Autonomous UAV surveillance of a ship's path with MPC for Maritime Situational Awareness

Fabio A. A. Andrade[1,2,3], Rune Storvold[1,2], Tor Arne Johansen[1]

[1]Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway
[2]Remote Sensing, Satellites and UAS, Norut Northern Research Institute, Tromsø, Norway
[3]Center of Technical and Scientific Education, Brazilian Naval Academic, Rio de Janeiro, Brazil
fabio@ieee.org, rune.storvold@norut.no, tor.arne.johansen@ntnu.no

***Abstract*—Maritime Situational Awareness is crucial in maritime operations to identify threats and to deal with them as soon as possible. These threats can be pirates in shipping operations, icebergs when sailing in the northern sea routes, or even unknown vessels or objects that might be on the ship's path. A solution to identify these threats is the use of UAV's to overfly the ship's planned path. This solution is described in this paper, using an autonomous fixed wing UAV. Based on the provided ship's planned path, the UAV should autonomously map the area close to the ship track. To do that, an optimization problem is solved using Model Predictive Control, where the turn rate for the next time period is optimized. Based on the turn rate, the future path of the UAV is calculated and the waypoints are sent to the autopilot. This application is thoroughly tested using a Software In Loop environment, where an aircraft model is used with the autopilot's simulation. The results show that surveillance performance is improved if the UAV has information about the ship's velocity in addition to position.**

Keywords—UAV; Path Planning; Model Predictive Control

I. INTRODUCTION

The purpose of Maritime Situational Awareness [1] is to develop the ability to identify existing threats as early and as far as possible. This is done by integrating intelligence, surveillance, observation, and navigation systems, all interacting in the same operational framework. For this capability to be effective, it is necessary to build a structure covering data collection and monitoring using air and naval sensors, and accurate analyzes of the data, allowing rapid and accurate response.

To build this capability, UAV's are well suited tools that can be equipped with a wide variety of sensors, such as cameras or radars. The cameras can be infrared, RGB, or multispectral. One application of these sensors is the tracking of floating elements in the sea [2], like drifting objects, enemy vessels, icebergs and others.

This information can be used in collision avoidance systems [3], where the vessel will find a new path based on the obstacles' positions; military operations [4], where the threats or targets can be identified earlier; ice management [5], to map the features of ice or icebergs drifting into the path of the vessel; and search and rescue [6], where the vessel needs to find missing crew.

In seismic data collection, vessels tow arrays of streamers that might be up to one kilometer wide and 5 kilometer long. When towing these arrays the vessel must keep steady direction and speed, hence to prevent damage to the streamers the area upstream must be inspected for icebergs and growlers and in case of suspected damage to streamers the array and cables themselves need to be inspected while moving [7].

Usually, the vessel is the center of the maritime operation, which as exemplified before, can be a shipping mission, a search and rescue, or a military surveillance operation. To collect the needed data to support the chosen operation, the UAV must fly over a region according to the vessel's planned path and speed, and in the case of scouting for growlers, combined with knowledge on i.e. potential drift velocities, in a cooperative way ensuring safety margins on the captured data

Therefore, in the situation proposed in this paper, a fixed wing UAV must fly over the ship's planned path at a certain distance ahead of the vessel to allow for time for the ship's captain to timely react to findings. For instance, if it is needed some information four minutes in advance, the UAV has to overfly the predicted position of the ship after four minutes from now. For the UAV to decide its optimum path to cover that area, an optimization problem has to be solved to minimize the error between the predicted and the desired position that the UAV must be. To solve this problem, using a Model Predictive Control (MPC) is beneficial, because it can consider the predicted output, based in current measurements, to fit the control inputs in a better manner. In this problem, the intended velocity of the vessel can be used to predict its future positions in order to decide the UAV's optimum path.

There are some researches in the use of MPC techniques in the field of UAVs, as in [8], where a UAV was used to follow a linear path. However, it did not take into account time constraints, so the problem was treated as a path following problem. [9] also shows a solution to track moving objects, but it optimizes the path according to the waypoints sequence, which has much more computational cost than the solution presented in this paper, where only the turn rate is controlled. In [10], a MPC algorithm running in a ground station is used to

control the UAV turning radius and the camera's gimbal to track objects in the sea. The solution in this paper brings a similar approach but to a different problem, also treating the sensors information, e.g., the angular velocities, in a more beneficial way to achieve more realistic results.

