Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-29 (2018) 1–7

Hybrid Collision Avoidance for Autonomous Surface Vehicles

Einvald Serigstad, Bjørn-Olav H. Eriksen, Morten Breivik

Centre for Autonomous Marine Operations and Systems
Department of Engineering Cybernetics
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
einvald@gmail.com
{bjorn-olav.h.eriksen, morten.breivik}@ieee.org

Abstract: Autonomous vehicles and collision avoidance (COLAV) systems are advancing rapidly. However, the majority of the COLAV methods developed are not designed for vehicles with second-order nonholonomic constraints, such as autonomous surface vehicles (ASVs). This paper proposes the hybrid dynamic window (HDW) algorithm, which in addition to acting as a reactive COLAV method, functions as a trajectory tracker in a hybrid COLAV architecture. The algorithm serves as an interface to any deliberate COLAV method which generates time parameterized trajectories, enabling vehicles to avoid local minima and track optimal trajectories towards the goal. Furthermore, a new distance function is developed for the dynamic window (DW) algorithm, improving the algorithm trajectory planning when operating close to obstacles. A case study is done for an ASV model using the HDW algorithm and the rapidly-exploring random tree (RRT) algorithm, which together form a hybrid COLAV system. The performance is evaluated through simulations using a defined set of performance metrics.

Keywords: Hybrid collision avoidance, trajectory planning, dynamic window algorithm, autonomous surface vehicles

1. INTRODUCTION

To employ autonomous surface vehicles (ASVs) in environments with obstacles and avoid collisions, the ASVs need a reliable collision avoidance (COLAV) system. Furthermore, in unknown dynamic environments the COLAV system is required to be running in real-time (Brock and Khatib, 1999). Reactive COLAV methods, also called local or sense-act methods, base their actions on currently available sensor data. These methods are memoryless, considering only the immediate environment, and have low computational requirements. Deliberate COLAV methods, also called global methods, find a route from a given start to a given goal based on available environment data. By planning ahead, a deliberate method with knowledge of the environment will increase the chance of reaching the goal without getting stuck in local minima. However, this comes at the cost of increased computational requirements. Hybrid algorithms combine the benefits of both reactive and deliberate algorithms by combining them, using the deliberate algorithm for long-term planning and the reactive algorithm for real-time COLAV.

Many reactive COLAV methods including velocity obstacles (Fiorini and Shiller, 1998) and the dynamic window (DW) algorithm (Fox et al., 1997), and deliberate methods including the probabilistic rapidly-exploring random trees (RRT) algorithm (LaValle, 1998), have been developed over the years. Most COLAV methods are designed for vehicles with only holonomic or first-order nonholonomic constraints, including the DW algorithm. However, few results are presented for vehicles with second-order nonholonomic constraints. A modification to the DW algorithm is presented in (Eriksen et al., 2016) which takes second-order nonholonomic constraints into account. The algorithm is shown to perform well for ASV COLAV (Serigstad, 2017), but introduces some weaknesses when operating close to obstacles. A safety region region around obstacles, which the vehicle is motivated to avoid is introduced in the algorithm. If the vehicle is inside the safety region, however, the algorithm will not be motivated to leave the region and will lose the ability to evaluate the vehicle distance towards obstacles. Furthermore, the region may cause problems when travelling through narrow passages. This paper resolves the weaknesses introduced with the safety region. An ASV used for full scale testing of the DW algorithm in (Eriksen et al., 2018) is depicted in Figure 1.

In this paper, we introduce the hybrid dynamic window (HDW) algorithm, which in addition to functioning as a reactive COLAV method, tracks a global trajectory generated by a deliberate algorithm in a hybrid system. Furthermore, a new distance function for the DW algorithm is presented, which greatly reduces the weaknesses of operating close to or inside the safety region. The hybrid architecture is tested on an ASV through simulations, and evaluated using performance metrics. Furthermore, a branching of the DW trajectory predictions is applied, enabling the algorithm to change desired velocity during the trajectory prediction. This article is based on the master thesis (Serigstad, 2017).

Fig. 1. The Telemetron is a dual-use surface vessel capable of operating as an ASV.

The rest of the paper is organized as follows: Section 2 presents the ASV model. A modified DW algorithm introduced in (Eriksen et al., 2016) is reviewed in Section 3. The HDW algorithm and a new distance function are introduced in Section 4. The HDW algorithm is evaluated through simulations in Section 5, while Section 6 concludes the paper.

