Sorbonne Université - Master SAR M2

Internship

Sea surface current estimation from autonomous boat navigation data

Patrick Auger

September 2023

Contents

I would like to thank sincerely everyone that contributed to this internship.
First, M. Morin for having given me opportunities throughout discussions in marine robotics, and all SAR Master teaching team for numerous learnings.
I adress a special thanks to M. Jaulin, tutor of this intership for regular support, Submeeting invitation and overall discussions and propositions along this period.
Finally, thank you Fabrice, Simon, Damien, Pierre, and Quentin members from LabSTICC team for hardware, installation advices and various discussions marine robotics environment and during lab seminars. I have learned a lot in few months and this was easier with you all.

1 Introduction

In France, the national objective to develop floating wind turbines in order to contribute to energy transition has been a major concern since the 2010s. Marine robots, useful for installation or for maintenance, have to be on the edge. Therefore, this internship has been conducted in this context and its focus was on sea surface currents.

Sea surface current estimation plays a vital role in various marine applications, such as navigation, environmental monitoring, and rescue operations. Accurate and real-time knowledge of the current patterns can significantly enhance the performance and efficiency of autonomous marine vehicles.

The goal of this internship was to answer this issue : how to estimate sea surface current from an autonomous boat in a basic mission without hydrographic additional system ?

Two different approaches to estimate sea surface current have been conducted : a method with standard Least-Squares estimator and a probabilistic state estimation method based on an Extended Kalman Filter (EKF).

This internship has been conducted in the LabSTICC laboratory in Brest, France. Thanks to ENSTA Bretagne and national defense projects, LabSTICC can deploy multiple marine robots with specialized hardware and waterproofness tests with an instrumented pool.

2 Materials and methods

2.1 Hélios : architecture and sensors

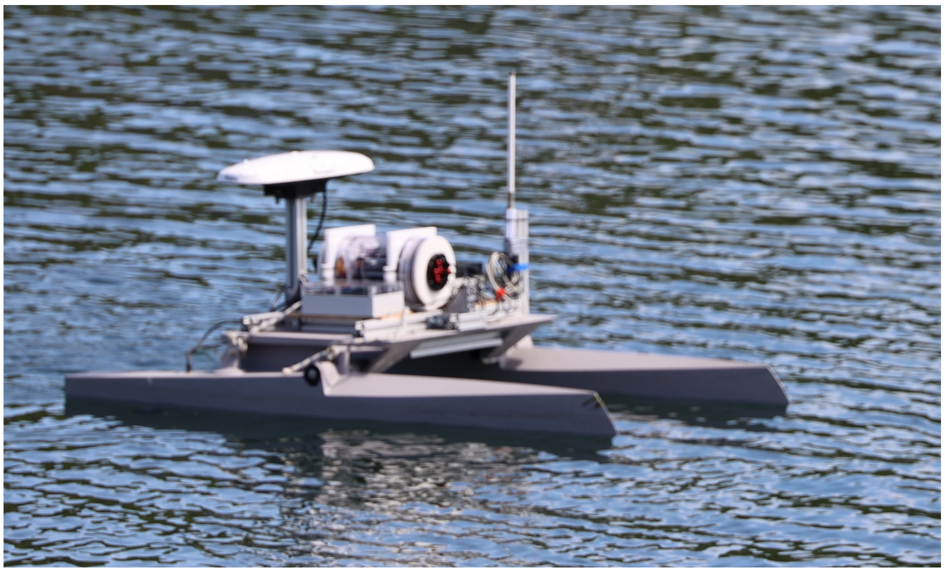


FIGURE 1: Hélios in Moulin Mer mission during IA and Ocean pole (LabSTICC), 2-3 May

Hélios is an instrumented catamaran that has performed as a marine drone for various missions: explore specific zone, test control algorithms or carry special echo sounder device. Its dimensions are 1.80m length, 1.20m width and approximately 0.20m height.

Hélios is equiped with a central linux NUC that is the heart of the drone, a Global Navigation Satellite System (GNSS) unit, and propelled by two pairs of T200 thrusters from BlueROV. GNSS position measurement (latitude, longitude) conversion in (x,y) coordinates is done with a pyproj Lambert projection transformer.

To make Helios work, energy sources are from 2 '4S' Li-Po batteries (with 14.8 V nominal voltage and 1000 mAh capacity) : one for thrusters motors and the wifi system and one for the NUC unit.

This enables anyone with a remote controller and a PC (with connection and launching procedures tackled in Appendix) the boat to launch specific mission, store mission data...

2.2 Hélios propulsion

After having presented Helios general architecture, this section presents how thrusters are controlled.

FIGURE 2: T200 thrusters used for Helios propulsion

Two pairs of thruster motors have been implemented on Hélios. Each one is speed-controlled by Electronic Speed Controllers (ESC).

With constructor documentation and previous work on Hélios, thrusters are controlled with PWM principle triggered by an Arduino interface.

As presented in the control laws section, the setpoint for thruster control is an angle in the range [-120°, +120°] (from left to right, by convention) and is transformed in an unsigned int stored on 8 bits (so that cover a range from 0 to 256)..

The thrusters limits choice is "PWM -100" for maximum thruster rotation in the reverse direction, and "PWM +100" for maximum thruster rotation in the forward direction.

So, for security reasons, here are the choices made for final control :

- if the objective is to turn left (negative setpoint s) : one wants to set more power in the right motor : the right motor control is set to a default value under maximum power (80) while the left one is set to a lower value (80+s),
- if the objective is to turn right (positive setpoint s) : one wants to set more power in the left motor : the left motor control is set to a default value under maximum power (80) while the right one is set to a lower value (80-s).

Note : with this choice, for situations in which Hélios has a heading far from global waypoint, highest angles can cause one of the two pairs of motors evolve in opposite rotation direction.

2.3 Boat evolution model

This section presents the evolution model and the reasons that led to these choices to describe Helios evolution in marine environment. The evolution model is taken from Dubin's [1] car model in addition to sea surface current effect :

$$\begin{cases} \dot{x} = v\cos\theta + c_x \\ \dot{y} = v\sin\theta + c_y \\ \dot{\theta} = (u_L - u_R)\dot{\alpha} \end{cases}$$

Where x, y are the position of Helios, v is its longitudinal speed due to propulsion thrusters, θ its heading, and c_x, c_y respectively horizontal and vertical component of sea surface current.
This model takes into account the sea surface current that can drift Hélios from its heading natural trajectory
u_L and u_R represent the left and right thruster contributions to heading evolution. α is for arbitrary minimum radius of curvature.
This model neglects the variation of the longitudinal speed from propulsion thrusters.
This model also neglects wind effect on Hélios structure that can cause pitch and roll variation.

2.4 Sea surface current model

Sea surface current modelization is based on an ellipsoidal hypothesis to consider tidal effects : it varies smoothly (the inertia is driven by water flow) on the time and space. Here is a proposition :

$$\begin{bmatrix} c_x \\ c_y \end{bmatrix} = \mathbf{R}_\alpha \begin{bmatrix} a_1\cos\left(\frac{2\pi t}{T} + \phi\right) \\ a_2\sin\left(\frac{2\pi t}{T} + \phi\right) \end{bmatrix}$$

Where $\mathbf{R}_\alpha$ is the rotation matrix that represent the orientation of the ellipse,

$$\mathbf{R}_\alpha = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

a_1, a_2 are the longitudinal and radial amplitudes of the ellipse,
T is the period of current cycles : tidal considerations [2] state that $T = 12\text{h } 25\text{min}$
ϕ is a tunable phase angle.

2.5 Control laws

In this section, the control laws that can fit Hélios missions have to be implemented on-board.
First, in order to justify a traditional 2D map study, here is how Helios GNSS measurement data is converted in x,y coordinates : In the file `conversion_gnss_lambert.py`, with pyproj transformer :

```
transformer=Transformer.from_crs(4326,2154,always_xy=True)
```

Where 4326 is for the World Geographic Coordinate System EPSG 4326 (longitude, latitude) and 2154 is for France Lambert geodesic 2D map (x,y) : EPSG 2154.
The transformation between (longitude, latitude) and (x,y) is called forward Lambert conformal conic LCC projection.

Two approaches are presented herein : either a set of waypoints (that can be a cycle) or a line-following algorithm.

2.5.1 Point following - waypoints

In some missions, Hélios has to follow a basic path between some waypoints $W = \{W_1, W_2, ..., W_{n_w}\}$ that act as checkpoints.

