Midrange voter weigth=3

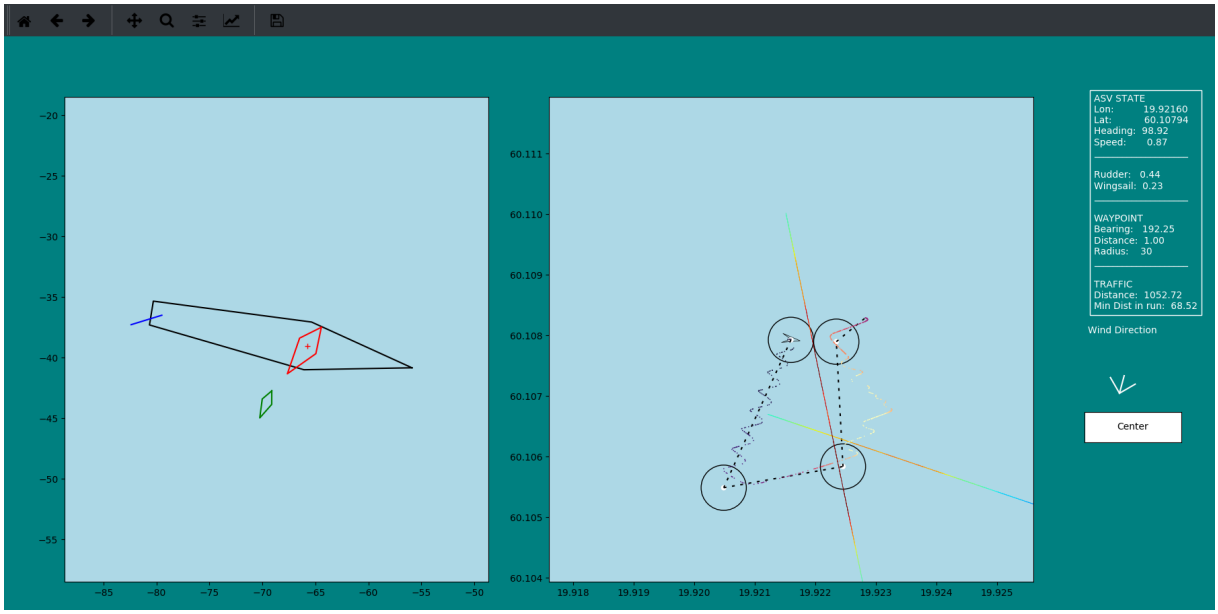

Figure 2.4: Proximity voter and midrange voter weights=1

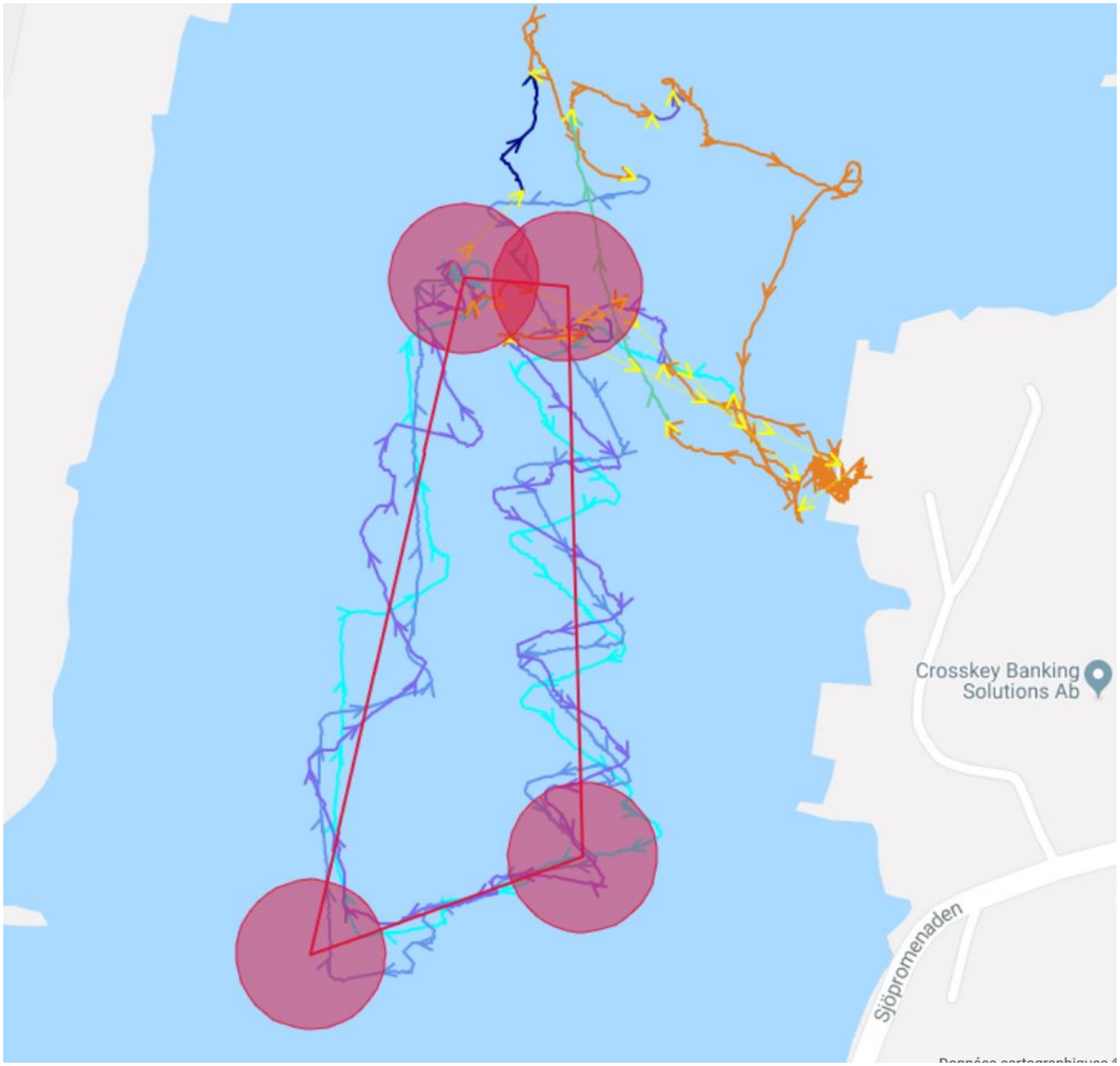

Figure 2.5: Navigation using line-follow algorithm (cyan and purple) and voter system (blue). Other colors correspond to manual mode when the boat is towed

Chapter 3

Work on the platforms

During this internship, I was mainly working on the ASPire boat, but also on a new boat for the World Robotic Sailing Championship in the last weeks.

3.1 Integration of the sensors on a CAN bus

I worked on the integration of the marine sensors and the CAN (Controller Area Network) bus used to plug in all the components in the boat. It is the same kind of technology used for cars for example, where you have to link an important number of components that need to communicate.
Each components sends and receive message using an ID and wrapping data into byte packets. One of the problem we had was the huge disparities between the structure and the way the data packets were filled. It made a lot of debugging processes harder because there was no sort of standard use for creating and filling a packet, and weird behaviour like having every data corrupted if only one component failed its writing.

3.2 Test

The ASPire has been launched multiple times during the summer, with the longest test being three days out in the south of the archipelago. The main goal of this longer sailing was to use the same kind of sensors the boat should use for detecting harbour porpoise, and having a better grasp on the power balance of the system in longer missions.
After these tests we found that the power balance of the boat is just under zero, and the rudder actuator was the cause of huge spikes in the power consumption. Changing the commands sent to the rudder, making them smoothier for example would be an effective way to lower the amount of these spikes.

The boat also had a hard time sailing in more adversary conditions, with more wind and more drift from the currents.

3.3 Preparation of a new sailbot for the WRSC

The end of my internship was matching with the participation to the World Robotic Sailing Championship, but because of technical problems the team could not travel with the ASPire boat so a smaller one, named Velvet, has been made for this purpose. It is a sailbot with a classic sail, around 1.5 meter long.
On this boat I had to prepare the hardware and software, taking into account the needs for the competition.