2. ASV MODEL

To model the ASV, we use the widely used 3 DOF model (Fossen, 2011):

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \tag{2}$$

where $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times S$ is the ASV pose in the North-East-Down (NED) reference frame, and $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$ is the ASV velocity in the body-fixed reference frame. The matrix $\boldsymbol{R}(\psi)$ is a rotation matrix defining the rotation between the body frame and the NED frame:

$$\boldsymbol{R}(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

The inertia matrix $\boldsymbol{M}$ is given as:

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{23} & m_{33} \end{bmatrix}, \tag{4}$$

while the Coriolis and centripetal matrix $\boldsymbol{C}(\nu)$ is given as:

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m_{23}r - m_{22}v \\ 0 & 0 & m_{11}u \\ m_{23}r + m_{22}v & -m_{11}u & 0 \end{bmatrix}. \tag{5}$$

The damping matrix is given as $\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D}_L + \boldsymbol{D}_{NL}(\boldsymbol{\nu})$, where $\boldsymbol{D}_L$ is a linear damping matrix and $\boldsymbol{D}_{NL}(\boldsymbol{\nu})$ is a non-linear damping matrix:

$$\boldsymbol{D}_L = \begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix}, \tag{6}$$

$$\boldsymbol{D}_{NL}(\boldsymbol{\nu}) = \boldsymbol{I}_{3\times3} \begin{bmatrix} X_{|u|u}|u| + X_{uuu}u^2 \\ Y_{|v|v}|v| + Y_{vvv}v^2 \\ N_{|r|r}|r| + N_{rrr}r^2 \end{bmatrix}. \tag{7}$$

ASVs usually employ a propeller and rudder for control, giving a control input $\boldsymbol{\tau}$ which may be modeled as:

$$\boldsymbol{\tau} \triangleq \boldsymbol{B}\boldsymbol{f}, \tag{8}$$

where $\boldsymbol{f} = [X \;\; N]^\top$ is a vector of the propeller force X and rudder torque N, and the matrix $\boldsymbol{B}$ is given as:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{l_r} \\ 0 & 1 \end{bmatrix}, \tag{9}$$

where $l_r > 0$ is the length from the rudder to the center of origin. The yaw moment N is modelled as:

$$N = -K_{\delta_\psi} u^2 l_r \delta_\psi, \tag{10}$$

where u is the surge speed, $K_{\delta_\psi} > 0$ is a rudder coefficient and δ_ψ is the rudder angle based on the model used in (Eriksen et al., 2016). The ASV rudder angle and angle rate is constrained by:

$$\left|\delta_\psi\right| \leq \delta_{\max} \tag{11}$$

$$\left|\dot{\delta}_\psi\right| \leq \dot{\delta}_{\max}, \tag{12}$$

where $\delta_{max} > 0$ is the maximum rudder angle, and $\dot{\delta}_{max} > 0$ is the maximum rudder angle rate. The propeller force is defined as:

$$X \in [X_{\min}, X_{\max}], \tag{13}$$

where $X_{\min} < 0$ and $X_{\max} > 0$ are the propeller saturation.

3. THE MODIFIED DYNAMIC WINDOW ALGORITHM

A modification to the DW algorithm (Fox et al., 1997) taking second-order nonholonomic constraints into account is presented in (Eriksen et al., 2016), and is denoted the modified dynamic window (MDW) algorithm in this paper. The algorithm searches for an optimal velocity within the velocity space formed by the vehicle translational speed and rotational rate. For an ASV, a point in the velocity space consists of the surge speed u and the yaw rate r, which form a velocity pair (u, r). The optimal velocity pair is used as a reference velocity for the vehicle controllers. The algorithm generates a search space ensuring feasible velocities, which is discretized into a finite set of velocity pairs. For every discrete velocity pair within the search space, a vehicle trajectory is predicted using a closed-loop error model. Finally, an objective function defines the optimal velocity pair.

To generate the search space in the DW algorithm, a union of three sets are used. The dynamic window V_d contains velocities the vehicle can reach within a short timeframe. The set of admissible velocities V_a contains velocities allowing the vehicle to stop before colliding with the closest obstacle along the predicted trajectory. The set of possible velocities V_s contains feasible velocities considering actuator saturation limits.