The presented system, with the challenge of integrating those different software components and to make it possible to use it in an onboard real time optimization problem, introduces an effective solution for many types of autonomous tracking problems, bringing a beneficial result especially when the motion of the tracking object is provided.

The UAV's path is optimized with MPC computing its turn rate at a constant altitude. The optimization problem is implemented using ACADO for MATLAB, a user-friendly interface of the ACADO Toolkit, which is a software environment and algorithm collection for automatic control and dynamic optimization, implemented in C++ [11]. MATLAB is used to hold the main software, which does all the integration between the simulated ship's data, the MPC and the autopilot.

ArduPilot[1] is an open source autopilot that supports many types of robotic vehicles, including the fixed wing UAV used in this application. It is possible to send commands to the ArduPilot or to read the UAV sensors via Micro Air Vehicle Communication Protocol (MAVLink[2]). The ArduPilot has also a Software In The Loop simulator that uses a flight dynamics model to get the sensor data. The JSBSim[3] was used for the flight dynamics model with the configuration of the X8, a fixed wing UAV for long range surveillance which can fly up to 3 hours.

To communicate with the autopilot, the messages travel between MATLAB and the ArduPilot though DUNE[4] [12], which is an open source robot framework developed by the Underwater Systems and Technology Laboratory (LSTS) of the University of Porto. Inter-Module Communication Protocol (IMC[5]) is then used for the messages between MATLAB and DUNE using the IMC Java library.

Details of the system architecture are described in the next section.

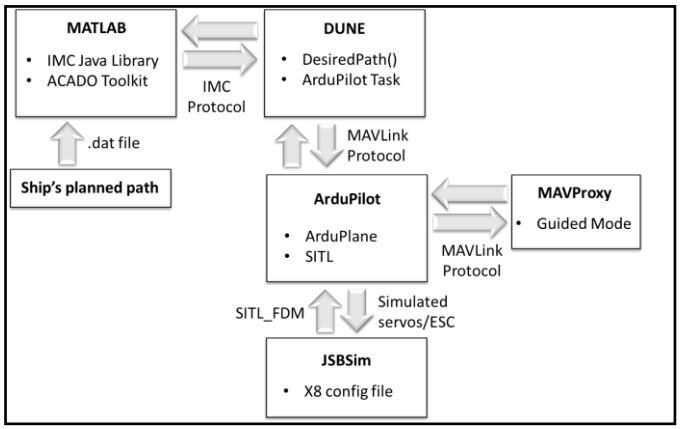

Fig. 1. System's block diagram, simulation setup.

[1] ardupilot.org
[2] mavlink.org
[3] jsbsim.org
[4] lsts.fe.up.pt/toolchain/dune
[5] lsts.fe.up.pt/toolchain/imc

II. System Description

Figure 1 shows the block diagram with the connections between all software components used in this system.

The main input is the simulated ship's planned path, where the position and velocity in the NED coordinate frame along the time were saved into a MATLAB data file. MATLAB also receives information from DUNE about the UAV state and uses it, combined to the ship's planned path, to solve the optimization problem and send the waypoints to DUNE. DUNE, in its turn, sends the waypoints to the ArduPilot and gets the information about the UAV sensors. The ArduPilot commands the UAV, which has its dynamics simulated by JSBSim. Besides, a MAVProxy ground station is also used to control the UAV in GUIDED mode if needed.

A. MATLAB Core Code

The data file is loaded by the core code and the starting position of the ship is taken as the origin of the NED frame. Therefore, the correspondent latitude and longitude is defined. In sequence, the update rate of the waypoints is defined and also the time in advance when the ship should receive information from the UAV about its path. This time is used to calculate where the ship might be according to its planned path and then use that information as the desired location where the UAV should overfly.

Position, velocity and attitude of the UAV are also necessary to be used as inputs to solve the two-dimension optimization problem, which assumes that the UAV will fly at a constant altitude. Therefore, the following information about the UAV is gotten from DUNE:

- Origin of its local NED frame (lat, lon, altitude);
- Position offset (x, y, z) of its NED frame origin;
- Body-Fixed frame 2D linear velocities (u, v);
- Euler angles (Roll (ϕ), Pitch (θ), Yaw (ψ)); and
- Angular velocities over body-fixed frame (p, q, r).