3.3.1 Hardware

For making the integration as easier as possible, there was no use of more advanced structures using CAN buses as it has been done with the ASPire boat. A CAN bus is really useful when having a large amounts of components that need to communicate but that was not the case for the Velvet, as we were going to use a minimal amount of hardware. The Velvet is equipped with only four sensors and two actuators, with an RaspberryPi3 as the embedded computer, one Arduino Uno for sensors reading, and a controller for servomotors with its multiplexer for using radio command inputs.
As the amount of hardware is low, I opted for a simple serial connection between the Arduino and Raspberry. In the end the Arduino job was only to send the wind sensor data to the Raspberry, so there was no risk at all of a saturated serial connection. The communication is pretty simple: the sensor data are read by the Arduino, sent as strings through the serial bus, that are tokenized on the Raspberry which is looking for a given sequence to spot which token is the actual data. Implementing it this way is useful because I can keep debugging informations going through the serial port in case I want to run a testing code for checking the sensor and Arduino status.
I integrated the exact same compass as the ASPire for easier integration, using an I2C connection. Same idea with the GPS, but using an USB connection. The camera for the Velvet was a regular RGB camera, either the RaspberryPi camera or a standard webcam with a USB connection.
For the batteries, one is exclusively used by the winch servomotor, which is using a lot more power than the rest of the system, and a 3S Lipo battery powers everything else.

3.3.2 Software

The software for the Velvet was quite tricky to set. On one part I wanted to use as much as what had been done for the ASPire, while using commands that are suited for sail boats and not wingsail boats. Taking the code base of the previous sailboat of the project and updating it with the overall changes made for ASPire was a good starting point. Then the difficulty increases when rewriting the sensor nodes and the controller nodes and setting up a database, while making everything fit in the current code architecture.
Concerning the image processing, as the boat is equipped with an RGB camera I had the choice to keep the current code, or take advantage of some features concerning the obstacle to be avoided during the competition. As it would be an orange obstacle I prepared some code for detecting single colour objects in case the algorithm used for the ASPire was not having good results.
The voters had to be changed also, in particular the avoidance voters, so I made one for the Velvet, replacing the proximity and midrange voters as this boat does not have any AIS installed. The processing also takes into account the wider field of view of these cameras compared to the thermal camera installed in the ASPire.

Conclusion

The obstacle avoidance using thermal camera can be improved a lot in the interpretation side, the first idea would be to use a clustering algorithm and then use the informations we have on the environment to separate objects from water. The meanshift filtering looks very good for this application, either considering the computation time or the output of the processing which is delimiting really well the different environment we have: sea, objects and land, sky.
On the voter system, having the avoidance voters use vetos is also an idea, but then the case of having all possible headings under a veto have to be taken into account. The sailing test are satisfactory as long as the boat is under good wind conditions but we can't really know if the problem is more about the software or the mechanics of the boat. The boat could use a complete mechanical review, especially the mast and the control sail, as some part looked severely worn.

Bibliography

[1] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1197–1203 vol.2, Sept 1999.

[2] Luc Jaulin and Fabrice Le Bars. A simple controller for line following of sailboats. In Colin Sauzé and James Finnis, editors, *Robotic Sailing 2012*, pages 117–129, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[3] Jordan Less'ard-Springett, Anna Friebe, and Maël Le Gallic. Voter based control system for collision avoidance and sailboat navigation. In Kjell Ivar Øvergård, editor, *Robotic Sailing 2017*, pages 57–68, Cham, 2018. Springer International Publishing.

[4] Dilip K. Prasad, C. Krishna Prasath, Deepu Rajan, Lily Rachmawati, Eshan Rajabally, and Chai Quek. Challenges in video based object detection in maritime scenario using computer vision. *CoRR*, abs/1608.01079, 2016.

Abstract — This report will describe my work as part of the Åland Sailing Robots research project during my internship in the Åland University of Applied Science. This project has been running since 2013, and the marine research platform is a three year project within it. The platform will be evaluated for marine sensor measurements and harbor porpoise monitoring in the Baltic Sea, as it is its last year now. I also worked on building a new autonomous sailbot for the participation to the World Robotic Sailing Championship.
My internship consisted in testing the current platform, integrating on progress features, and doing some improvement on algorithms. In this report I will focus mainly on image processing for detecting obstacle in a maritime environment, and a navigation algorithm based on a voter system.

The end of my internship matched with the participation to the World Robotic Sailing Championship, for which I worked on preparing another smaller sailboat.

Keywords : Image processing, obstacle avoidance, sailboat, navigation