Between two waypoints, validation criteria is when Hélios has reached the target in a tolerance circle. Global algorithm stops when final waypoint is reached. The principle is illustrated here

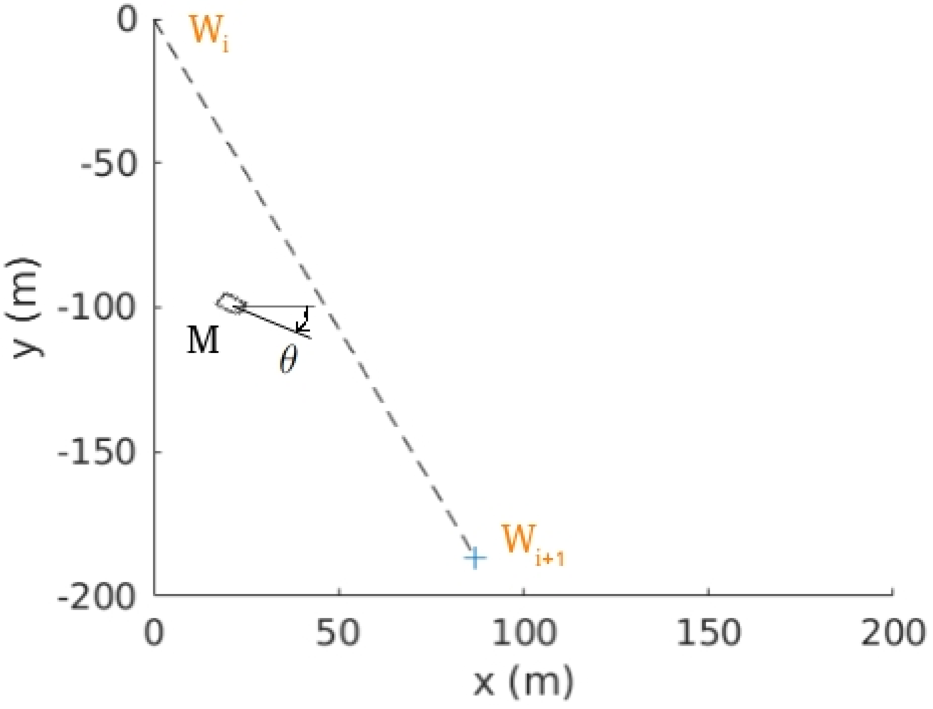


FIGURE 3: Point following - waypoints algorithm illustration

: Hélios the mobile robot M with state variables (x, y) 2D-coordinates and a heading θ wants to reach the target waypoint W_{i+1}. The ideal heading in this configuration is calculated with :

$$\theta_d = atan2(W^y_{i+1} - y, W^x_{i+1} - x)$$

So the angular setpoint δ can be obtained with the difference and the correcting factor k :

$$\delta = k \text{ sawtooth}(\theta_d - \theta)$$

This is a simple proportionnal controller with sawtooth security function to manage 2-π periodicity [3] :

$$\text{sawtooth}(\tilde{\theta}) = 2 \arctan\left(\tan\frac{\tilde{\theta}}{2}\right) = mod(\tilde{\theta} + \pi, 2\pi) - \pi$$

2.5.2 Line following

In marine context, point following algorithm can lead to instability (unreached target even with tolerance radius) due to important sea surface currents. The goal for Helios can be to follow a line between two reference points A and B.

The algebraic distance to line is calculated with :

$$e = \frac{\det\Big(m - a, b - a\Big)}{||\ b - a\ ||}$$

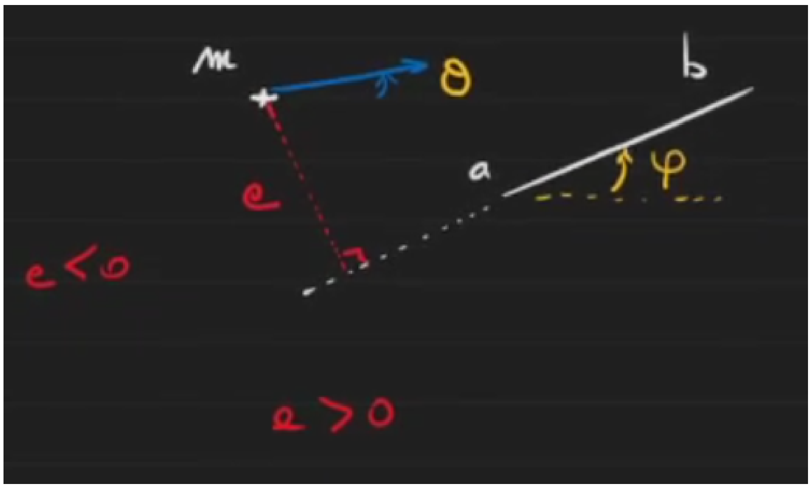


FIGURE 4: Line following illustration

To make Helios follow the line :
If $e \to \infty$ then $\theta_d = \varphi + \frac{\pi}{2}$ and if $e \to -\infty$ then $\theta_d = \varphi - \frac{\pi}{2}$

To avoid a chattering effect when approaching the line in practice, a confidence width r can be introduced to make sure the desired heading does not oscillate too quickly :

$$\theta_d = \varphi + \arctan\left(\frac{e}{r}\right)$$

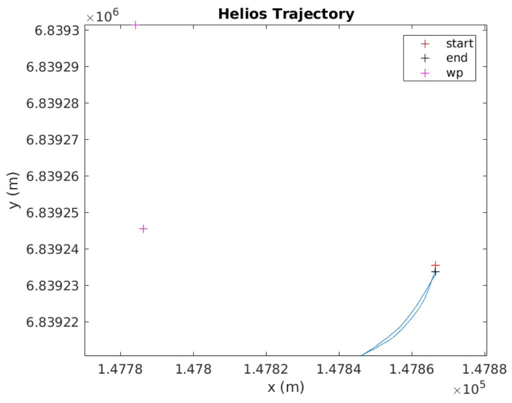
3 Results and discussions

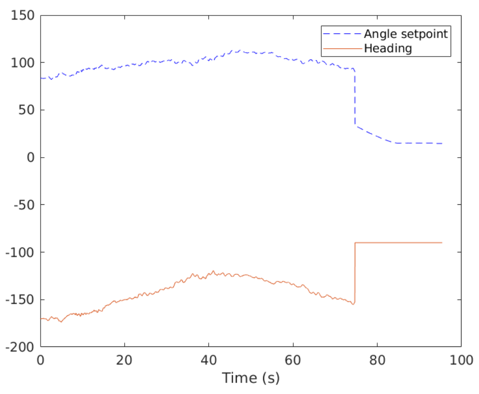
3.1 Experimental setup

3.1.1 Preparation

In order to prepare for missions in marine environment, first preparation tests have been conducted carrying Helios manually on a stadium next to LabSTICC to test data export, vizualisation and control law. Waypoints have been set with national geoportail tool. Recorded mission files (.bag) have been used to import mission data in MATLAB environment.

For instance, here are the Helios trajectory during the first test and corresponding heading and angle setpoint : in this configuration with selected waypoints, Helios needs to turn right which corresponds to this positive angle setpoint.





3.1.2 Guerlédan

Even though final sea surface current estimation was planned for Moulin Mer mission, Helios was used for real condition tests during Submeeting 2023 (April 24-28), a seminar gathering researchers in marine robotics and some marine startups. For example, Helios was used to perform visual odometry in marine environment for a team from Ecole Navale. A GoPro camera has been attached during a mission and informations on buoy landmark position have been shared for their project to make sure mission data was trustable.

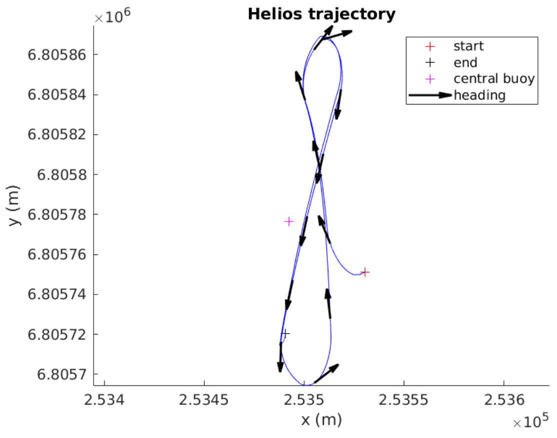


FIGURE 5: Visualisation of Helios trajectory and some headings

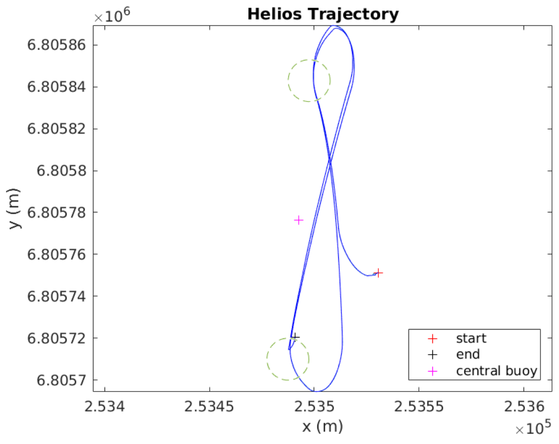


FIGURE 6: Visualisation of Helios trajectory with waypoints and tolerance circle

3.1.3 Moulin Mer

In Moulin Mer (May 2), Helios was ready for long missions to collect data in order to estimate sea surface current. Moulin Mer is near Brest, on the wester French coast. The initial goal was to make Helios navigate a whole afternoon but hardware issues have forced to stop mission after approximately 2 hours. 3 datasets are available : the morning test, and 2 afternoon samples. Here was the initial mission (waypoints have been changed due to traffic) :

