Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 53-2 (2020) 14628–14635

A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries in Confined Waters

Emil H. Thyri [*] **Morten Breivik** [*] **Anastasios M. Lekkas** [*]

[*] *Centre for Autonomous Marine Operations and Systems (AMOS), Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU). NO-7491 Trondheim, Norway.*

Abstract: A deliberate collision avoidance approach for autonomous surface vehicles operating in confined waters with high traffic is presented. The approach focuses on dynamic obstacles, by assuming a predefined set of paths that are collision-free with respect to static obstacles. Hence, the collision avoidance problem is reduced to a velocity planning problem, which is solved by first transforming all dynamic obstacles to a path-time space and subsequently constructing a conditioned visibility graph and traversing it with Dijkstra's algorithm. The performance of this approach is demonstrated through both simulations and full-scale experiments in Trondheim harbor, using the NTNU milliAmpere ferry platform and virtual dynamic obstacles.

Keywords: Autonomous surface vehicle, collision avoidance, path-velocity decomposition

1. INTRODUCTION

Zero-emission autonomous passenger ferries have received increasing attention in recent years. Such ferries can provide a much needed option for reducing the stress on existing urban infrastructure by utilizing the waterways. Moreover, their acquisition and operation are expected to cost less than pedestrian and bicycle bridges, while at the same time enabling bridge-free waters for leisure boats and sailboats. In addition, electric autonomous transport can reduce operator cost by automation, and air pollution by replacing fossil-fuel alternatives. The task repetitiveness and small area of operation restrict the complexity of autonomous navigation for urban waterways, and make it a fine case for bridging the interaction between autonomous systems and humans.

There are two major tasks that must be solved for autonomous systems to become a reality:

(1) Situational awareness, which is sensing and describing all aspects of the situation that are relevant for the execution of the mission. For autonomous surface vehicles (ASVs) this includes navigation and obstacle tracking.
(2) Automated planning in accordance with the understanding of the situation in a way that solves the mission in a satisfactory manner. For ASVs, this consists of planning trajectories from the current position to a desired destination which gives a predictable behaviour and avoids collision.

In the remainder of this paper, we will focus on the task of collision avoidance (COLAV) through deliberate trajectory planning. We assume that data on position, heading, velocity and extent of all relevant moving obstacles is available.

Previous work considering COLAV in the maritime domain includes development of reactive short-term and deliberate long-term methods, as well as combinations of such methods in hybrid structures. The hybrid structures can reap the benefit of both the long-term or global planning that often comes at a high computational cost, as well as the short-term responsiveness of reactive algorithms which typically have low computational cost. Figure 1 illustrates such a hybrid COLAV architecture with a long-term path and trajectory planner in combination with a reactive short-term COLAV algorithm. The work done by Bitar et al. (2019) describes a long-term trajectory planner that uses an optimization-based path planner to determine a path that is collision-free with static obstacles, in combination with an MPC approach to plan a local trajectory that is collision-free with regard to dynamic obstacles. In (Kuwata et al., 2014), a version of the reactive velocity obstacle (VO) algorithm with COLREGs features is implemented and validated through full-scale experiments. The VO algorithm selects a velocity pair from a finite set of admissible velocities that ensure no collision, at the cost of potentially deviating from the initial trajectory. Another reactive algorithm is the branching-course model predictive control (BC-MPC) algorithm developed by Eriksen et al. (2019). The algorithm simulates a finite set of maneuvers for a short time-horizon and chooses the best combination of maneuvers according to some cost function. Both the VO and BC-MPC algorithms receive a path or trajectory which they track while adapting to dynamic obstacles by generating adjustments to the heading and velocity references. However, such adjustments can cause large deviations from the nominal path and might not be feasible for confined waters.

In this paper, we propose a trajectory planning method with COLAV functionality for the specific case of autonomous passenger ferries operating in confined waters

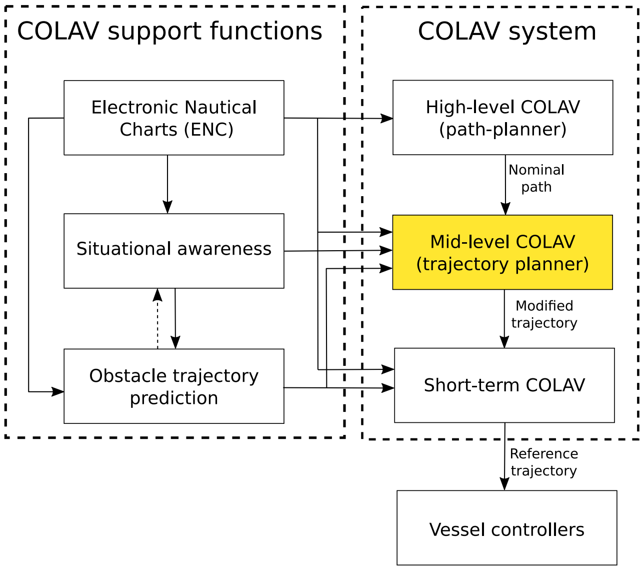

Fig. 1. A hybrid COLAV architecture as suggested in (Eriksen, 2019), with support functions such as ENC and situational awareness.

with high traffic. The approach is based on the principle of path-velocity decomposition, where the paths are predetermined and collision-free with regard to any static obstacles. The paths can be computed either offline or in a higher level in the COLAV architecture as in Fig. 1. The proposed method is suitable as a mid-level COLAV system in a hybrid system as the one depicted. In addition to the trajectory planning capabilities of the method, it is augmented with functionality for validation of the current trajectory according to updated obstacle data, where a failed validation triggers a replanning. This augmentation provides reactive features that make the method robust to changes in obstacle course and velocity.