As the starting position of the ship is used in this application as the NED frame origin, it is needed to convert the UAV position in its local NED frame to the application's NED frame. To do that, first it is made a conversion to geodetic coordinates (latitude and longitude) using the origin of the UAV's Local NED frame and then it is made a conversion to the application's NED frame using the ship's starting position as the origin. The origin of the UAV's Local NED frame is updated every one kilometer of distance that it moves from the previous origin. Besides, for all conversions, the World Geodetic System of 1984 (WGS 84) was used as the reference ellipsoid.

Regarding the yaw rate, as a two-dimensional model is used in this optimization problem and the body-fixed angular velocity r, which is received from DUNE, is not the rate referring to the UAV's yaw angle, the yaw rate has to be calculated using the relationship between the Euler-angle rates vector $[\dot{\phi}, \dot{\theta}, \dot{\psi}]^{\mathrm{T}}$ and the body-fixed angular velocity vector $[p, q, r]^{\mathrm{T}}$ as (1) [13].

This is also needed because the UAV used in this application does not have a rudder and it uses roll to turn. Therefore, in the two-dimensional model described in the next section, the yaw rate r used is actually the yaw rate $\dot{\psi}$ obtained from (1).

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \sec\phi\sec\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \quad (1)$$

Finally, after solving the optimization problem, the target waypoint is sent to DUNE using the "DesiredPath" function. In that function it is also possible to send a second waypoint as a backup that can be used in case there is a temporary communication problem.

The overall description of the Core Code is shown in Figure 2. The program repeats every chosen update time step and it runs until the last planned position of the ship is reached by the UAV.

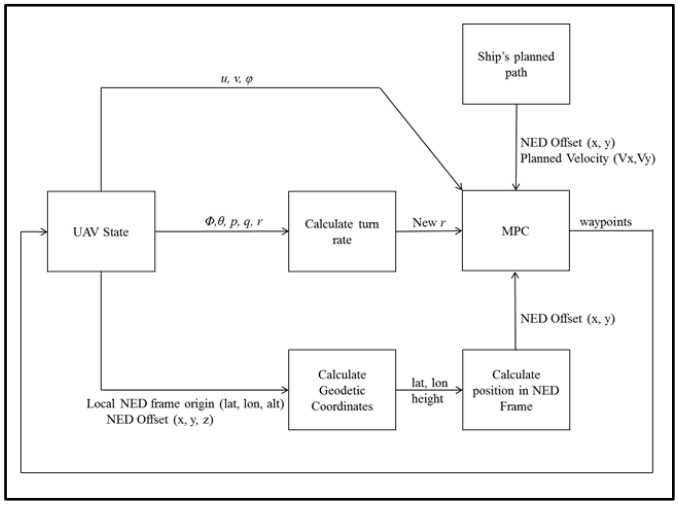

Fig. 2. Overall Core Code description.

B. IMC Java library

To send or receive IMC messages, IMC Java library is required to provide the necessary functions to MATLAB. Besides, in that protocol, the communication is done between nodes. Therefore, the core code has to be started as a IMC node and then connect to the desired node, in this case, the X8 UAV node in DUNE.

C. DUNE

DUNE is compatible with many different autopilots. In this application, as the ArduPilot is used, when DUNE receives the "DesiredPath()" command, it calls the Task responsible for the communication with the ArduPilot. However, the standard Task available in DUNE's repository operates the UAV in GUIDED Mode. In that mode, the UAV starts to loiter when it reaches a certain distance from the UAV and never pass through the waypoint. This behavior would make impossible the proposed application.

Fig. 3. X8 UAV. (*Source: NTNU*)

Therefore, a change was made in DUNE's ArduPilot Task, where the UAV could be operated in AUTO Mode. In the new Task, a mission containing the waypoint is sent to the autopilot. After the acknowledgement from the autopilot, a command to start the mission is then sent.

D. ArduPilot Software In The Loop

The ArduPilot works together with JSBSim. JSBSim simulates the behavior of the UAV in a real flight. The X8 UAV (Figure 3) model [14] was used, taking into consideration the mass balance, ground reactions, propulsion, aerodynamics, buoyant forces, external forces, atmospheric effects and/or gravity.