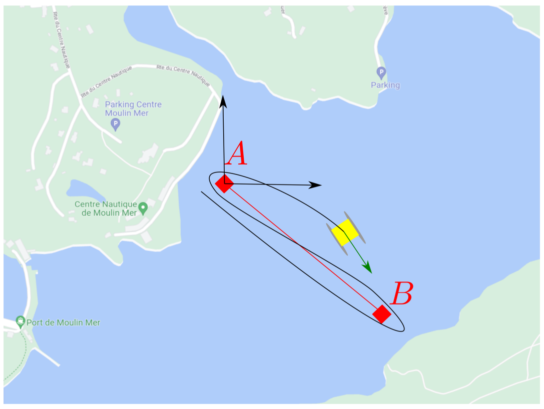
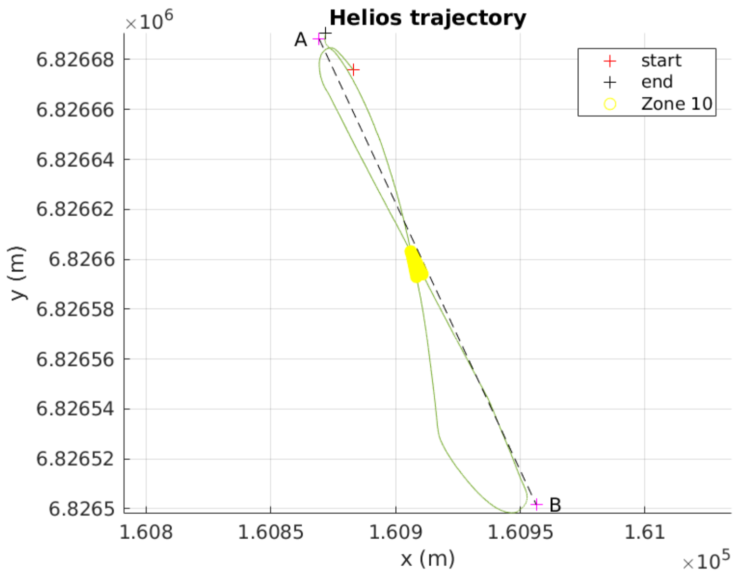


FIGURE 7: Mission plan for Moulin Mer

Because remote controlled and PC setup was way further than during Guerlédan tests, occasional connection loss was experimented.

Therefore, to make sure that estimation procedures were not biaised by fake data, a zone separator has been developed to avoid extreme points. Here are the middle zone highlighted points thanks to this separator with 20 zones for one of the Helios trajectories :



To compute Hélios speed over ground (SOG), two steps have been chosen : differentation and filtering. Filtering raw position data was unnecessary due to boat inertia dynamics (not same dynamics compared to e.g. flying AUV) and GNSS station accuracy (order of magnitude 10cm). After testing iterations, a butterworth 6-th order filter with a cutoff frequency of 0.05 Hz has been chosen. That can be surprising at first but since the boat is not meant to accelerate, that has given appropriate results :

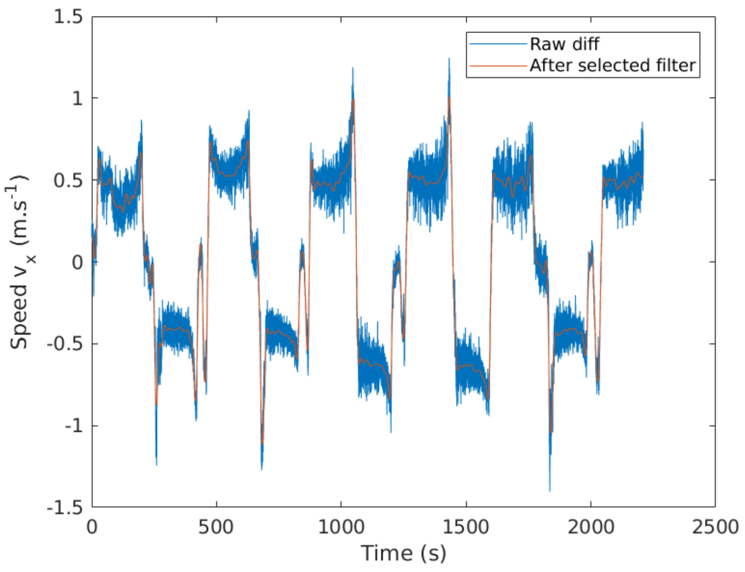


FIGURE 8: Computed v_x speed before and after filtering during last Moulin Mer mission

3.2 Estimation procedures

3.2.1 Least Squares Estimator (LSE)

The goal of this method is to make a first estimate of the sea surface current based on the shift between heading and speed vector direction during Hélios mission. Its simplicity has drawbacks : this method is unefficient when dealt with very few points and or very few directions.

The speed over ground (SOG) is defined as :

$$SOG = \sqrt{\dot{x}^2 + \dot{y}^2}$$

The course over ground (COG) is defined as :

$$COG = atan2(\dot{y}, \dot{x})$$

The differential bearing ϕ is defined as :

$$\phi = \theta - COG$$

Then, the radial speed (not caused by thruster propulsion) is :

$$v_{rad} = SOG \sin(\phi)$$

With boat evolution model, the current to be identified is the only cause to radial speed :

$$v_{rad} = c^T u_\theta^O$$

Where $c = [c_x, c_y]^T$ the current $u_\theta = [\cos\theta, \sin\theta]^T$ and $u_\theta^O = [-\sin\theta, \cos\theta]^T$ its orthogonal vector.

The estimation algorithm is based on estimate the optimal $\hat{c}$ that fit the most with linear model $Y = A\hat{c}$ with stacked data :

$$Y = \begin{bmatrix} v_{rad_i} \\ \cdots \end{bmatrix} \;,\; A = \begin{bmatrix} (u_{\theta_i}^O)^T \\ \cdots \end{bmatrix}$$

The Least Squares best estimate is obtained with the Moore pseudo-inverse :

$$\hat{c} = (A^T A)^{-1} A^T Y$$

To evaluate the performance of this estimator, the corresponding residuals $\hat{r}$ and covariance matrix P can be defined as :

$$\hat{r} = A\hat{c} - Y$$

$$P = \frac{\hat{r}^T \hat{r}}{n_{obs} - n_p} (A^T A)^{-1}$$

The calculated COG highly depends on the quality of speed computation. Indeed : if the cutoff frequency is too high, radial speed will be blurred into noise.

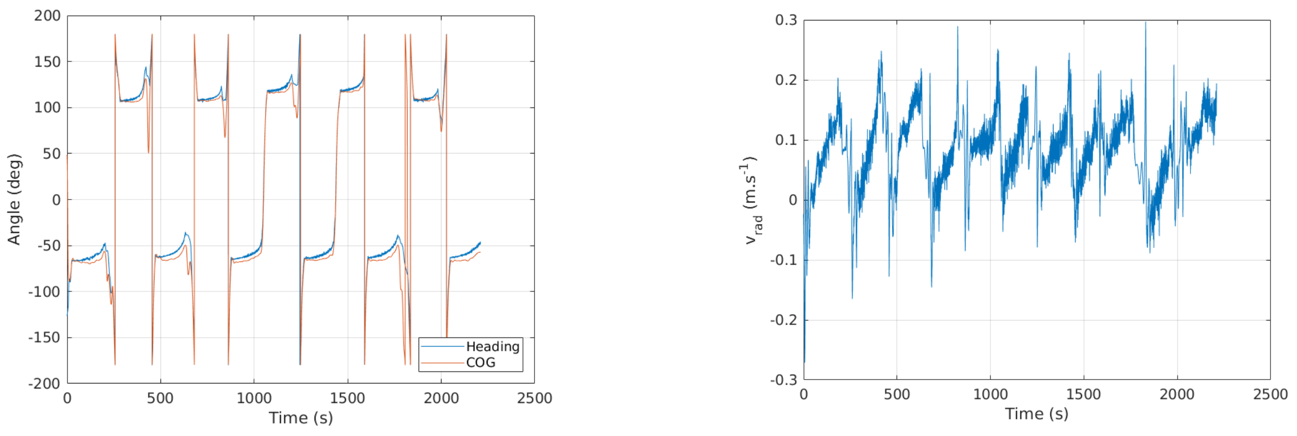


FIGURE 9: COG verification and radial speed during last Hélios mission in Moulin Mer

Gathering the two last missions on Moulin Mer data (the first one was mainly for testing purposes), here are the estimation results :