The method of path-velocity decomposition was first suggested in (Kant and Zucker, 1986), where the moving obstacles are transformed onto the path-time space, based on the assumptions of constant obstacle behaviour, which in our case correspond to constant course over ground (COG) and speed over ground (SOG). A conditioned visibility-graph (Vgraph) is constructed in path-time space, and then traversed with Dijkstra's algorithm to find a collision-free velocity profile. This approach has been suggested for land-based robotics in (Fraichard and Laugier, 1993), and a similar approach is also applied to maritime COLAV with regard to static obstacles as a path-planner in (Casalino et al., 2009), but has to our best knowledge not been applied to COLAV for dynamic obstacles in the maritime domain.

We use a three degree-of-freedom (3-DOF) notation where $\boldsymbol{\eta}_v = [N_v, E_v, \psi_v]^\top$ is the ownship vessel pose in an Earth-fixed North-East-Down (NED) reference frame denoted $\{n\}$, and $\boldsymbol{\eta}_k = [N_k, E_k, \psi_k]^\top$ is the pose of obstacle k in $\{n\}$.

The remainder of this article is structured as follows: In Section 2, the velocity planning algorithm is described in detail along with its reactive features. Section 3 presents the simulation environment and results, while Section 4 presents the full-scale experimental results. Finally, Section 5 provides concluding remarks and future work.

2. COLLISION AVOIDANCE APPROACH

The suggested COLAV approach can be separated into two algorithms:

- **Algorithm 1** is the deliberate part of the COLAV system, which calculates a velocity profile for a path, and outputs a trajectory.
- **Algorithm 2** is the reactive part, which validates the current trajectory against most recent obstacle data. Algorithm 2 is used to trigger a replanning in Algorithm 1, see pseudocode below.

```
Algorithm 1 VELOCITY_PLANNER()
Input: Paths, Obstacle Tracking Data,Waypoints
atDestination = false
while atDestination == false do
    isValid = VALIDATE_CURRENT_WAYPOINTS();
    if isValid == false then
        calculate_obstacle_representations();
        transform_obstacles_to_path_time();
        construct_Vgraph();
        traverse_Vgraph();
        calculate_trajectory_from_waypoints();
    end
    atDestination = check_if_at_destination();
end
```

```
Algorithm 2 VALIDATE_CURRENT_WAYPOINTS()
Input: Waypoints, Obstacle Tracking Data
Output: boolean isValid
isValid = true;
if Current Waypoints is Empty then
    isValid = false;
    return isValid;
else
    calculate_obstacle_representations();
    for Every subpath in Waypoints do
        transform_obstacles_to_path_time();
        Intersection=test_for_intersection(subpath);
        if Intersection then
            isValid = false;
            return isValid;
        end
    end
    return isValid;
end
```

2.1 Predefined Paths

A set of five nominal paths is used. The paths are numbered 1 to 5, from port to starboard relative to the transit direction. Each nominal path is defined by n indexed waypoints, with $n-1$ subpaths connecting the waypoints. The subpaths are assumed to be straight lines between two consecutive waypoints. However, a path can be made up from an arbitrary number of waypoints to approximate a curved path. All nominal paths end up in the same point, the destination of the transit, but do not have the same starting point. Since the position of the ferry might not

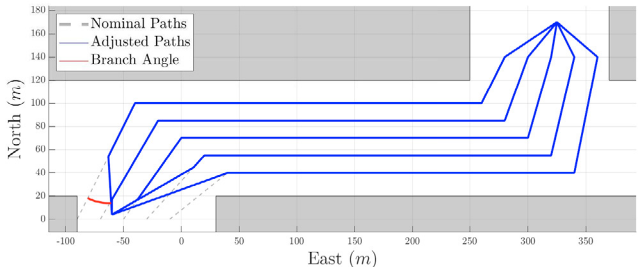

Fig. 2. Nominal paths and adjusted paths for a vessel at $[N_v, E_v] = [4, -60]$. The branch angle of 30° is illustrated in red for the first nominal path.

be on any of the nominal paths, a set of five adjusted paths are constructed. The first waypoint of all adjusted paths, wp_1, is the position of the ferry. For each of the five adjusted paths, the position of the second waypoint wp_2 is found by identifying the subpath i in the corresponding nominal path that is closest to the ferry, and subsequently calculating the position of wp_2 on the subpath such that the line from wp_1 to wp_2 makes a 30° angle to subpath i. We call this angle the branch angle. If the coordinates of wp_2 exceed the length of the subpath, wp_2 is set to the position of waypoint $i+1$. Lastly, the nominal waypoints from index $i+1$ to n are added to the adjusted paths. Fig. 2 shows the nominal paths through an east-west oriented canal, and the adjusted paths for a vessel located in $[N_v, E_v] = [4, -60]$. The branch angle is illustrated in red. In the choice of branch angle, factors such as passenger comfort, vessel maneuverability, transit length and COLREGs should be considered. As a compromise between readily apparent maneuvers and excessive maneuvering, we have used a branch angle of 30°.

2.2 Obstacle Representation

For robustness and ease of computation, we suggest a simplified obstacle representation that captures the relevant features of a dynamic obstacle. A diamond shaped region of collision (ROC) is placed around the obstacle in a way that fully encapsulates its estimated extension, in addition to the extension of ownship (OS), which is a designation used for the ferry. By doing this, the OS can be considered as a point without extension when constructing the Vgraph in the path-time space. A method for calculating the extent of the ROC is described by Lozano-Perez (1983). In addition to the ROC, we introduce two more regions: The high penalty region (HPR) and the low penalty region (LPR). A simplified obstacle representation is shown in Fig. 3. Each of the ROC, HPR and LPR for obstacle k are calculated according to

$$c_f = [N_k + l_f cos(\psi_k), E_k + l_f sin(\psi_k)], \quad (1)$$

$$c_a = [N_k + l_a cos(\psi_k + \pi), E_k + l_a sin(\psi_k + \pi)], \quad (2)$$

$$c_s = [N_k + l_s cos(\psi_k + \pi/2), E_k + l_s sin(\psi_k + \pi/2)], \quad (3)$$

$$c_p = [N_k + l_p cos(\psi_k - \pi/2), E_k + l_p sin(\psi_k - \pi/2)], \quad (4)$$

where c_f, c_a, c_s and c_p are the coordinates of the vertices in $\{n\}$ for the fore, aft, starboard and port direction respectively, and l_f, l_a, l_s and l_p denote the corresponding lengths of extension from the center of the obstacle. An appropriate length needs to be assigned to comply with the dimensions of the OS and the obstacle, including any desired safety factor and perimeter size. Here, a symmetrical extension has been used, with