III. Model Predictive Control

Model Predictive Control (MPC) is a class of techniques to solve numerical optimization problems, controlling the input variable to minimize the deviations between a desired output and the predicted output in a finite time-horizon [15].

In this application, the first assumption to be made is that the fixed wing UAV is controlled by an Autopilot system, which needs to be fed with the waypoints of the optimal flight trajectory. The MPC module is responsible to calculate this trajectory, based on the UAV's and Ship's attitude, position and velocity.

A. UAV Dynamics

Considering that the fixed wing UAV will fly in a constant altitude, maintained by the Autopilot, its dynamics is treated as two-dimensional motion problem (Figure 4). Therefore, the position and heading of the UAV can be expressed in the NED frame as

$$\boldsymbol{p_{UAV}} = [x_{UAV}, y_{UAV}, \psi]^{\mathrm{T}}, \quad (2)$$

where x_{UAV} and y_{UAV} are the horizontal positions and ψ is the yaw angle. The velocities in the BODY frame and the rate of change of the yaw angle forms the following vector.

$$\boldsymbol{v_{UAV}} = [u, v, r]^{\mathrm{T}}. \quad (3)$$

Besides, the relation between (2) and (3) is given by

$$\dot{\boldsymbol{p}}_{\boldsymbol{UAV}} = \mathbf{R}_{\boldsymbol{z}}(\psi)\boldsymbol{v_{UAV}}, \quad (4)$$

where $\mathbf{R}_z(\varphi)$ is the rotation matrix between the BODY and NED frame as shown in (5).

$$\mathbf{R}_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

The constraint of the system is the maximum turning rate (r) of the UAV.

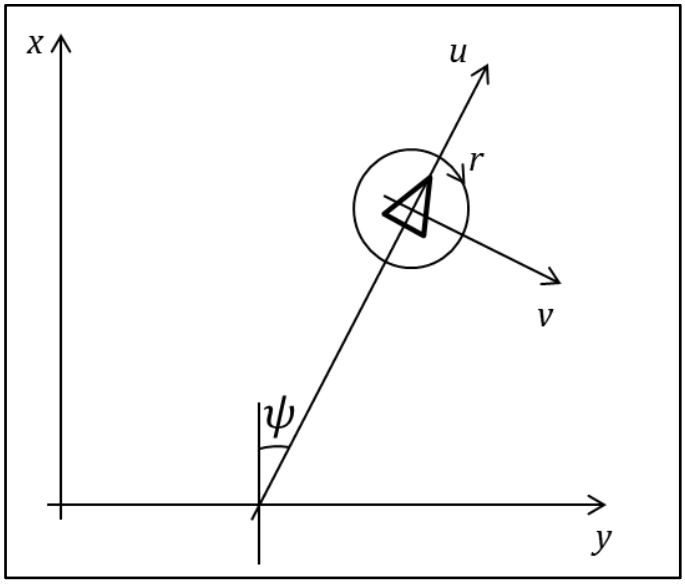

Fig. 4. UAV model in two dimensions.

B. Ship Dynamics

To use the ship dynamics in the MPC, a linear motion model is implemented based on the position and velocity of the ship in its intended path, at the instant the UAV should fly over that location.

$$\boldsymbol{p}_{\boldsymbol{ship}}^{t+\tau} = \boldsymbol{p}_{\boldsymbol{ship}}^{t} + \boldsymbol{v}_{\boldsymbol{ship}}^{t}\tau \quad (6)$$

where $\boldsymbol{p}_{\boldsymbol{ship}} = [x_{ship}, y_{ship}]^{\mathrm{T}}$ is the position of the ship on the path, τ is the time between measurements and the velocity $\boldsymbol{v}_{\boldsymbol{ship}} = [v_x, v_y]^{\mathrm{T}}$.

C. Cost Function

The Least Squares (LS) function (7) is used as the function to be minimized by the MPC algorithm.

$$J_t = \frac{1}{2}\sum_{i=t}^{t+T} \|h(\boldsymbol{z}_i)\|^2 \quad (7)$$

where $\boldsymbol{z}_k = \left[x_{UAV\,k}, y_{UAV\,k}, x_{ship_k}, y_{ship_k}\right]^{\mathrm{T}}$ is the state vector.