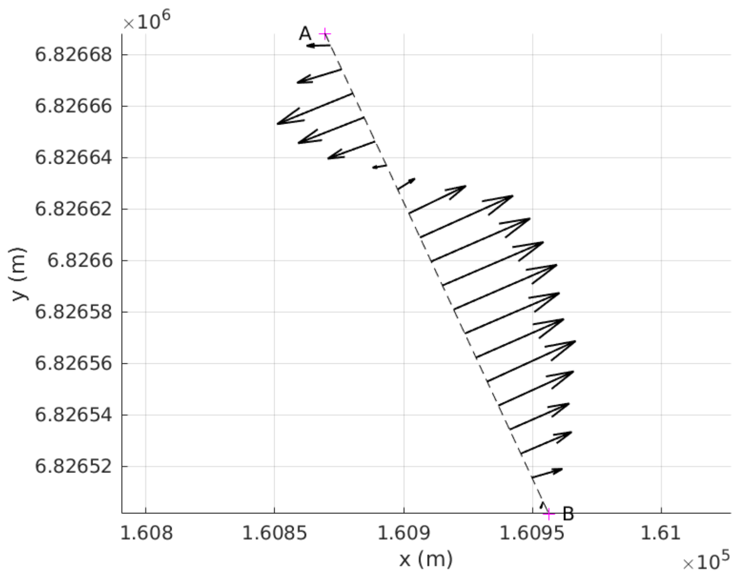


FIGURE 10: Current estimation result for each zone between the two waypoints

Due to low heading dispersion, current seems to be highly overestimated (the reference was 0.1 m/s in north-east direction) :

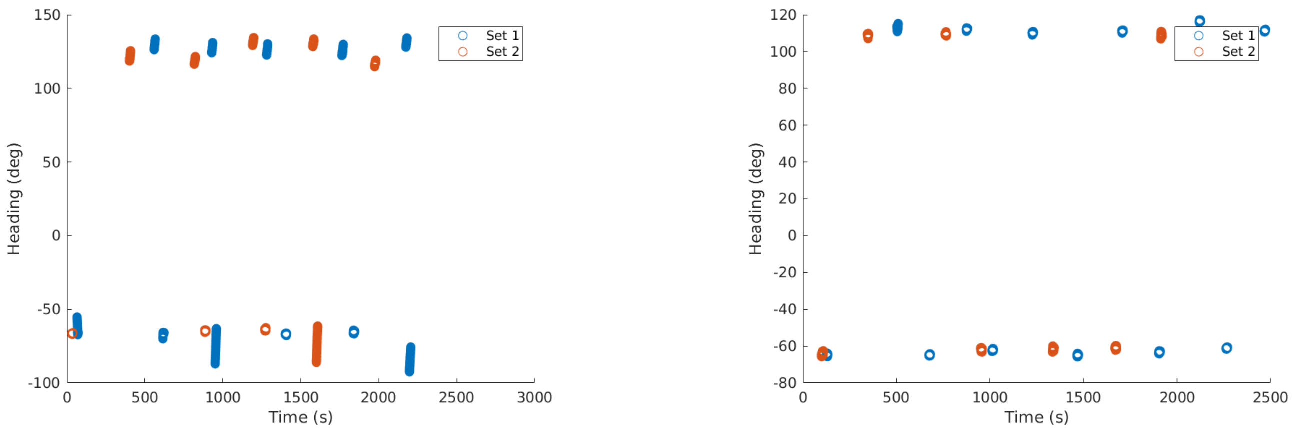


Figure 11: Heading dispersion for zone 2 and 10 during the 2 last Moulin Mer missions

The estimation covariance is high when the estimated current at lowest amplitude. Indeed, that can come from low radial speed in this section.

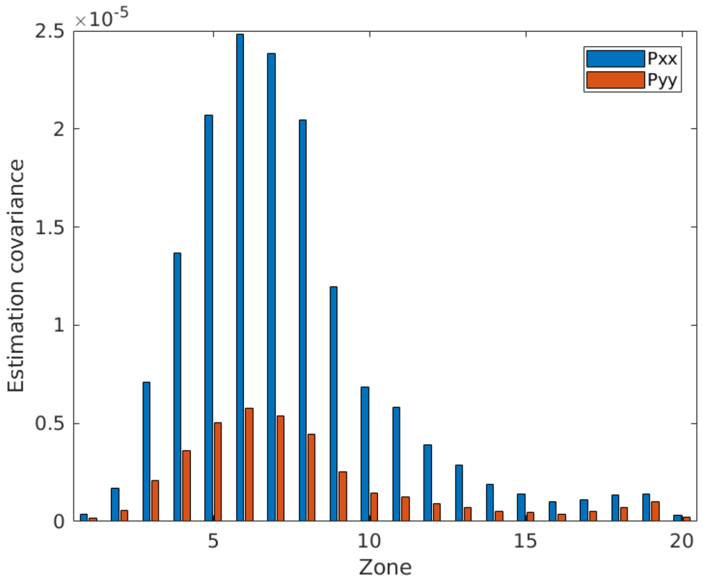


Figure 12: Estimation covariance for each zone

3.2.2 Kalman Estimator

Another method to estimate current is applying Kalman approach. Indeed, the first step is to build augmented Hélios state with current to be estimated. Since the evolution model is non-linear, Extended Kalman Filter (EKF) has been chosen for this section. However, due to connection loss around extreme point, this estimator cannot be performed on an entire dataset but has better tuning potential.

The augmented Hélios state is $x_{EKF} = [x, y, \theta, c_x, c_y]^T$, with same notations from previous sections. The augmented state evolution equation is :

$$\begin{cases} \dot{x} = v_0 \cos\theta + c_x + \zeta_x \\ \dot{y} = v_0 \sin\theta + c_y + \zeta_y \\ \dot{\theta} = \zeta_\theta \\ \dot{c}_x = \zeta_c \\ \dot{c}_y = \zeta_c \end{cases}$$

Where ζ_i represents model noise evolution for each variable.
The measurement is limited to position and heading so $y = Cx_{EKF}$ where $C = \begin{bmatrix} I_{3\times 3} & O_{3\times 2} \end{bmatrix}$
The evolution matrix F_k was obtained by direct Jacobian :

$$F_k = I_{5\times 5} + \begin{bmatrix} 0 & 0 & -v_0 \sin\theta & 1 & 0 \\ \vdots & \vdots & v_0 \cos\theta & 0 & 1 \\ \vdots & \vdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} dt$$

Initialization step
For this method, the initial augmented state is the initial measured position, heading and a null current.
Therefore, covariance associated with position and heading has to be a lot lower (higher confidence) than the covariance associated with current.

Model and measurements confidence
Q and R are the Kalman tuning parameters that respectively manage our confidence on model and measurements.
First, thinking about water dynamics makes our confidence in the current constant model high : this will be the lowest tuned coeficient by far.
Then for positions : a feedback from members in LabSTICC was that GNSS station measurements uncertainty was in the same order of magnitude as 10 cm. Kalman algorithm should apply a little higher confidence in the position evolution model (that makes heading and current appear) rather than in its position measurements.
Finally for the heading tunable parameters, the lack of control input in model is faced up with a visible heading noise. So the heading confidence parameters should be tuned as relatively even.

Prediction step

$$\hat{x}_{k+1|k} = F_k \hat{x}_{k|k}$$
$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q$$

Correction step with incoming measurement y_{k+1}

$$K_{k+1} = P_{k+1|k} C_{k+1}^T (C_{k+1} P_{k+1|k} C_{k+1} + R)^{-1}$$
$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(y_{k+1} - C\hat{x}_{k+1|k})$$

Kalman method results
A first application of this method with a fragment of first dataset gives the following results :

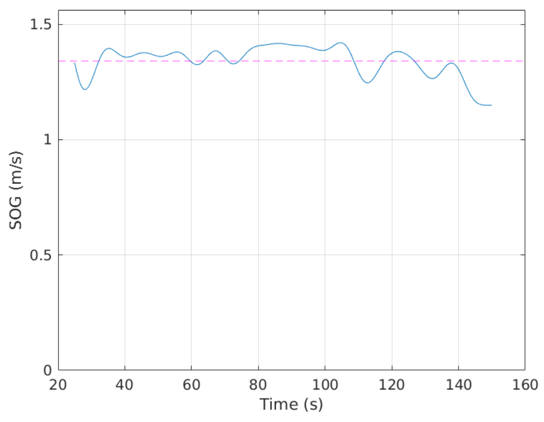


FIGURE 13: SOG result to check speed model hypothesis

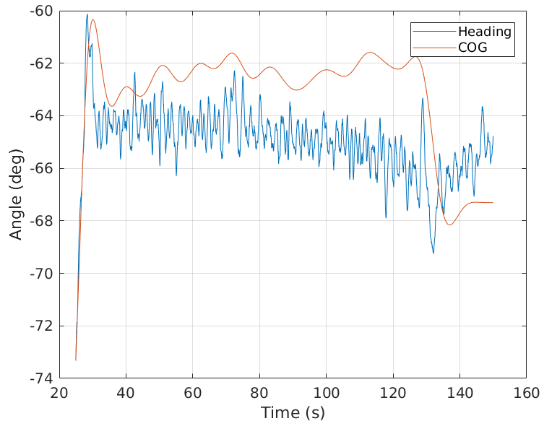
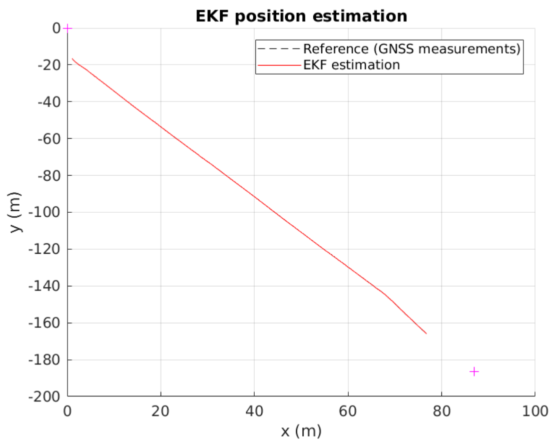