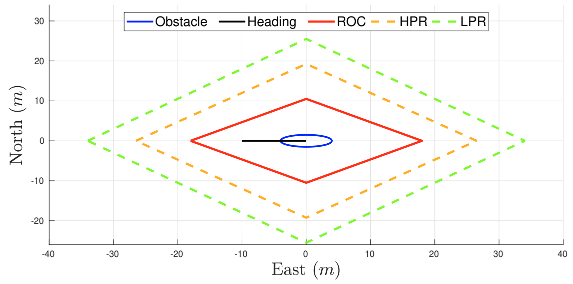

Fig. 3. Simplified obstacle representation for an 8 m long and 4 m wide obstacle traveling west.

$$l_f = l_a = (l_k + l_{os} + l_{fa\_region}) \quad (5)$$

$$l_s = l_p = (w_k + l_{os} + l_{sp\_region}) \quad (6)$$

where l_k and w_k are the length and width of obstacle k, and l_{os} is the length of OS. The variables l_{fa_region} and l_{sp_region} are the additional region sizes and are set to $l_{sp\_LPR} > l_{sp\_HPR} > l_{sp\_ROC} > 0$ and $l_{fa\_LPR} > l_{fa\_HPR} > l_{fa\_ROC} > 0$ for the ROC, HPR and LPR, respectively.

2.3 Transformation to Path-Time Space

To construct the Vgraph, the obstacle representation in $\{n\}$ is transformed to the path-time space. The transformation gives surfaces in path-time space that correspond to the obstacle representations occupation of the path in time. Since the obstacle representation consists of line segments moving in $\{n\}$, the problem consists of finding the intersection between each line segment with each subpath as a function of time. The assumption of constant obstacle behaviour ensures that the intersection is a line segment, and can be defined by its end-points $I_1 = [p_1, t_1]$ and $I_2 = [p_2, t_2]$ in path-time space, where the intersection line can be parameterized as

$$I = \frac{p - p_1}{p_2 - p_1} = \frac{t - t_1}{t_2 - t_1}, \quad I \in [0, 1], \quad (7)$$

where $[p, t]$ is a point on the line I. A general method for finding the intersection in $\mathbb{R}^3$ is given in (Kant and Zucker, 1986). We use the following method for finding I_1 and I_2 in $\mathbb{R}^2$:

The subpath P_j is subpath no. j in an adjusted path. In particular, P_j starts in $wp_s = [N_s, E_s]$, ends in $wp_e = [N_e, E_e]$ and has length $d_j = \sqrt{(N_e - N_s)^2 + (E_e - E_s)^2}$. The subpath is parameterized by

$$P_j = \frac{N - N_s}{a} d_j + d_{j-1} = \frac{E - E_s}{b} d_j + d_{j-1}, \quad (8)$$

where the point $[N, E]$ is located on the subpath for $P_j \in [d_{j-1}, d_j]$, with $a = N_e - N_s$ and $b = E_e - E_s$ where

$$d_{j-1} = \sum_{m=1}^{j-1} d_m \quad (9)$$

is the accumulated length for all prior subpaths. This gives the path the intuitive unit [m]. Further, assume an obstacle representation line segment L from $l_1 = [N_1, E_1]$ to $l_2 = [N_2, E_2]$ at time $t = t_0$, moving with a linear velocity $v = [v_N, v_E]$. The line is parameterized as

$$L = \frac{N - N_1 - v_N(t - t_0)}{c} = \frac{E - E_1 - v_E(t - t_0)}{d}, \quad (10)$$

where the point $[N, E]$ is located on the line segment for $L \in [0, 1]$ with $c = N_2 - N_1$ and $d = E_2 - E_1$, giving the four equations

$$\frac{P_j - d_{j-1}}{d_j} a + N_s = N, \tag{11}$$

$$\frac{P_j - d_{j-1}}{d_j} b + E_s = E, \tag{12}$$

$$Lc + N_1 + v_N(t - t_0) = N, \tag{13}$$

$$Ld + E_1 + v_E(t - t_0) = E. \tag{14}$$

By combining (11) and (13), and (12) and (14), N and E is eliminated, and the set of equations is reduced to

$$Lc + N_1 + v_N(t - t_0) = \frac{P_j - d_{j-1}}{d_j} a + N_s, \tag{15}$$

$$Ld + E_1 + v_E(t - t_0) = \frac{P_j - d_{j-1}}{d_j} b + E_s. \tag{16}$$

Equations (15)-(16) contain three unknown variables, $P_j \in [d_{j-1}, d_j]$, $L \in [0, 1]$ and $t \in (-\infty, \infty)$.

There are three possibilities for the intersection: I) the full length of L passes through P_j, II) L does not pass through P_j, or III) part of L passes through P_j. Start by assuming the first case, and solve (15)-(16) for $L = 0$ and $L = 1$ to get I_1 and I_2 respectively. If this gives $p_1, p_2 \in [d_{j-1}, d_j]$ this corresponds to I. If either $p_1, p_2 < d_{j-1}$ or $p_1, p_2 > d_j$, this is II and $I_1, I_2 = \emptyset$. Any other combination, where p_1 and p_2 is either on each side of the set $[d_{j-1}, d_j]$, or one is within the set and one is outside the set, corresponds to III. In this case, for each $p_i \notin [d_{j-1}, d_j], i \in [1, 2]$, set $P_j = d_{j-1}$ or $P_j = d_j$, corresponding to the boundary p_i is violating, and solve (15)-(16) to find t_i. The path-time coordinates of the ends of the intersection are $I_i = [p_i, t_i]$.