In this application, the LS function $h(\boldsymbol{z}_k)$ is the distance between the UAV and the future ship's position in its planned path as shown in (8).

$$\boldsymbol{h}(\boldsymbol{z}_k) = \sqrt{\left(x_{UAV\,k} - x_{ship_k}\right)^2 + \left(y_{UAV\,k} - y_{ship_k}\right)^2} \quad (8)$$

IV. Simulated Data

Two simulated datasets with the ship's path were generated. Both have the duration of 15 minutes.

The first dataset was generated with fixed heading of -2π/3 in NED frame and fixed velocity of 18 knots.

The second has the same starting heading and velocity of the first one but it changes the direction two times. After going straight for some time, it makes a slight turn to the right, then it goes straight again and finally it turns left and it goes straight for the rest of the time. It also changes its velocity from 18 knots to 21 knots half way.

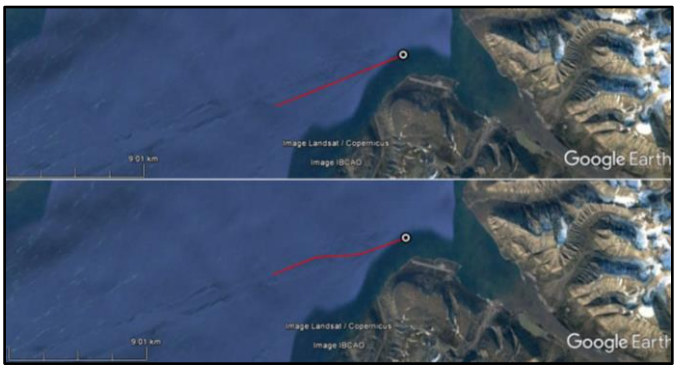

Fig. 5. Two simulated ship's path

Both datasets were generated for a simulation of a ship navigating close to Longyearbyen, Svalbard (Figure 5).

V. System Configuration

The following parameters were chosen when running the application:

- 2 minutes between the ship's current position and the desired position in its planned path to be overflown;
- 4 seconds as the update period for the UAV waypoints;
- 20 seconds of MPC Horizon;
- 10 MPC input steps;
- 6 seconds between the current position and waypoint;
- Maximum of 30 iterations in the MPC optimization;
- Maximum turn rate of 0.3 rad/s;
- 18 m/s of UAV velocity; and
- 500m of altitude.

The reason to choose 4 seconds between each update is that if the time step is too small, the autopilot may set the waypoint as reached before the next update. As the cruise speed of the X8, which is 18 m/s, was chosen as the target velocity, if the UAV is flying straight, 4 seconds between each update means that the UAV will fly 72 meters until the next update.