FIGURE 14: COG result to observe shift



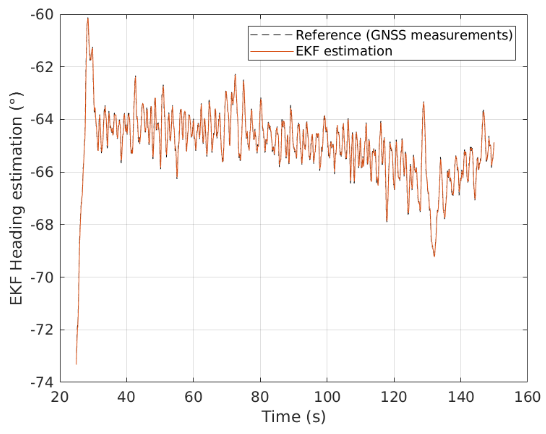
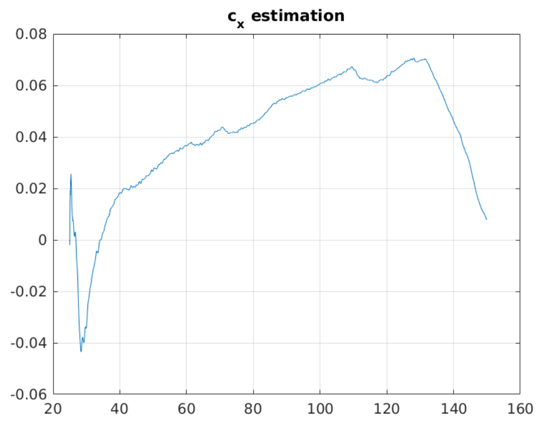


FIGURE 15: Position and heading estimates compared to measurements



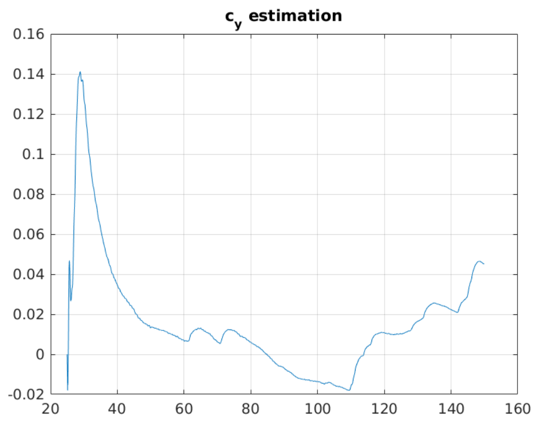


FIGURE 16: Current estimates (in m/s) during the selected period

The resulting current estimation is also north/north-east oriented as the previous estimator.

A second application of this method when Hélios is heading the opposite gives these results :

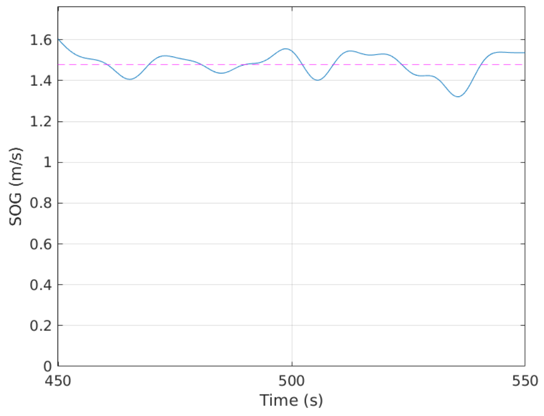


FIGURE 17: SOG result to check speed model hypothesis

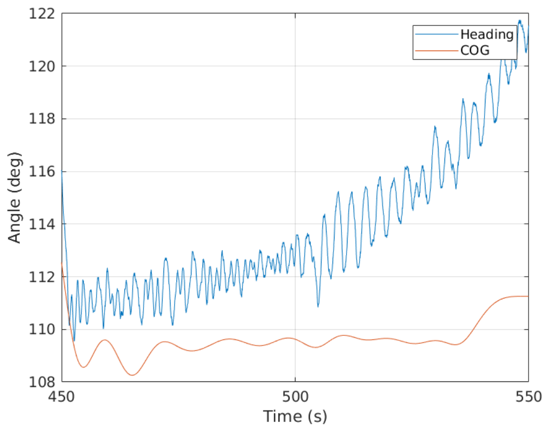
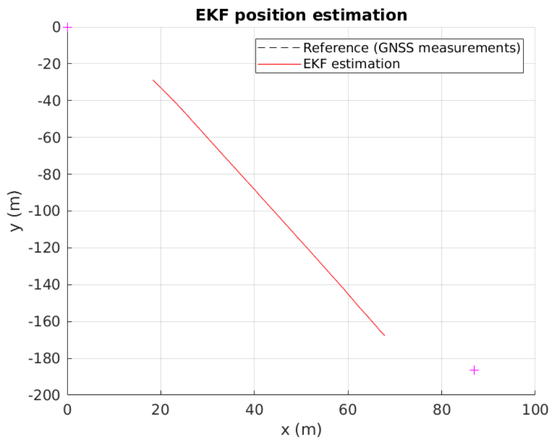


FIGURE 18: COG result to observe shift



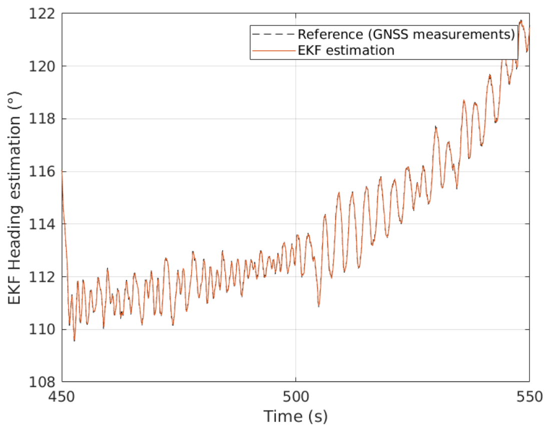
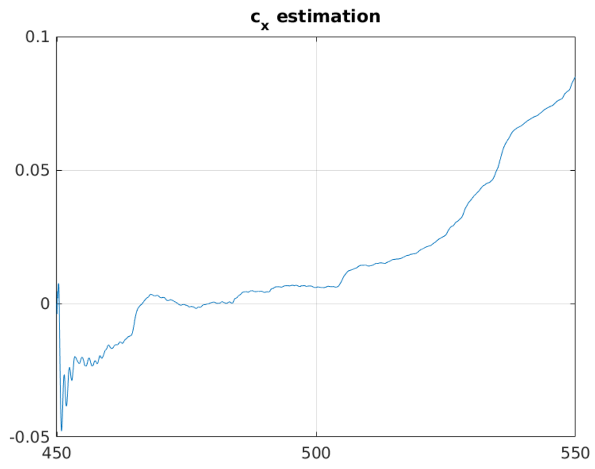


FIGURE 19: Position and heading estimates compared to measurements



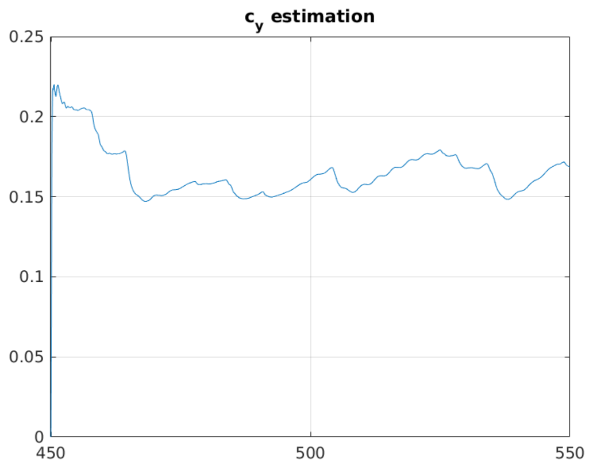


FIGURE 20: Current estimates (in m/s) during the selected period

This current estimation is more north oriented than the previous one. The estimated current has a higher order of magnitude, which is consistent with the observation of the differential between Hélios COG and Hélios heading.

4 Conclusions

As a final result, several sea surface current estimators with discussions on parameters have been created and tested on experimental data. Data filtering has also been presented. To consider feasible perspectives for this particular purpose, simulators that could generate boat trajectories with known currents could help to better evaluate estimator performances.

Furthermore, the experimental setup plays a major role in the quality of the estimations : if there exists an important timing section before or after launching an autonomous mission, this could let the boat derivate without any thruster propulsion and facilitate the estimation. Estimators with interval approaches have also been considered [4] [5] but have not given significant results.