Allowing a time T_a for changing the rudder setting, we obtain a set of possible rudder angles:

$$\delta_\psi \in sat\Big(\Big[\delta_\psi^* - T_a\dot{\delta}_{\max}, \delta_\psi^* + T_a\dot{\delta}_{\max}\Big], \delta_{\max}\Big), \tag{14}$$

where $sat(\cdot)$ is the saturation function, and δ_ψ^* is the current rudder angle, constrained by (11) and (12). The possible vessel acceleration is defined as:

$$\dot{\boldsymbol{\nu}}_i = \boldsymbol{M}^{-1}(\boldsymbol{\tau}_i - \boldsymbol{C}(\boldsymbol{\nu}^*)\boldsymbol{\nu}^* - \boldsymbol{D}(\boldsymbol{\nu}^*)\boldsymbol{\nu}^*), \tag{15}$$

where $i \in \{\min, \max\}$, and

$$\boldsymbol{\tau}_{\min} \triangleq \boldsymbol{\tau}(\boldsymbol{\nu}^*, \max(\delta_\psi), \min(X)) \tag{16}$$

$$\boldsymbol{\tau}_{\max} \triangleq \boldsymbol{\tau}(\boldsymbol{\nu}^*, \min(\delta_\psi), \max(X)), \tag{17}$$

where the actuation constraints are given by (13) and (14). Note that a positive rudder moment causes a negative yaw rate.

Allowing a time $T_s > T_a$ for accelerating the vessel, the dynamic window is defined as

$$\begin{aligned} V_d = \big\{&(u,r) \in \mathbb{R} \times \mathbb{R} \\ &\mid u \in [u^* + \dot{u}_{\min} \cdot T_s, u^* + \dot{u}_{\max} \cdot T_s] \\ &\wedge r \in [r^* + \dot{r}_{\min} \cdot T_s, r^* + \dot{r}_{\max} \cdot T_s]\big\} \end{aligned}. \tag{18}$$

The set of possible velocities V_s is found from:

$$V_s = \Big\{(u,r) \in \mathbb{R} \times \mathbb{R} \mid g(u,r) \geq 0\Big\}, \tag{19}$$

where $g(u,r)$ is positive for feasible steady-state velocities and negative otherwise. The function $g(u,r)$ is computed by finding the boundaries of the steady state solution of the vehicle dynamics, given the actuator magnitude constraints (11) and (13).

To penalize velocity pairs leading towards obstacles, two regions are defined as the safety region

$$\Omega \triangleq \left\{\boldsymbol{p} \in \bar{\boldsymbol{\mathcal{C}}} \middle| \left\|\boldsymbol{p} - \boldsymbol{p}_{forb}\right\|_2 \leq r_\Omega \right\}, \tag{20}$$

and the collision region

$$\mathcal{T} \triangleq \left\{\boldsymbol{p} \in \bar{\boldsymbol{\mathcal{C}}} \middle| \left\|\boldsymbol{p} - \boldsymbol{p}_{forb}\right\|_2 \leq r_{\mathcal{T}} \right\}, \tag{21}$$

where $\boldsymbol{p}_{forb} \in \mathbb{R}^2$ is the position of obstacles, $\boldsymbol{p} = [x, y]^\top$ is the configuration in the configuration space $\bar{\boldsymbol{\mathcal{C}}}$, $r_{\mathcal{T}} \geq 0$ defines the size of $\mathcal{T}$, and $r_\Omega \geq r_{\mathcal{T}}$ is a scalar defining the size of the safety region Ω. The ASV configuration space $\bar{\boldsymbol{\mathcal{C}}} \in \mathbb{R}^2$ is defined by reducing $\boldsymbol{\mathcal{C}} \in \mathbb{R}^2 \times SO(2)$ by approximating the vehicle footprint as a circle. The size of the collision region r_T is then selected to be equal or larger than the vehicle radius. The regions are named the avoidance and antitarget region in (Eriksen et al., 2016; Serigstad, 2017), but are renamed in this paper to more intuitive terms. A set of admissible velocities is defined as:

$$V_a = \left\{ \begin{aligned} &(u,r) \in \mathbb{R} \times \mathbb{R} \big| u \leq \sqrt{2\rho'(u,r)|\dot{u}_{\min}|} \\ &\wedge |r| \leq \left\{ \begin{matrix} \sqrt{2\rho'(u,r)|\dot{r}_{\max}|}, & r < 0 \\ \sqrt{2\rho'(u,r)|\dot{r}_{\min}|}, & r \geq 0 \end{matrix} \right\} \end{aligned} \right\}, \tag{22}$$

where

$$\rho'(u,r) = \max(\rho(u,r) - \Delta_s, 0), \tag{23}$$

is the distance to the collision region at next algorithm iteration, Δ_s is the distance travelled until the next iteration, and $\rho(u,r)$ is the distance to the safety region.