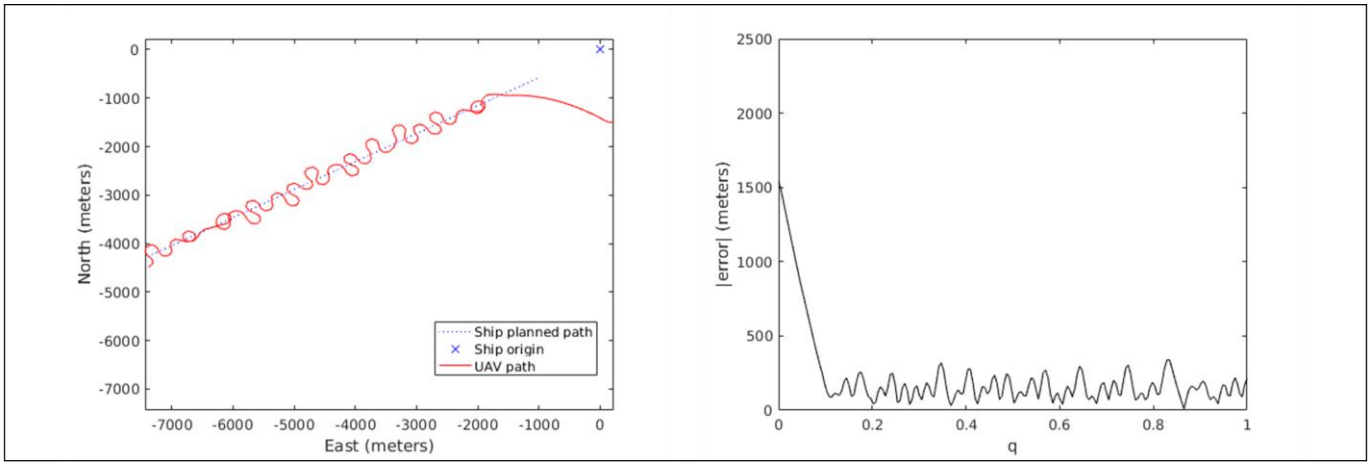

Fig 6. UAV path and error for constant ship velocity and heading using the ship model in the MPC

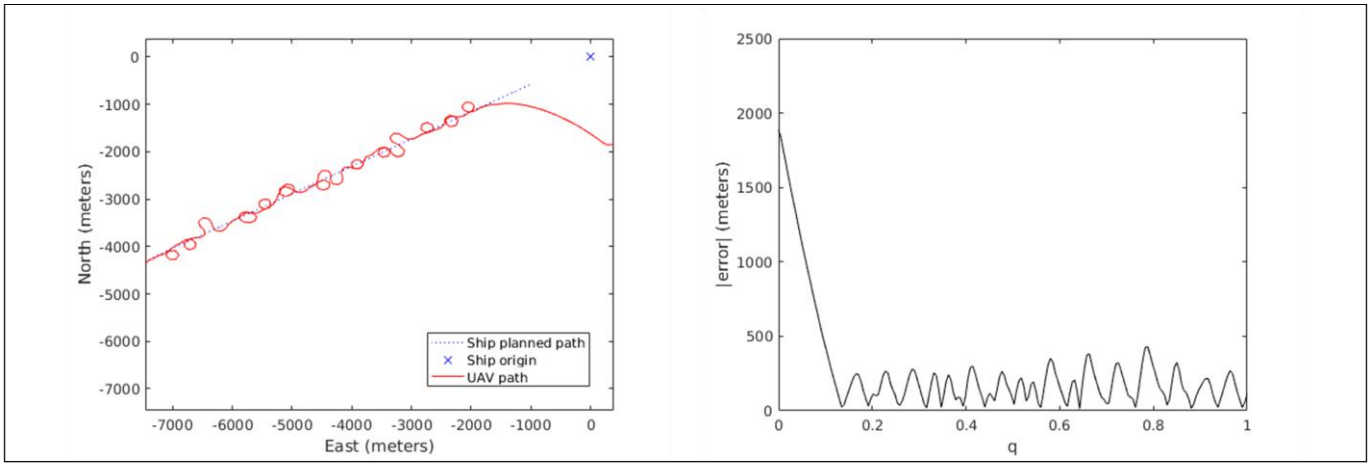

Fig 7. UAV path and error for constant ship velocity and heading without using the ship model in the MPC