5 Appendices

5.1 Thrusters T200 technical documentation

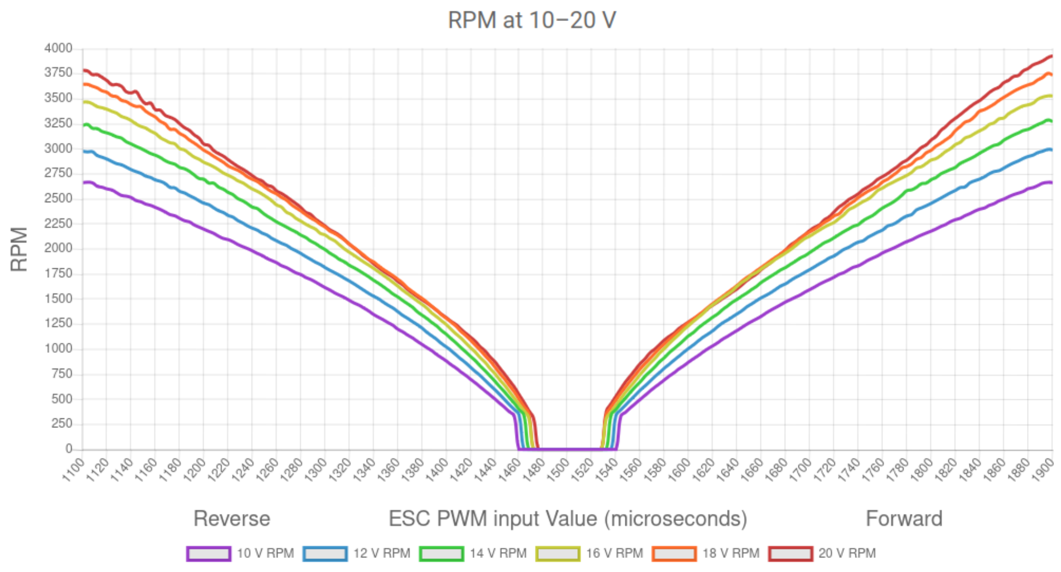


FIGURE 21: Rotation speed of T200 thrusters evolution with PWM input

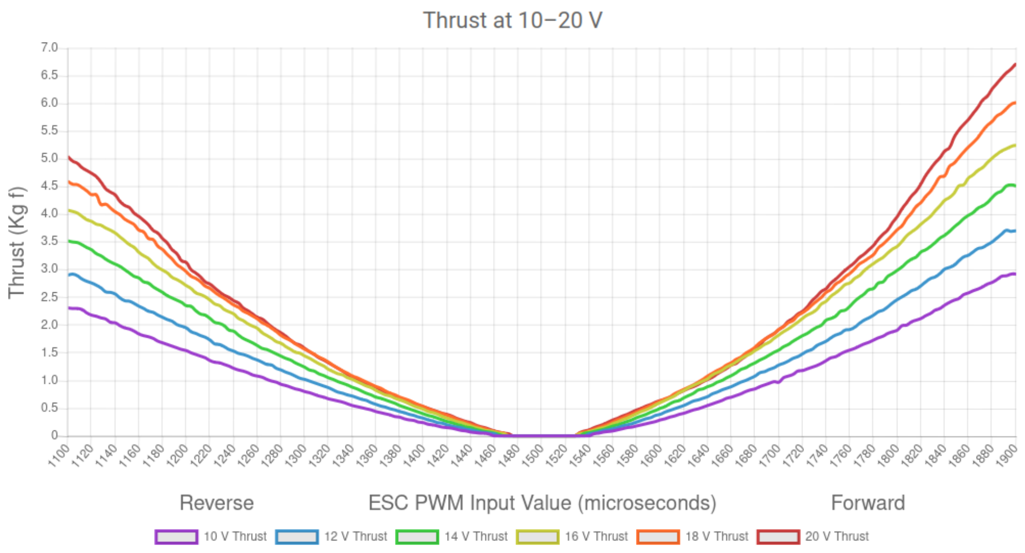


FIGURE 22: Thrust developed evolution with PWN input

5.2 Hélios launching procedure

Before mission :

- prepare mission with waypoints in the chosen following file (path or line following)
- change data save filenames in both gnss_v500_infos and following file,
- check the 3 motors, nuc, and remote controler batteries and its spare ones,
- connect the batteries, switch the motors batteries on, switch the magnetic for nuc,
- connect to munu-ubnt wifi
- establish an SSH connection helios_ros for more,
- copy paste mission files :
  1- .py files :

  ```
  scp <fichier_a_copier> s100@10.43.20.223:catkin_ws/src/s100/src/
  ```

  2- launch file(s) :

  ```
  scp <fichier_a_copier> s100@10.43.20.223:catkin_ws/src/s100/
  ```

During mission : prepare SSH terminal to :

- activate the GNSS station, send RTK corrections with the commands :

  ```
  narval_supply_control.py -e usbl
  send-rtk-corrections
  ```

- start a roscore,
- start the mission with the corresponding roslaunch file,
- start recording with *rosbag record -a*
- use polotjuggler for real-time vizualisation

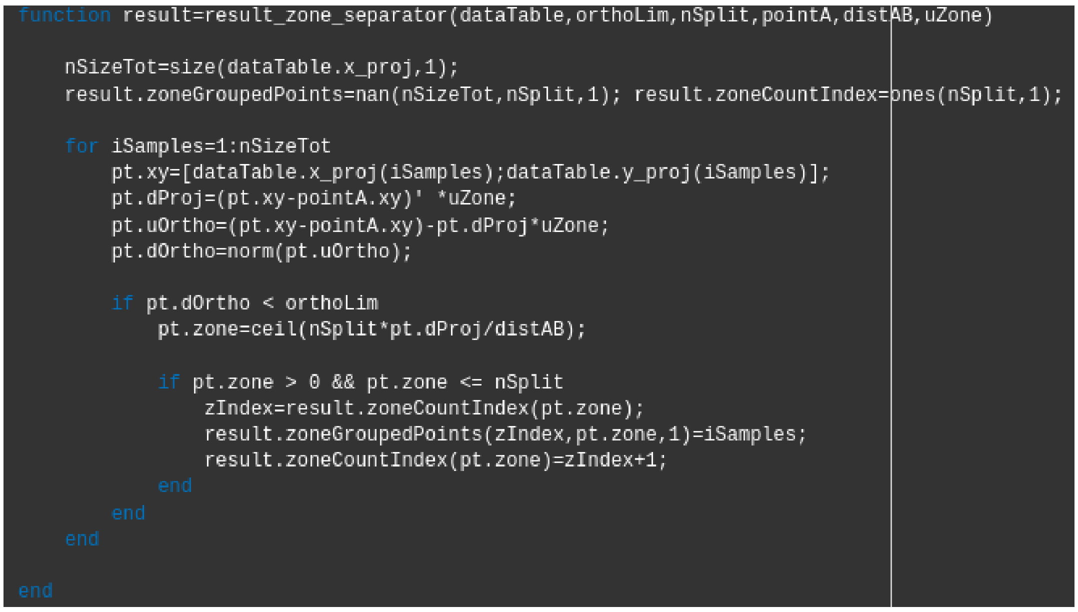
After mission :

- copy and paste mission files outside Helios NUC environment :

  ```
  scp s100@10.43.20.223:catkin_ws/src/s100/logs/<fichier_path>.txt .
  scp s100@10.43.20.223:catkin_ws/src/s100/logs/<fichier_traj>.txt .
  scp s100@10.43.20.223:<bag_file> .
  ```

- export path and trajectory .txt files in GPX with 'create_gpx.py' with adapted filenames

5.3 Zone separator algorithm



```
function result=result_zone_separator(dataTable,orthoLim,nSplit,pointA,distAB,uZone)

    nSizeTot=size(dataTable.x_proj,1);
    result.zoneGroupedPoints=nan(nSizeTot,nSplit,1); result.zoneCountIndex=ones(nSplit,1);

    for iSamples=1:nSizeTot
        pt.xy=[dataTable.x_proj(iSamples);dataTable.y_proj(iSamples)];
        pt.dProj=(pt.xy-pointA.xy)' *uZone;
        pt.uOrtho=(pt.xy-pointA.xy)-pt.dProj*uZone;
        pt.dOrtho=norm(pt.uOrtho);

        if pt.dOrtho < orthoLim
            pt.zone=ceil(nSplit*pt.dProj/distAB);

            if pt.zone > 0 && pt.zone <= nSplit
                zIndex=result.zoneCountIndex(pt.zone);
                result.zoneGroupedPoints(zIndex,pt.zone,1)=iSamples;
                result.zoneCountIndex(pt.zone)=zIndex+1;
            end
        end
    end

end
```