Lastly, the search space V_r is defined by the union of the three sets

$$V_r = V_s \cap V_a \cap V_d. \tag{24}$$

To take second-order nonholonomic constraints into account, a trajectory prediction using a closed-loop error model is defined in (Eriksen et al., 2016). By solving (2) for $\dot{\boldsymbol{\nu}}$, the system can be described as:

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f} - \boldsymbol{n}(\boldsymbol{\nu}), \tag{25}$$

where

$$\boldsymbol{n}(\boldsymbol{\nu}) = \boldsymbol{M}^{-1}(\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}). \tag{26}$$

It can be shown that by using the control law

$$\boldsymbol{f} = (\boldsymbol{\Gamma}_1\boldsymbol{M}^{-1}\boldsymbol{B})^{-1}(\boldsymbol{\Gamma}_1\boldsymbol{n}(\boldsymbol{\nu}) + \boldsymbol{a}_{1_d}), \tag{27}$$

where $\boldsymbol{a}_{1_d}$ is the desired acceleration, the closed-loop vessel velocity is given as:

$$\tilde{\boldsymbol{\nu}}(t) = e^{\boldsymbol{A}t}\tilde{\boldsymbol{\nu}}(0) - \boldsymbol{A}^{-1}(\boldsymbol{I} - e^{\boldsymbol{A}t})(\boldsymbol{\beta}\boldsymbol{\nu}_{1_d} + \boldsymbol{G}). \tag{28}$$

The terms of (28) are defined as:

$$\begin{aligned} \boldsymbol{A} &= -(\boldsymbol{\Gamma}_1^\top \boldsymbol{K}_p \boldsymbol{\Gamma}_1 + \boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{N}) \\ \boldsymbol{\beta} &= -\boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{N} \boldsymbol{\Gamma}_1^\top \\ \boldsymbol{G} &= -\boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{b}(\boldsymbol{\nu}*), \end{aligned} \tag{29}$$

where

$$\boldsymbol{\Gamma}_1 \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{\Gamma}_2 \triangleq \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \tag{30}$$

and $\boldsymbol{K}_p = \operatorname{diag}(k_{p_u}, k_{p_r}) > 0$, where k_{p_u} and k_{p_r} are parameters for a proportional controller. The Jacobian matrix $\boldsymbol{N}$ is defined as:

$$\boldsymbol{N} = \left.\frac{\partial \boldsymbol{n}(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}}\right|_{\boldsymbol{\nu}=\boldsymbol{\nu}*}. \tag{31}$$

Using (28) as input, the modified Euler method is used for predicting the vessel trajectory:

$$\begin{aligned} \boldsymbol{\eta}(t_{n+1}) &= \boldsymbol{\eta}(t_n) + h\boldsymbol{k}_2 \\ \boldsymbol{k}_1 &= \boldsymbol{R}(\boldsymbol{\eta}(t_n))\boldsymbol{\nu}(t_n) \\ \boldsymbol{k}_2 &= \boldsymbol{R}(\boldsymbol{\eta}(t_n) + \frac{h}{2}\boldsymbol{k}_1)\boldsymbol{\nu}(t_n + \frac{h}{2}), \end{aligned} \tag{32}$$

where h is the time step of the integration.

The optimal velocity is defined by the optimization problem:

$$\begin{aligned} \max_{(u,r)} \ G(u,r) =& \alpha \cdot yawrate(r, r'_d) + \beta \cdot dist(u,r) \\ &+ \gamma \cdot velocity(u, u'_d) \\ s.t. \ (u,r) \in V_r & \end{aligned}, \tag{33}$$

where the functions are weighted by the variables $\alpha, \beta, \gamma > 0$, and the terms are given as:

$$yawrate(r, r'_d) = 1 - \frac{|r'_d - r|}{\max\limits_{r \in V_r}(|r'_d - r|)} \in [0,1], \tag{34}$$

$$velocity(u, u'_d) = 1 - \frac{|u'_d - u|}{\max\limits_{u \in V_r}(|u'_d - u|)} \in [0,1], \tag{35}$$

$$dist(u,r) = \frac{\bar{\rho}(u,r)}{\int_0^T \left\|\boldsymbol{\chi}(u,r,t)\right\|_2 dt} \in [0,1]. \tag{36}$$