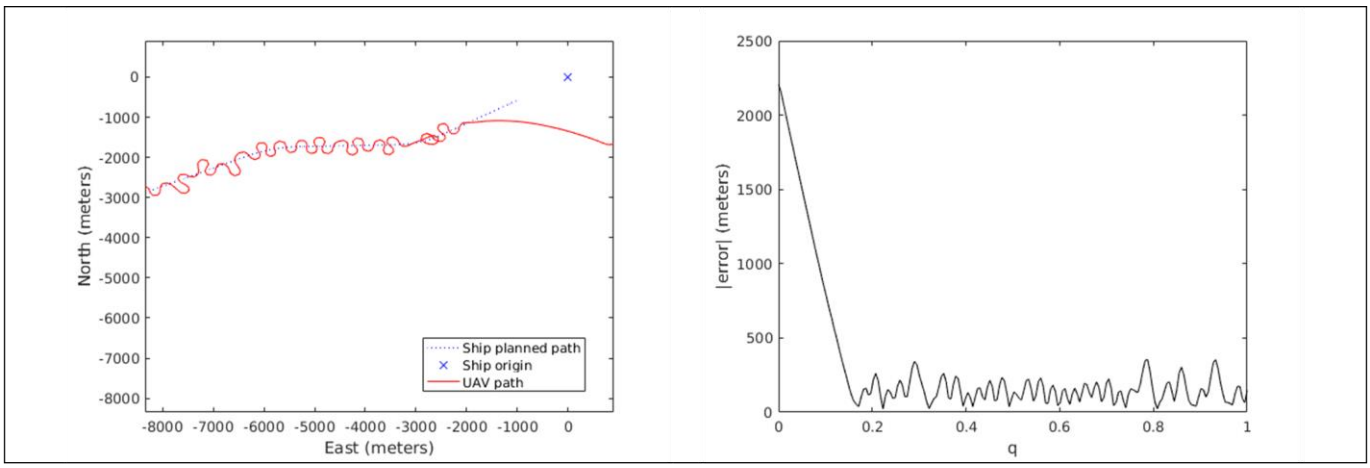

Fig. 8. UAV path and error for varying ship velocity and heading using the ship model in the MPC

Besides, the target waypoint is 6 seconds from the UAV's current position, what means that it is expected that the waypoint is updated about 36 meters before the UAV reaches the target one. That choice was made because if the system wanted to wait the UAV to reach the waypoint or if it allowed the UAV to get very close to the waypoint, the UAV would start to sorely turn trying to reach the waypoint or even it could start to loiter around the waypoint without ever reaching it. This kind of behavior is harmful for the optimization, which should control the UAV in a smoothly way.

For the MPC parameters, the choice of the maximum of 30 iterations, 10 input steps and 20 seconds of horizon were due to hardware limitations for a fast waypoints update. However, as an update period of 4 seconds was used due to the ArduPilot's uncommon behavior when the UAV is too close to the waypoint, this longer period of 4 seconds makes it possible to have better parameters if a fine tuning is done. However, as good results were achieved and the solution proved viable, it was found that the choice of the parameters was convenient and a fine tuning would not represent a significant improvement.

Although the maximum turn rate of the X8 with the standard tuning is around 0.4 rad/s, if a value close to 0.4 is chosen, the UAV has an uncommon behavior because the MPC asks for too strong maneuvers. That uncommon behavior is due to the delay of the mission acknowledgement by the ArduPilot and also due to the inertia, for instance when the UAV is turning to the right and the control suddenly wants it to turn to the left. Therefore, a turn rate of 0.3 rad/s was chosen, but it can be fine-tuned with the other parameters to extract the best performance of the UAV.

VI. Results

Figure 6 shows the path and the distance error of a UAV following the simulated ship's path with constant velocity and heading. The ship dynamics' model was considered in this simulation, using a constant velocity ship trajectory prediction. In the graph on the left, the ship's path is shown in a dotted line while the UAV's path is shown in a solid line. The "x" marker represents the origin of the ship. The right graph shows the error according to its variation along the path (q from 0 to 1).

It is possible to notice that the UAV tries to overfly the path but as its velocity is higher than the ship's, the UAV has to maneuver, crossing the path many times doing turns. The mean error was 217 meters in this case.

When not using the ship's model in the MPC (Figure 7), the UAV does much more 360º turns because the optimization problem does not consider the motion of the ship, so it does not know that the ship is moving forward. The error in this case was 265 meters, compared to 217 of the previous case when the ship's model was considered, i.e., an improvement of 18% was achieved when considering the ship's model.

Even if the ship's path is not just a straight and if its velocity changes along the path, the UAV had a good performance and the error was kept low, as shown in Figure 8.

VII. Discussion

As this application was developed to run in an onboard assembly, the Software In Loop simulation is an viable way to test the integration of the different systems. For instance, there is no need to worry about communication delays between the systems as it will be running onboard the UAV. In the other hand, hardware processing performance has to be taken into account and that would be possible with a Hardware In Loop simulation. However, as the application was running in MATLAB installed in a laptop with Ubuntu with its graphic interface and also many processes that are useless for the application, for sure it will run smoothly in a dedicated board running a compiled code in python or C++. In this manner, it would be possible to have more MPC steps and also more iterations to achieve better results.

Besides, it was noticed that just sending waypoints to the autopilot is not the best way to guide it. It would probably be better if the guidance was done by changing the UAV's attitude and velocity. Even if that solution would demand a more complex model to be used in the MPC, the improvement might worth it.