FIGURE 23: Zone separator code using MATLAB struct

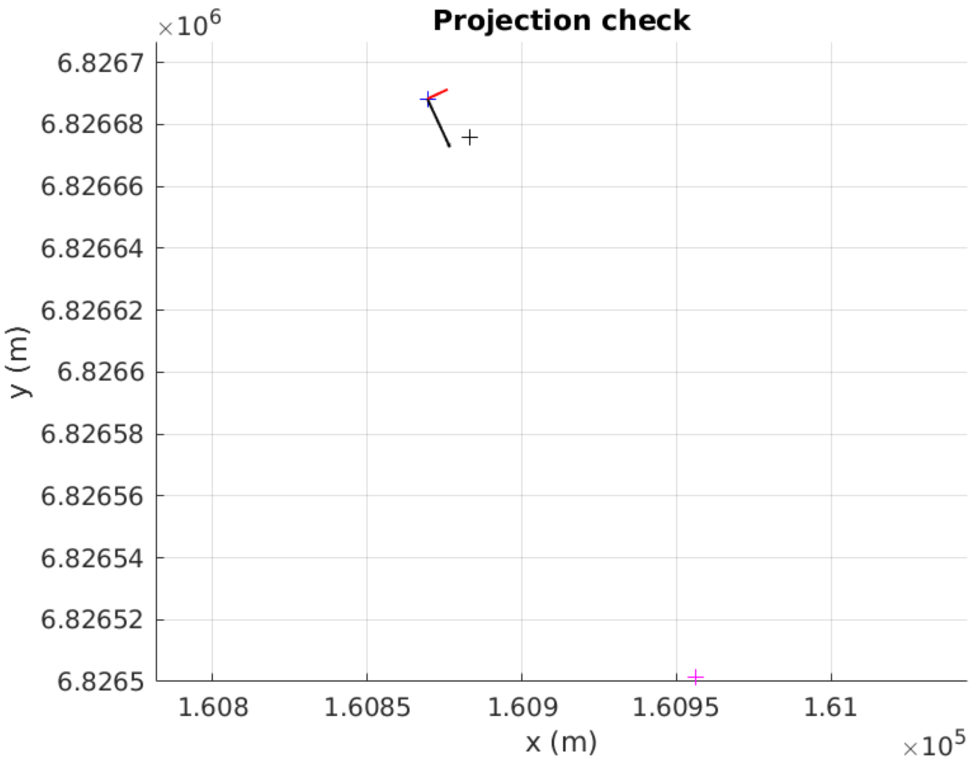
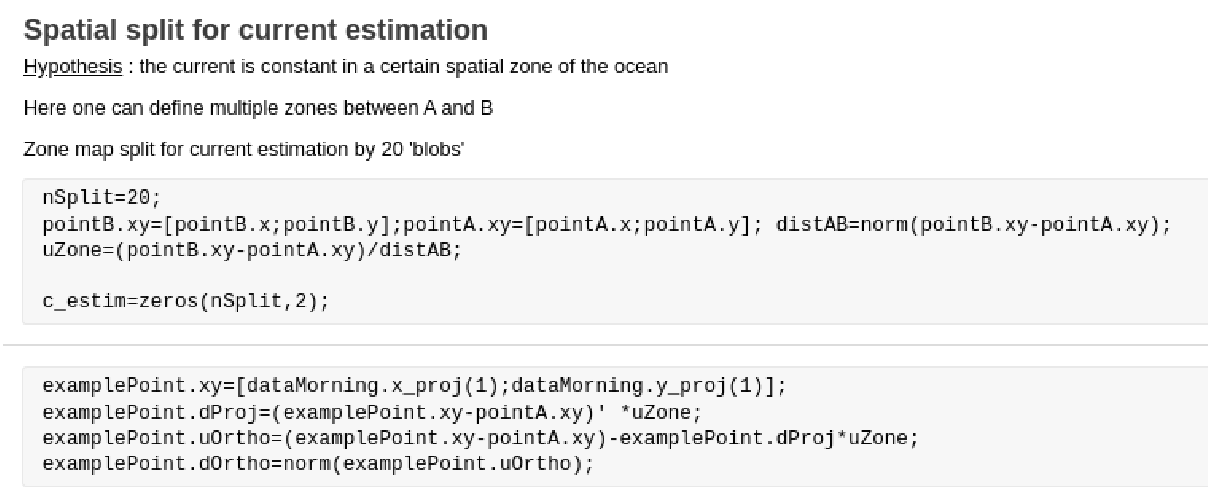


FIGURE 24: Projection check for zone separator

5.4 Core LS estimator code



Spatial split for current estimation

Hypothesis : the current is constant in a certain spatial zone of the ocean

Here one can define multiple zones between A and B

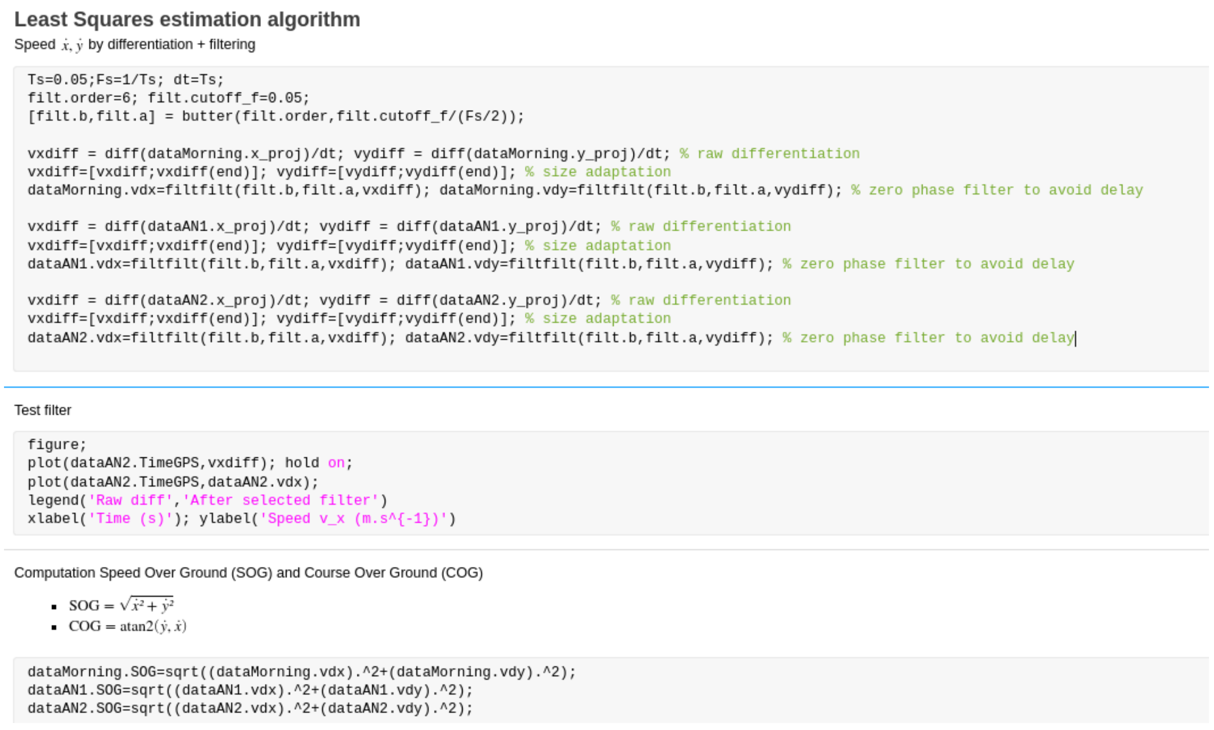
Zone map split for current estimation by 20 'blobs'

```
nSplit=20;
pointB.xy=[pointB.x;pointB.y];pointA.xy=[pointA.x;pointA.y]; distAB=norm(pointB.xy-pointA.xy);
uZone=(pointB.xy-pointA.xy)/distAB;

c_estim=zeros(nSplit,2);
```

```
examplePoint.xy=[dataMorning.x_proj(1);dataMorning.y_proj(1)];
examplePoint.dProj=(examplePoint.xy-pointA.xy)' *uZone;
examplePoint.uOrtho=(examplePoint.xy-pointA.xy)-examplePoint.dProj*uZone;
examplePoint.dOrtho=norm(examplePoint.uOrtho);
```

FIGURE 25: Projection code



Least Squares estimation algorithm

Speed $\dot{x}, \dot{y}$ by differentiation + filtering

```
Ts=0.05;Fs=1/Ts; dt=Ts;
filt.order=6; filt.cutoff_f=0.05;
[filt.b,filt.a] = butter(filt.order,filt.cutoff_f/(Fs/2));

vxdiff = diff(dataMorning.x_proj)/dt; vydiff = diff(dataMorning.y_proj)/dt; % raw differentiation
vxdiff=[vxdiff;vxdiff(end)]; vydiff=[vydiff;vydiff(end)]; % size adaptation
dataMorning.vdx=filtfilt(filt.b,filt.a,vxdiff); dataMorning.vdy=filtfilt(filt.b,filt.a,vydiff); % zero phase filter to avoid delay

vxdiff = diff(dataAN1.x_proj)/dt; vydiff = diff(dataAN1.y_proj)/dt; % raw differentiation
vxdiff=[vxdiff;vxdiff(end)]; vydiff=[vydiff;vydiff(end)]; % size adaptation
dataAN1.vdx=filtfilt(filt.b,filt.a,vxdiff); dataAN1.vdy=filtfilt(filt.b,filt.a,vydiff); % zero phase filter to avoid delay

vxdiff = diff(dataAN2.x_proj)/dt; vydiff = diff(dataAN2.y_proj)/dt; % raw differentiation
vxdiff=[vxdiff;vxdiff(end)]; vydiff=[vydiff;vydiff(end)]; % size adaptation
dataAN2.vdx=filtfilt(filt.b,filt.a,vxdiff); dataAN2.vdy=filtfilt(filt.b,filt.a,vydiff); % zero phase filter to avoid delay
```