The I_1 and I_2 coordinates are calculated for all line segments in all obstacle representations, for each subpath to construct one path-time space representation for each of the five adjusted paths. Figure 4(a) shows the transformed obstacle representation of four moving obstacles onto a 104 m long straight-line path. In the transformation, three sets are constructed for each adjusted path; $\boldsymbol{N}$ is the node set, containing all intersection-points I_1 and I_2 from the transformation of all HPR and LPR, also including the node $I_{OS} = [0, t]$ at the start of the path, where t is the current time. $\boldsymbol{R}$ and $\boldsymbol{H}$ are the set of intersection line segments I from the transformation of all ROC and HPR, respectively.

2.4 Constructing the Vgraph

A traditional Vgraph is constructed from all edges connecting two nodes that do not intersect any edges in $\boldsymbol{R}$. Since the Vgraph is in the path-time space, it requires a few additional conditions, the second and third condition were formulated in (Kant and Zucker, 1986). First, time along an edge must be monotonically increasing. Furthermore, the velocity of an edge must be limited to the maximum velocity of the ferry, V_{max}. In addition, since the OS is located at $I_{OS} = [0, t]$, edges that are not reachable from the current coordinates are also omitted. For the Vgraph to span from I_{OS} to the end of the path, a set of end-nodes $\bar{\boldsymbol{N}}$ are calculated, one for each node in $\boldsymbol{N}$. For all end-nodes $\bar{n}_s$, $p_s = d_{n-1}$ according to (9), and

$$t_s = t_k + \frac{p_s - p_k}{V_{des}} \tag{17}$$

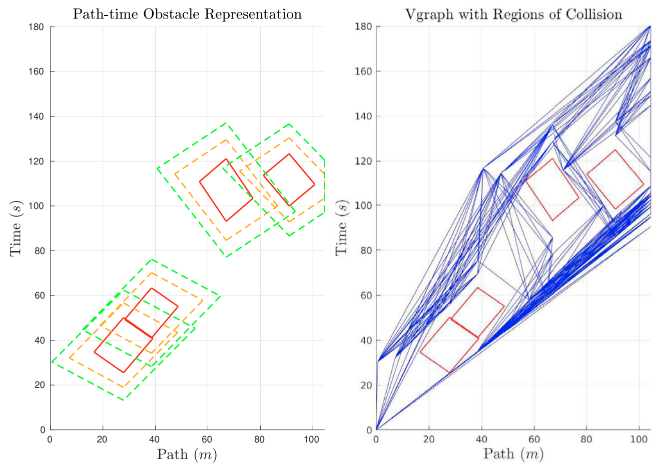

Fig. 4. Path-time space. (a) Four obstacle representations transformed to path-time space. (b) Constructed Vgraph with ROC from four obstacles.

where p_k and t_k are the path and time coordinates of the corresponding node in $\boldsymbol{N}$, and V_{des} is the desired transit velocity. The edge set $\boldsymbol{E}$, for the Vgraph, is determined by including all combinations of node pairs (n_i, n_j) from $\boldsymbol{N} \cup \bar{\boldsymbol{N}}$ that hold the conditions

C_1: $\overline{n_i, n_j} \cap \boldsymbol{R} = \emptyset$,
C_2: $|\frac{p_j - p_i}{t_j - t_i}| \leq V_{max}$,
C_3: $t_i < t_j$,
C_4: $|\frac{p_i}{t_i - t}| \leq V_{max}$ and
C_5: $t_i > t$.

For each edge in $\boldsymbol{E}$, a cost is calculated as

$$c_{ij} = c_t + c_v + c_l + c_n + c_{HPR}, \tag{18}$$

where c_t, c_v and c_l is the cost on time, velocity and length,

$$c_t = (t_j - t_i)k_t, \tag{19}$$

$$c_v = \left| \frac{(p_j - p_i)}{(t_j - t_i)} - V_{des} \right| k_v, \tag{20}$$

$$c_l = |p_j - p_i| k_l, \tag{21}$$

where $k_t > 0$, $k_v > 0$ and $k_l > 0$ are tuning parameters, c_n is the cost related to the node n_j and is either $k_{HPR} > 0$ or $k_{LPR} > 0$ depending on whether the node originates from the HPR or LPR. Lastly, c_{HPR} is set to k_{HPR} if $\overline{n_i, n_j} \cap \boldsymbol{H} \neq \emptyset$ and 0 if $\overline{n_i, n_j} \cap \boldsymbol{H} = \emptyset$ to penalize edges that pass through high penalty regions.

Figure 4(b) shows a Vgraph in path-time space with the four obstacles from Fig. 4(a). Joseph Kirk's Matlab implementation of Dijkstra's algorithm is used to traverse the graph (Kirk, 2015).

2.5 Trajectory from Waypoints

Dijkstra's algorithm outputs a set of path-time waypoints that make up the minimum-cost path. Furthermore, (11)-(12) are used to find the corresponding NED waypoints of the path-time waypoints. From the NED waypoints, along with time-coordinates of the path-time waypoints, we calculate a time-parameterized reference trajectory by using a third-order reference model (Fossen, 2011).

Fig. 5. The milliAmpere moving in Trondheim harbour.