About the mission itself, during the development of this solution, it was glimpsed that it would be more useful if the UAV flew over an area around the ship's planned path instead of overflying only the planned path as a line. Thus, it would be possible to detect objects that could be moving into the ship track [3]. Therefore, this will be implemented in future works.

VIII. Conclusion

In a cooperative mission with a ship and a UAV, a common use for the UAV is to use it to overfly an area of interest. In this problem, that area is around the ship's planned path. Therefore, it has been defined that the UAV should overfly the position that the ship may be in 2 minutes from the current time. The UAV should control its turning rate to keep the distance from that position as minimum as possible. Then, according to the optimized turn rate, waypoints are generated to feed the autopilot.

The path planning system based on Model Predictive Control has been shown very effective for the proposed problem. The UAV could keep its position very close to the ship's path most of the time, sometimes moving away from it with strategic turns to get back to an optimum position. For the second dataset, where the ship changes its velocity and heading over time, the system also achieved a good result showing that the MPC can adapt to any behavior of a ship's path.

The difference between the results with and without using the ship's model in the MPC also proved that it is fundamental to use the ship's model to get a better performance.

Besides, more simulations can be done for fine tuning, as changing the number of steps per second as well as using hardware in loop simulation aiming a flight test.

Acknowledgment

This work has been supported by the MarineUAS project, funded by the European Comission under the H2020

Programme (MSCA-ITN-2014-642153). We also acknowledge the Research Council of Norway, grant number 223254 - Centre for Autonomous Marine Operations and Systems (NTNU-AMOS).

REFERENCES

[1] M. Koscielski, R. Miler and M. Zieliński, "Maritime Situational Awareness (MSA)". Zeszyty Naukowe Akademia Marynarki Wokennej, vol. 4, Gdynia, 2007.

[2] F. Leira, T. A. Johansen and T. I. Fossen, "Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera", IEEE Aerospace Conference, Big Sky, 2015.

[3] T. A. Johansen, T. Perez, "Unmanned Aerial Surveillance System for Hazard Collision Avoidance in Autonomous Shipping", International Conference on Unmanned Aircraft Systems, 2016.

[4] F. Andrade, "Planejamento de Trajetória de Veículos Aéros não Tripulados para Consciência Situacional Marítima", Revista de Villegagnon, ISSN 1981-3589, 2015.

[5] J. Haugen, "Autonomous Aerial Ice Observation", Ph.D. thesis, NTNU, Trondheim, 2014.

[6] F. R. Ramirez, D. S. Benitez, E. B. Portas and J. A. L. Orozco, "Coordinated sea rescue system based on unmanned air vehicles and surface vessels ", IEEE OCEANS, 2011.

[7] A. Weintrit, "Marine Navigation and Safety of Sea Transportation: Navigational Problems", CRC Press/Balkema, 2013.

[8] Y. Kang and J. Hedrick, "Linear tracking for a fixed-wing uav using nonlinear model predictive control," IEEE Transactions on Control Systems Technology, vol. 17, no. 5, pp. 1202–1210, 2009.

[9] J. Haugen and L. Imsland. Monitoring moving objects using aerial mobile sensors. IEEE Transactions on Control Systems Technology, 24(2):475– 486, 2016

[10] E. Skjon, S. A. Nundal, F. S. Leira and T. A. Johansen, "Autonomous search and tracking of objects using model predictive control of unmanned aerial vehicle and gimbal: Hardware-in-the-loop simulation of payload and avionics", International Conference on Unmanned Aircraft Systems, Denver, 2015.

[11] B. Houska, H. J. Ferreau, M. Diehl, "ACADO Toolkit - An Open Source Framework for Automatic Control and Dynamic Optimization", Optim. Control Appl. a Methods, 2011.

[12] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, J. Sousa, "The LSTS toolchain for networked vehicle systems", IEEE OCEANS, 2013

[13] R. Stengel, "Flight Dynamics", Princeton University Press, 2004.

[14] K. Gryte, "High Angle of Attack Landing of an Unmanned Aerial Vehicle", Master thesis, NTNU, Trondheim, 2015.

[15] E. F. Camacho, "Model Predictive Control", Springer-Verlag, 2004.