Test filter

```
figure;
plot(dataAN2.TimeGPS,vxdiff); hold on;
plot(dataAN2.TimeGPS,dataAN2.vdx);
legend('Raw diff','After selected filter')
xlabel('Time (s)'); ylabel('Speed v_x (m.s^{-1})')
```

Computation Speed Over Ground (SOG) and Course Over Ground (COG)

- $\text{SOG} = \sqrt{\dot{x}^2 + \dot{y}^2}$
- $\text{COG} = \text{atan2}(\dot{y}, \dot{x})$

```
dataMorning.SOG=sqrt((dataMorning.vdx).^2+(dataMorning.vdy).^2);
dataAN1.SOG=sqrt((dataAN1.vdx).^2+(dataAN1.vdy).^2);
dataAN2.SOG=sqrt((dataAN2.vdx).^2+(dataAN2.vdy).^2);
```

FIGURE 26: Filter design and testing code

5.5 Core Kalman estimator code

```
nState=5; % x,y,psi,cx,cy
dataEKF=table(t_tar,x_tar,y_tar,psi_tar);
dataEKF.Properties.VariableNames={'TimeGPS','x','y','heading'};
```

Initialisation

```
P00_pos=1; P00_psi=10*pi/180; P00_c=100;
P00=diag([P00_pos,P00_pos,P00_psi,P00_c,P00_c]); Pkk=P00;

cx0=0;cy0=0;
x00=[dataEKF.x(1);dataEKF.y(1);dataEKF.heading(1);cx0;cy0];
C=eye(3,nState);
v0=1.34;

% For formatting storage (this will be overwritten) to keep initial state estimation apart
result.x_hat(1,:)=x00';
result.diagPkk(1,:)=diag(P00);
xkk=x00;

% For struct lisibility
EKFParams.P00=P00; EKFParams.x00=x00;
```

Covariance Q <-> Confidence in model

- positions : 10cm,
- heading : 10 deg (has to be >> than the one corresponding to heading measurement)
- current : order of magnitude >> 0.2 m/s (feedback from hydro team)

```
EKFParams.Q=diag([0.1,0.1,10*pi/180,0.1e-3,0.1e-3]);
```

Covariance R <-> Confidence in measurements

- positions : 10cm GNSS station+corrections RTK (feedback from Fabrice, Pierre),
- heading : 1 deg default, to be tuned

```
EKFParams.R=diag([100e-2,100e-2,10*pi/180]);
```

(Recurrence)

Prediction

```
    psik=xkk(3); vk=xkk(4);
    Fk=eye(nState,nState); Fk(1:2,4:5)=eye(2)*dt;
    Fk(1:2,3)=[-v0*sin(psik);v0*cos(psik)]*dt;
    xdot=evolution_constant_speed(xkk);
    xk1=xkk + xdot*dt;
    Pk1=Fk*Pkk*Fk' + EKFParams.Q;
```

Correction

Kalman gain

```
    K=Pk1*C'/(C*Pk1*C' + EKFParams.R);
```

Import measurement

```
    yk1=[dataEKF.x(k+1);dataEKF.y(k+1);dataEKF.heading(k+1)];
```

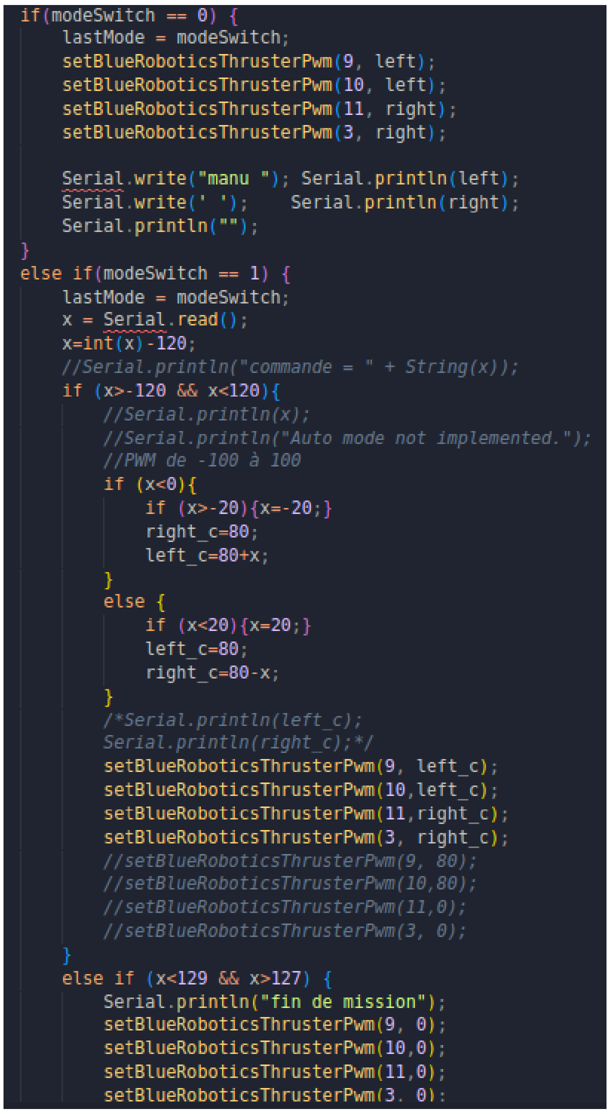
Apply correction with Kalman gain

```
    xkk=xk1 + K*(yk1-xk1(1:3));
    Pkk=(eye(nState) - K*C)*Pk1;
```

Storage

```
    result.x_hat(k+1,:)=xkk';
    result.diagPkk(k+1,:)=diag(Pkk)';
    result.K(k+1,:,:)=K;
end
```

5.6 Core Arduino code giving PWM input for thrusters



```
if(modeSwitch == 0) {
    lastMode = modeSwitch;
    setBlueRoboticsThrusterPwm(9, left);
    setBlueRoboticsThrusterPwm(10, left);
    setBlueRoboticsThrusterPwm(11, right);
    setBlueRoboticsThrusterPwm(3, right);

    Serial.write("manu "); Serial.println(left);
    Serial.write(' ');    Serial.println(right);
    Serial.println("");
}
else if(modeSwitch == 1) {
    lastMode = modeSwitch;
    x = Serial.read();
    x=int(x)-120;
    //Serial.println("commande = " + String(x));
    if (x>-120 && x<120){
        //Serial.println(x);
        //Serial.println("Auto mode not implemented.");
        //PWM de -100 à 100
        if (x<0){
            if (x>-20){x=-20;}
            right_c=80;
            left_c=80+x;
        }
        else {
            if (x<20){x=20;}
            left_c=80;
            right_c=80-x;
        }
        /*Serial.println(left_c);
        Serial.println(right_c);*/
        setBlueRoboticsThrusterPwm(9, left_c);
        setBlueRoboticsThrusterPwm(10,left_c);
        setBlueRoboticsThrusterPwm(11,right_c);
        setBlueRoboticsThrusterPwm(3, right_c);
        //setBlueRoboticsThrusterPwm(9, 80);
        //setBlueRoboticsThrusterPwm(10,80);
        //setBlueRoboticsThrusterPwm(11,0);
        //setBlueRoboticsThrusterPwm(3, 0);
    }
    else if (x<129 && x>127) {
        Serial.println("fin de mission");
        setBlueRoboticsThrusterPwm(9, 0);
        setBlueRoboticsThrusterPwm(10,0);
        setBlueRoboticsThrusterPwm(11,0);
        setBlueRoboticsThrusterPwm(3, 0);
```

References

[1] L. E. Dubins, “On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents,” *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957, ISSN: 00029327, 10806377. [Online]. Available: http://www.jstor.org/stable/2372560.

[2] R. Mazumder and M. Arima, “Tidal rhythmites and their implications,” *Earth-Science Reviews*, vol. 69, no. 1-2, pp. 79–95, 2005.

[3] L. Jaulin, *Mobile Robotics*. Wiley, 2019, ISBN: 9781119663492. [Online]. Available: https://books.google.fr/books?id=OMWxDwAAQBAJ.

[4] B. Desrochers, “Set-Membership Approach to Map-Based Localization,” Tech. Rep., 2014. [Online]. Available: https://hal.science/hal-01065126.

[5] S. Rohou, B. Desrochers, and L. Jaulin, “Set-membership state estimation by solving data association,” May 2020, pp. 4393–4399. DOI: 10.1109/ICRA40945.2020.9197039.