2.6 Validation and Replanning

Algorithm 2 introduces the reactive features to the COLAV approach by periodically validating the current trajectory. The validation is performed by first constructing the set $\boldsymbol{R}$ like in Subsection 2.3, by transforming the ROC of all obstacles onto the path spanned by the current NED waypoints. Then, the algorithm constructs the set $\boldsymbol{S}$ of line segments from all subpaths connecting two consecutive waypoints in the current path-time waypoints, and lastly, checks for any intersections between the two sets. The current trajectory is valid if $\boldsymbol{S} \cap \boldsymbol{R} = \emptyset$, and not valid if $\boldsymbol{S} \cap \boldsymbol{R} \neq \emptyset$. If the test renders the current trajectory not valid, a replanning is performed in Algorithm 1, while the other case triggers no actions. In the simulations and experiments, the validation is performed with a time period of $T = 4$ s. The validation period should reflect the situational awareness system's ability to detect changes in obstacle speed and course. Another approach could be to trigger the validation when a change in course or heading for one or more of the obstacles is detected. In that case, only the ROCs of the obstacles with changed behaviour need to be transformed onto the path-time space.

3. SIMULATION RESULTS

Three simulations are implemented in Simulink with a 3-DOF model of the experimental ferry prototype milliAmpere shown in Fig. 5. A system overview of the simulator and guidance, navigation and control (GNC) system is given in Fig. 6. The vessel model and thruster model parameters are taken from the work of Pedersen (2019), where the vessel and thruster parameters used are listed in tables 3.4 and 3.9 respectively. The control system is the model reference adaptive controller described in (Sæther, 2019).

In the simulations, obstacle data without noise is available at all times. The data is provided by an obstacle simulator that can simulate different obstacle behaviours. In the presented results, three obstacle behaviours have been used; constant behaviour, "pass in front" and "slow down". The "pass in front" behaviour for obstacle k is active when it has a relative bearing to the ferry $b_{rel} \in (\pi/4, 3\pi/4)$ or $b_{rel} \in (-3\pi/4, -\pi/4)$, while having closing speed $v_{close} > 0$ and $l < l_{trig}$, where $l = \sqrt{(N_v - N_k)^2 + (E_v - E_k)^2}$ is the distance from the obstacle to the ferry and $l_{trig} > 0$ is a trigger distance. When the "pass in front" behaviour is triggered, the obstacle course is adjusted to aim for a point 20 m straight in front of the ferry. The "slow down" behaviour scales the obstacle velocity according to

$$\bar{v_k} = v_k f(l) \tag{22}$$

where $\bar{v_k}$ and v_k are the scaled and nominal velocity of obstacle i respectively, and

$$f(l) = \begin{cases} 1, & \text{if } l > r_1 \\ \frac{l-r_2}{r_1-r_2}, & \text{if } r_1 \geq l \geq r_2 \\ 0, & \text{otherwise,} \end{cases} \tag{23}$$

where $r_1 > r_2 > 0$ are the limits for the scaling. The parameter values for simulations and experiments are given in Table 1.

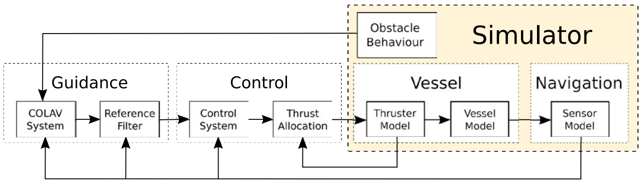

Fig. 6. Simulator and GNC system overview. The orange box contains the modeled vessel and sensor systems. The guidance and control systems are the same for both the simulations and experiments.

Table 1. Parameters

Name	Value	Unit	Name	Value	Unit
V_{des}	1.0	$\text{m} \cdot \text{s}^{-1}$	k_{HPR}	20	1
V_{max}	1.2	$\text{m} \cdot \text{s}^{-1}$	k_{LPR}	0	1
k_t	10	s^{-1}	l_{fa_ROC}	5.0	m
k_v	2	$\text{s} \cdot \text{m}^{-1}$	l_{sp_ROC}	2.5	m
k_l	1	m^{-1}	l_{fa_HPR}	12.5	m
l_{trig}	40	m	l_{sp_HPR}	11.2	m
r_1	40	m	l_{fa_LPR}	20.0	m
r_2	10	m	l_{sp_LPR}	17.5	m

The COLAV approach is tested in two environments:

- Environment 1 is a crossing over a narrow canal with traffic traveling close to perpendicular to the nominal paths.
- Environment 2 is a canal-crossing with a majority of the transit along the canal and hence parallel to the traffic.

3.1 Simulation 1

Simulation 1 is in Environment 1, with four obstacles traveling along the canal, two approaching from each side. The obstacles have constant behaviour. The ferry starts with initial condition $\boldsymbol{\eta_{v_0}} = [0, 0, 0.2915]^\mathsf{T}$. Fig. 7 gives situation overviews at different times of the transit. In the situation overviews, the ferry with course and track history is blue, and the obstacles with course and track history are red. The obstacle representations are included. The grey dashed lines give the nominal paths, and the solid grey areas indicate the canal-banks. The along path velocity and velocity reference are shown in Fig. 8, where the green dashed line denotes the time of velocity-planning.

From the figures, one can see that the ferry starts off with a close to zero velocity to let the two obstacles approaching from the port side pass, while moving from *path*3 to *path*4. Subsequently it accelerates to near transit velocity for the remainder of the transit. The change of path allows the

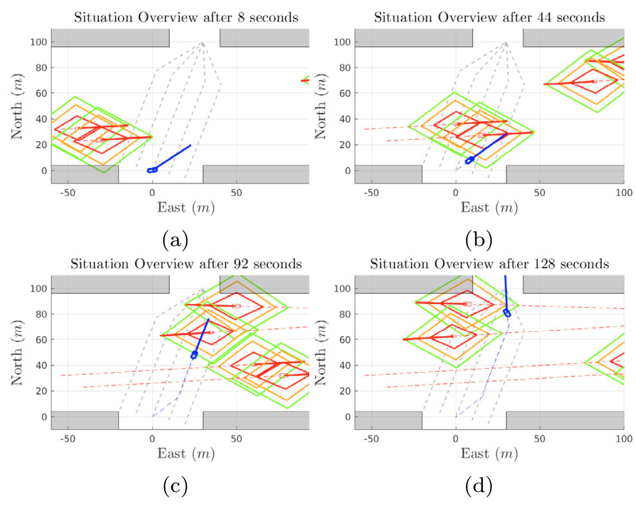

Fig. 7. Simulation 1: Situation overview of the transit. Obstacles have constant behaviour.

ferry to keep transit velocity and still pass behind the two obstacles approaching from starboard. From Fig. 8 one can see that only the initial planning was necessary. This is due to the obstacle behaviour complying with the assumptions made in the transformation to the path-time space.