The yawrate and velocity terms favor a velocity pair close to an input (u'_d, r'_d), which is used to guide the DW algorithm. The distance function $dist(u,r)$ is the approximated time until a collision occurs along the given trajectory, where $\bar{\rho}(u,r)$ is the distance to Ω, and $\boldsymbol{\chi}(u,r,t)$ is the predicted vehicle surge and sway speed along the predicted trajectory given the velocity pair (u,r). A line of sight (LOS) tracking method (Fossen, 2011) is used to find a desired vehicle heading ψ_d. Using this, we define the desired yaw rate using the yaw control law:

$$r'_d = -k_\psi(\psi - \psi_d) + \dot{\psi}_d, \tag{37}$$

where $k_\psi > 0$ is a constant gain. The desired surge speed u'_d is set to a constant value. The introduction of r'_d and u'_d enables external guidance of the desired surge speed and yaw rate. The MDW algorithm takes the desired velocity pair (r'_d, u'_d) as input.

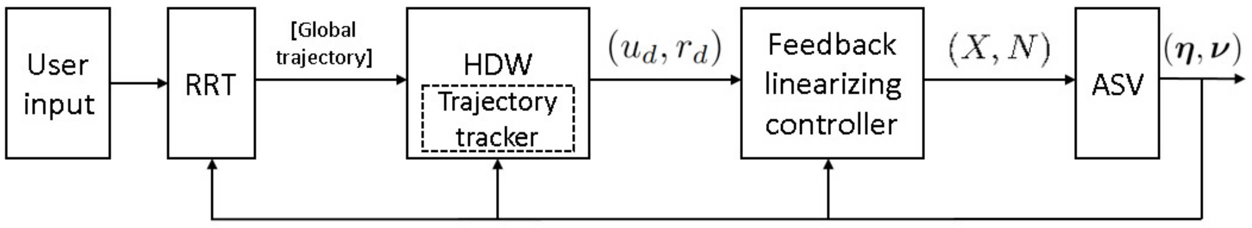

Fig. 2. HDW algorithm architecture.

4. HYBRID COLAV SYSTEM

The MDW algorithm returns promising results as a reactive COLAV method for ASVs (Serigstad, 2017). The algorithm is, however, reactive and there is a significant risk that the ASV can get stuck in a local minima. To reduce this risk, a hybrid architecture allowing the use of deliberate algorithms is introduced.

4.1 Interface between reactive and deliberate COLAV

A hybrid COLAV system for ASVs using the DW algorithm which takes global paths as input is proposed in (Loe, 2008). However, not taking trajectories as input excludes the use of deliberate methods that takes moving obstacles into account. To combine a deliberate and a reactive COLAV method, an interface between the methods is required. One alternative is to generate a desired surge speed and yaw rate (u'_d, r'_d) using a trajectory tracker, based on a trajectory from the deliberate algorithm. In the MDW algorithm, the desired velocity (u'_d, r'_d) are already chosen based on an external guidance system. Hence, any trajectory tracker returning a desired velocity pair are suitable as the interface between the MDW algorithm and a deliberate method.

Alternatively, we can include the global trajectory directly into the DW algorithm and chose velocity pairs by how well the corresponding predicted trajectory aligns with the global trajectory. This will remove the need of an external guidance system and make the DW algorithm suitable with any global method returning a time parametrized trajectory with the ASV position

$$\boldsymbol{p}_{gt}(t) = [x_{gt}(t), y_{gt}(t)]^\top, \tag{38}$$

at any time $t \geq 0$ along the trajectory. The DW algorithm will proactively favor velocities which are predicted to lead the vessel along the global trajectory ahead in time. To make the DW algorithm suitable for hybrid architectures, we introduce the hybrid dynamic window (HDW) algorithm which takes the global trajectory as input as illustrated in Figure 2. In this paper, the RRT algorithm is used as a deliberate method when simulating the hybrid COLAV system using the HDW algorithm.