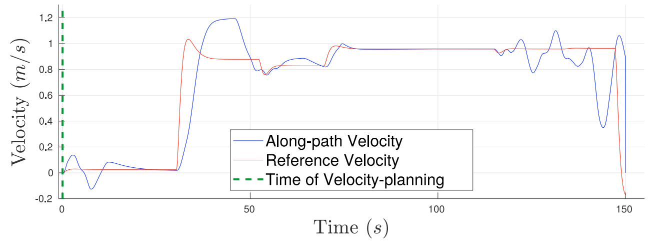

Fig. 8. Simulation 1: Along-path velocity and reference. Only the initial velocity-planning was necessary.

3.2 Simulation 2

Simulation 2 is also in Environment 1, with four dynamic obstacles traveling along the canal, two approaching from each side. The obstacles maneuver to pass in front of the ferry if they enter the enabling sector. The ferry has initial condition $\boldsymbol{\eta_{v_0}} = [0, 0, 0.2915]^\top$. Fig. 9 shows snapshots from the transit and Fig. 10 shows the along-path velocity and velocity reference.

From the velocity profile, we can see that the ferry starts off at transit velocity, and performs a replanning at $t = 8\,\text{s}$, which is triggered by one of the obstacles from the port side adjusting its heading as it enters the enabling sector. The replanning makes the ferry stop and wait for the obstacles to pass, as can be seen from Fig. 10. At $t = 36\,\text{s}$, the ferry proceeds at transit velocity. At $t = 84\,\text{s}$, a second replanning is triggered by the second obstacle approaching from the starboard side as it alters its course. This gives a re-planned trajectory that changes to *path*5 and follows this path at transit velocity to the destination. This simulation shows the reactive qualities of the COLAV system that are introduced by Algorithm 2.

3.3 Simulation 3

Simulation 3 is in Environment 2 with ten dynamic obstacles traveling along the canal, five approaching from each side. The obstacles have constant behaviour. The ferry starts with initial condition $\boldsymbol{\eta_{v_0}} = [0, -50, 0]^\top$. Fig. 11 shows snapshots from the transit, and Fig. 12 shows the along-path velocity and reference. This simulation demonstrates two interesting aspects of the COLAV method. One is the ability to handle high traffic, where the OS merges in-between five moving obstacles, and is unaffected by the five moving obstacles passing in the other direction within a short distance to the OS. An optimization-based algorithm like the one in (Hagen et al., 2018) might handle the same situation by altering course and/or heading to minimize some risk function. The other aspect is its ability to trail behind (or in front of) another obstacle at matching velocity in a lane-keeping manner, which is an intuitive and safe way of maneuvering in such confined waters.

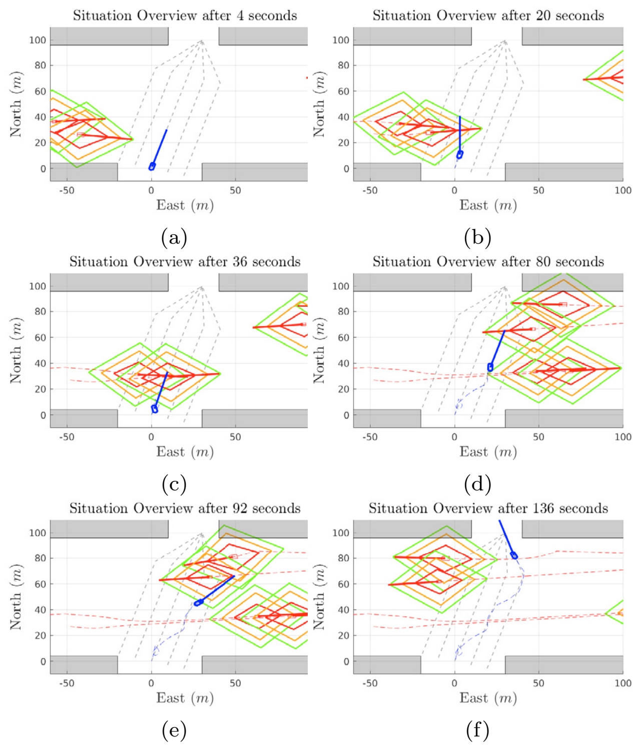

Fig. 9. Simulation 2: Situation overview of the transit. Obstacles have the ”pass in front” behaviour. The ferry changes path at each replanning.

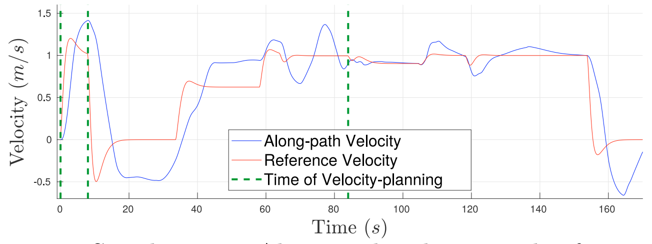

Fig. 10. Simulation 2: Along-path velocity and reference. Two replans are triggered in addition to the initial velocity-planning.

4. EXPERIMENTAL RESULTS

The experiment is conducted with milliAmpere depicted in Fig. 5. The experimental test platform is a 5 m by 2.8 m prototyping platform developed by NTNU that is used for developing and testing sensor systems, situational aware-

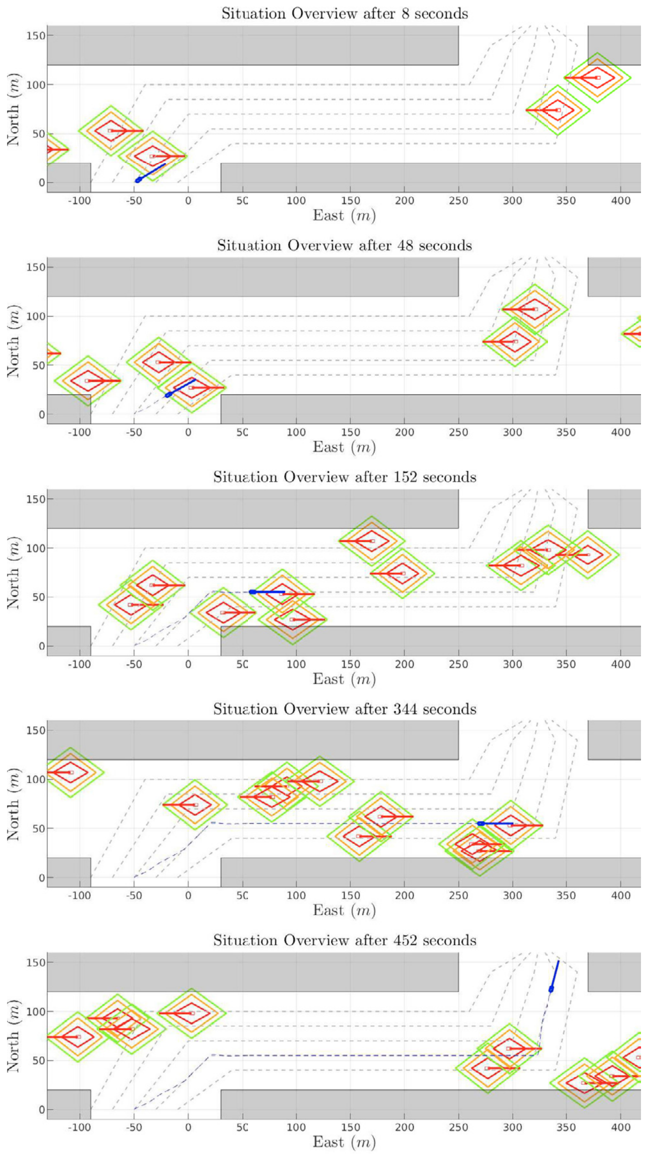

Fig. 11. Simulation 3: Situation overview of the transit. Obstacles have constant behaviour.

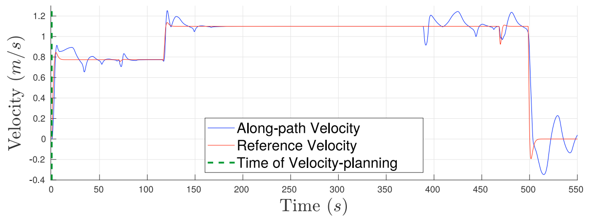

Fig. 12. Simulation 3: Along-path velocity and reference. Only the initial velocity-planning is necessary due to constant obstacle behaviour.

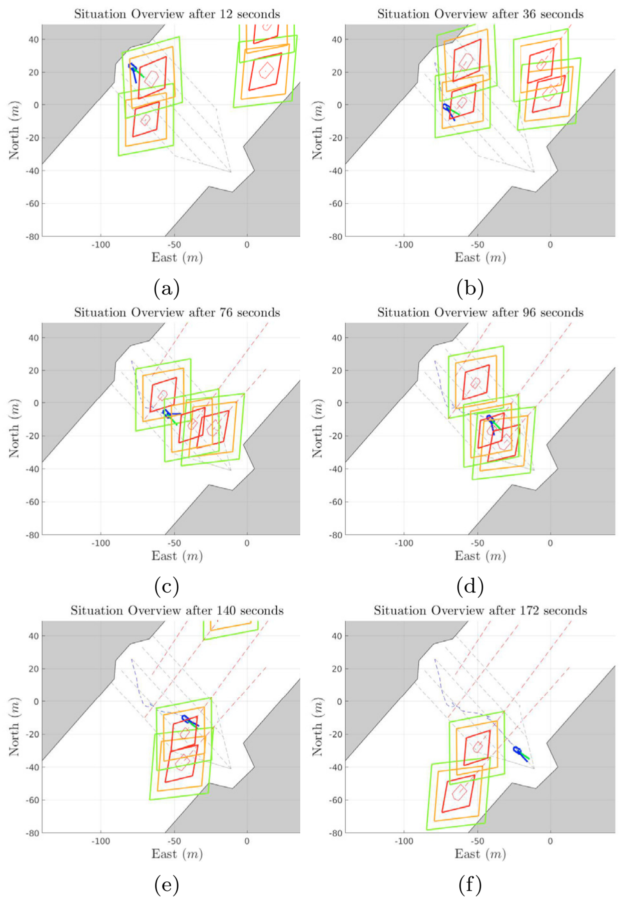

Fig. 13. Experiment: Situation overview of the transit. Obstacles act according to the "slow-down" behaviour.