4.2 Hybrid Dynamic Window (HDW) algorithm

The HDW algorithm is intended to select velocity pairs that follow the global trajectory. However, a single velocity pair may not yield a suitable trajectory since the desired yaw rate may change along the global trajectory. To improve on this, we allow for branches by changing the velocity pair during the trajectory. The trajectories are generated by first making predictions based on a single velocity pair within the search space as defined in (24). By using the end state of every predicted trajectory from (24) as current state, a new search space is generated in the same manner. For each new search space, new trajectory predictions are generated based on the velocity pairs within it, as shown in Figure 3, where two velocity pairs are used to define a trajectory. A similar modification is presented in (Ögren and Leonard, 2005). Note that by using multiple velocity pairs to define trajectories, the computation load increases exponentially with the numbers of branches.

We hence define a trajectory as a sequence of M velocity pairs:

$$\begin{aligned}&((u_1, r_1), (u_2, r_2), ..., (u_M, r_M))\\ \in&(\mathbb{R}\times\mathbb{R})\times(\mathbb{R}\times\mathbb{R})\times ...\times(\mathbb{R}\times\mathbb{R}),\end{aligned} \tag{39}$$

where (u_1, r_1) is the first velocity pair, (u_2, r_2) the second and so on. To simplify the notation, we define a sequence of velocity pairs as $(\boldsymbol{u}, \boldsymbol{r}) = ((u_1, r_1), (u_2, r_2), ..., (u_M, r_M))$, where $\boldsymbol{u} = [u_1, u_2, ..., u_M]^\top$ and $\boldsymbol{r} = [r_1, r_2, ..., r_M]^\top$. Using this notation, we write (39) to:

$$(\boldsymbol{u}, \boldsymbol{r}) \in \bar{V_r}, \tag{40}$$

where $\bar{V_r} = (\mathbb{R}\times\mathbb{R})\times(\mathbb{R}\times\mathbb{R})\times ...\times(\mathbb{R}\times\mathbb{R})$.

The predicted trajectory of a set of velocity pairs yields a complete vessel state prediction $\boldsymbol{p}_{pt}(t)$ at time t along the trajectory defined as:

$$\boldsymbol{p}_{pt}(t) = [x_{pt}(t), y_{pt}(t)]^\top. \tag{41}$$

The predicted trajectories are discretized into N points between the current time t and $t + t_{pt}$, where t_{pt} is the time frame of the predicted trajectory. Using the predicted trajectories, we define the a measurement of alignment between a predicted trajectory and the global trajectory:

$$align(\boldsymbol{p}_{pt}, \boldsymbol{p}_{gt}) = \frac{k_a}{N}\sum_{i=1}^{N}\left\|\boldsymbol{p}_{pt}(t_i), \boldsymbol{p}_{gt}(t_i))\right\|_2, \tag{42}$$

where t_i is expressed as:

$$t_i = t + \frac{i \cdot t_{pt}}{N}, \tag{43}$$

and $k_a = 1$ $[1/m]$ makes the function unitless.

An optimization problem for the HDW algorithm is defined as:

$$\begin{aligned}\max_{(\boldsymbol{u},\boldsymbol{r})} G(\boldsymbol{u}, \boldsymbol{r}) =&\bar{\alpha}\sum_{i=1}^{M} dist(u_i, r_i)\\ &- (1-\bar{\alpha})align(\boldsymbol{p}_{pt}, \boldsymbol{p}_{gt})\\ s.t.\ (\boldsymbol{u}, \boldsymbol{r}) \in \bar{V_r},&\end{aligned} \tag{44}$$