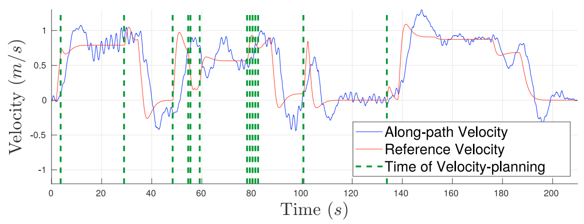

Fig. 14. Experiment: Along-path velocity and reference. Several replans are triggered due to the unpredictable obstacle behaviours.

ness algorithms, human machine interaction and COLAV systems. The experiment is conducted in a shielded part of Trondheim harbour with little traffic. The guidance and control system is the same as for the simulations. The code is generated from Simulink to run in ROS [1], and runs on an Axiomtek eBOX670-883-FL with an Intel Core I7 processor and Ubuntu OS. The sensor model is replaced by a navigation system consisting of a Vector™ VS330 GNSS Receiver with Real Time Kinematic (RTK) capacity, and an Xsens MTi-20 IMU, providing a position accuracy of 10 mm in the horizontal plane and 0.05° in heading. The experiment is run with virtual obstacles in the same obstacle simulator as for the simulations. The reference filter, control system, thrust allocation and obstacle simulator all run with a period of 0.1 s to ensure sufficient trajectory tracking. As mentioned, the COLAV algorithms run the validation with eventual replanning at a period of 4 s. In a scenario with 4 moving obstacles and 5 nominal paths, each made from 3 subpaths, Algorithm 2 and Algorithm 1 in sequence are able to finish calculations in about 0.2 s without any effort to optimize the code for runtime.

The presented experiment is chosen from the set of experiments in (Thyri, 2019). In the experiment the obstacles

[1] Robot operating system, `https://www.ros.org/about-ros/`

act according to the "slow-down" behaviour. Figure 13 gives a situation overview of the transit and Fig. 14 gives the planned and actual velocity profile for the transit. The ferry starts off along $path5$ at about 0.8 m/s to pass behind the two obstacles heading north-east. As the obstacles slow down, a replan is triggered that brings the ferry to a stop for about 15 s awaiting the obstacles to pass. Subsequently, the ferry moves back to $path3$ to pass behind the two obstacles traveling south-east. As the obstacles slow down, the current plan is rendered infeasible, and a series of replans are triggered around 80 s into the transit. The ferry is eventually halted for about 50 s to wait for the obstacles to proceed before the transit is completed without further replans. This scenario shows how the reactive features of the COLAV approach makes it able to adapt to deviations from the assumptions on constant obstacle behaviour that are made in the transformation to path-time space. Another remark is that the velocity planner causes the ferry to stop at two occasions which is because the 30° branch angle used in calculating the adjusted paths and the short distance to the closest obstacle makes all five adjusted paths pass through the ROC of that obstacle. A solution to this could be to have a dynamic construction of the adjusted paths, e.g. in a higher level of the COLAV architecture. Another solution could be to pair the velocity planner with a more reactive COLAV algorithm that would allow deviations from the adjusted paths. A third thing to note about the stop-and-go behaviour is that it puts demands on the vessel actuation system. For the algorithm to work in the presence of environmental disturbances from wind and current, the vessel needs to be fully actuated in order to maintain tracking for both position and heading when stationary. This is particularly important for passenger transport to ensure passenger comfort and safety. For underactuated ASVs, a solution can be to introduce a $V_{min} > 0$ condition on the edges when constructing the Vgraph.

5. CONCLUSION

A simple and robust deliberate COLAV approach for autonomous passenger ferries is presented in this paper. The method can run in real time and produces predictable and intuitive trajectories in confined waters with high-traffic. It can also adapt to unforeseen changes without deviating from the predefined transit paths, ensuring safe maneuvering in confined waters.

Future work includes improving COLREGs compliance by augmenting the obstacle representation, analyzing the completeness of the method, pairing the velocity planner with a reactive COLAV algorithm, improving the trajectory tracking and control system, and testing the COLAV approach together with real object tracking data from exteroceptive sensors to see how it handles noise and uncertainty.

ACKNOWLEDGEMENTS

This work was supported by the NTNU Digital transformation project Autoferry and the Research Council of Norway through the Centres of Excellence funding scheme with project number 223254.

REFERENCES

Bitar, G., Eriksen, B.O.H., Lekkas, A.M., and Breivik, M. (2019). Energy-optimized hybrid collision avoidance for ASVs. In *Proc. 18th European Control Conference (ECC)*, 2522–2529. Naples, Italy. doi:10.23919/ECC.2019.8795645.

Casalino, G., Turetta, A., and Simetti, E. (2009). A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields. In *Proc. OCEANS 2009-EUROPE*, 1–8. Bremen, Germany. doi:10.1109/OCEANSE.2009.5278104.

Eriksen, B.O.H., Breivik, M., Wilthil, E.F., Flåten, A.L., and Brekke, E.F. (2019). The branching-course model predictive control algorithm for maritime collision avoidance. *Journal of Field Robotics*. doi:10.1002/rob.21900.

Eriksen, B.O.H. (2019). *Collision Avoidance and Motion Control for Autonomous Surface Vehicles*. Ph.D. thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Fossen, T.I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons.

Fraichard, T. and Laugier, C. (1993). Path-velocity decomposition revisited and applied to dynamic trajectory planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, 40–45. Atlanta, USA.

Hagen, I.B., Kufoalor, D.K.M., Brekke, E.F., and Johansen, T.A. (2018). MPC-based collision avoidance strategy for existing marine vessel guidance systems. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 7618–7623. Brisbane, Australia. doi:10.1109/ICRA.2018.8463182.

Kant, K. and Zucker, S.W. (1986). Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5, 72–89.

Kirk, J. (2015). Dijkstra's minimum cost path algorithm. URL `https://se.mathworks.com/matlabcentral/fileexchange/20025-dijkstra-s-minimum-cost-path-algorithm`.

Kuwata, Y., Wolf, M.T., Zarzhitsky, D., and Huntsberger, T.L. (2014). Safe maritime autonomous navigation with COLREGs, using velocity obstacles. *IEEE Journal of Oceanic Engineering*, 39(1), 110–119. doi:10.1109/JOE.2013.2254214.

Lozano-Perez (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2), 108–120. doi:10.1109/TC.1983.1676196.

Pedersen, A.A. (2019). *Optimization Based System Identification for the milliAmpere Ferry*. Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Sæther, B. (2019). *Development and Testing of Navigation and Motion Control Systems for milliAmpere*. Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Thyri, E.H. (2019). *A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries*. Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.