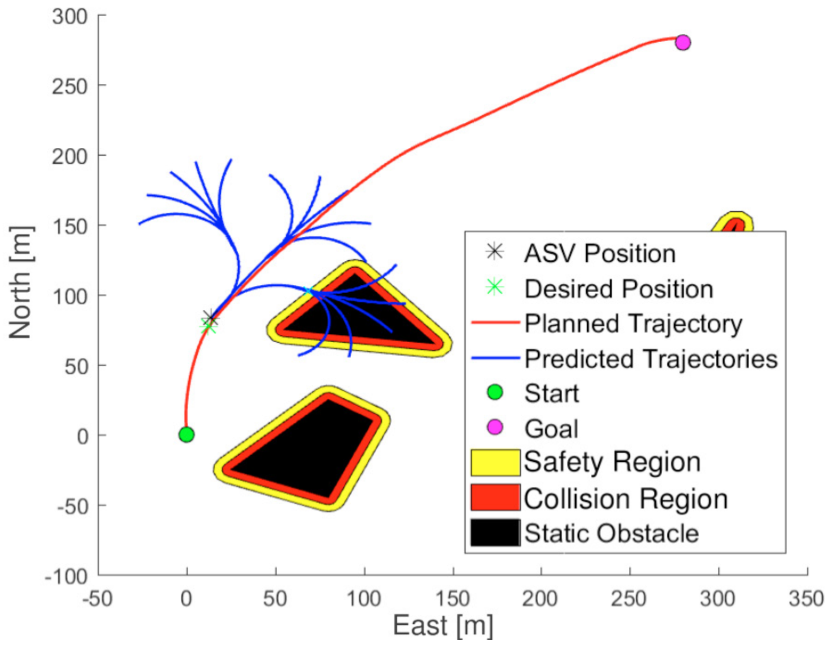

Fig. 3. Trajectory branching, where each trajectory is defined by two velocity pairs.

where the tuning parameter $\bar{\alpha} \in [0,1]$ sets the weighting between the obstacle distance and the trajectory alignment in the cost function. Specifically, $\bar{\alpha}$ sets how risk averse the ASV is when tracking the trajectory. By increasing $\bar{\alpha}$ towards 1, the HDW algorithm will favor avoiding the safety region over following the global trajectory. On the other hand, if $\bar{\alpha} = 0$, the algorithm will only care about following the global trajectory, but will still only consider admissible velocity pairs. Note that the distance function uses the set of multiple velocity pairs $(\boldsymbol{u}, \boldsymbol{r})$ defined in (39).

Now, the cost function favors velocity pairs that will lead the ASV along the global trajectory in the near future, and takes upcoming turns into account. The HDW algorithm functions both as a reactive method and a trajectory tracker in a hybrid architecture which can apply any deliberate method that generates a trajectory. Notice that the HDW algorithm only has one tuning parameter, while the DW algorithm has three.

To make the algorithm able to converge to the global trajectory, we use a trajectory tracker adding velocity pair suggestions to the search space. A method for tracking paths is described in (Breivik and Fossen, 2004). The method returns a desired surge speed u_d and heading ψ_d, which is used in a controller to find the desired yaw rate r'_d for the DW algorithm using (37) as control law. The desired speed can be defined as:

$$u'_d = \begin{cases} \frac{U_{gt} - \gamma s}{cos(\chi - \chi_t)}, & \text{if } \chi - \chi_t \neq 0 \\ u_{max}, & \text{if } \chi - \chi_t = 0, \end{cases} \tag{45}$$

where U_{gt} is the surge speed of the globally planned trajectory. If the velocity pair (u'_d, r'_d) is within the range of the search space, $(u'_d, r_i) \forall i \in \{1, \ldots, N_r\}$ and $(u_j, r'_d) \forall j \in \{1, \ldots, N_u\}$ are added to the search space, where N_u and N_r are the amounts of surge speeds and yaw rates in the search space, respectively. Note that the HDW algorithm can receive suggestions from an external trajectory tracker, but is not dependent on it.

4.3 A new distance function

When a vehicle using the MDW algorithm resides inside the safety region, the distance function yields zero for all velocity pairs and loses the ability to favor trajectories keeping a safe distance from obstacles (Serigstad, 2017). By using the predicted trajectory, we define a measure of the portion of a trajectory residing inside the safety region:

$$\tilde{\rho}(u, r) = \frac{\sum_{n=1}^{N} \frac{\lambda(u,r,n)}{\sqrt{n}}}{\sum_{n=1}^{N} \frac{1}{\sqrt{n}}} \in [0, 1], \tag{46}$$

where $\lambda(u, r, n) = 0$ if part n of the trajectory resides inside Ω, and $\lambda(u, r, n) = 1$ otherwise.

To prevent the ASV from choosing trajectories cutting through the safety region unnecessarily, we define a new distance function:

$$dist(u, r) = \kappa \frac{\bar{\rho}(u, r)}{\int_0^T \left\| \boldsymbol{\chi}(u, r, t) \right\|_2 dt} + (1 - \kappa)\tilde{\rho}(u, r) \in [0, 1], \tag{47}$$

where $\bar{\rho}$ is defined in (23), and $\kappa \in [0, 1]$ is a tuning parameter deciding the weight between subfunctions $\bar{\rho}(u, r)$ and $\tilde{\rho}(u, r)$. Finally, the new distance function is used in the optimization function (44).

Compared to the MDW distance function, the new distance function enables the algorithm to consider what lies beyond the entrance of the safety region and whether a trajectory leads out of the region. Consequently, the algorithm is less hesitant to enter the safety region when the predicted trajectory leads through the region, compared to the MDW algorithm (Serigstad, 2017). Notice that this distance function adds an extra tuning parameter κ, resulting in two tuning parameters for the HDW algorithm.

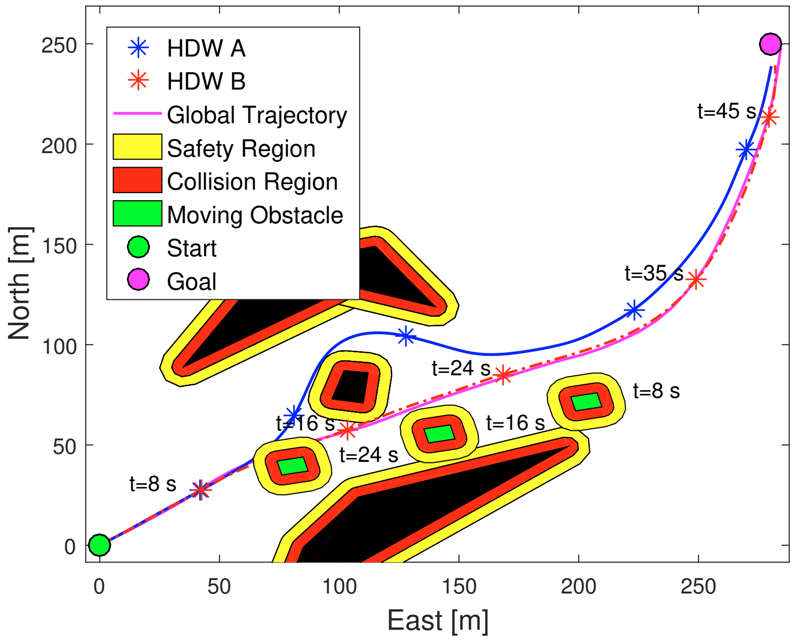

Fig. 4. Simulation of HDW A and HDW B at Bleikja.

Table 1. Tuning parameters used in the simulations for the hybrid COLAV architecture

Constant	Value	Description
$r_{\mathcal{T}}$	5 m	Collision region size
r_{Ω}	10 m	Safety region size
T_s	1 s	Dynamic window sample time
T_a	0.8 s	Time limit for changing rudder angle
t_{pt}	12 s	Time frame of the DW trajectories
$\bar{\alpha}$	0.98	HDW weighting parameter
κ	0.5	Tuning parameter for distance function
M	2	Number of velocity pairs per trajectory
N	40	Parts in the discretized trajectories

5. SIMULATION RESULTS

This section contains simulations of the hybrid COLAV architecture using the distance function from the MDW algorithm (36), denoted HDW A, and the modified distance function (47), denoted HDW B. The ASV model parameters used for simulation are described in detail in (Loe, 2008).

5.1 Performance metrics

To evaluate the algorithm performance, we introduce performance metrics measuring change of control input, distance from obstacles, and the algorithms ability to track the global trajectory.

To combine surge and yaw actuation in one metric, we need to introduce weighting since the units are different. We compute a normalized control input:

$$\bar{\tau}(t) = \sqrt{\bar{X}(t)^2 + \bar{N}(t)^2}, \tag{48}$$

where $\bar{X}(t) \in [0,1]$ and $\bar{N}(t) \in [-1,1]$ for the expected operating region of the ASV (Eriksen and Breivik, 2017) defined by the rudder angle constraints and propeller saturation from (11) and (13), respectively. Given this signal, the integral of absolute differentiated control (IADC) is defined as:

$$IADC(t) = \int_0^t \left|\dot{\bar{\tau}}(\sigma)\right| d\sigma, \tag{49}$$

which is a measure of the actuator wear and tear, where $\dot{\bar{\tau}}(\sigma)$ is numerically derived.

The integral of the distance inside the safety region (IDI) is expressed as:

$$\text{IDI}(t) = \int_0^t \bar{\lambda}(\sigma) d\sigma, \tag{50}$$

where $\bar{\lambda}(\sigma)$ is defined as:

$$\bar{\lambda}(t) = \begin{cases} 1 - \frac{D_{\mathcal{T}}(t)}{r_{\Omega} - r_{\mathcal{T}}}, & \text{if } D_{\mathcal{T}} < r_{\Omega} - r_{\mathcal{T}} \\ 0, & \text{otherwise,} \end{cases} \tag{51}$$

which gives the distance inside the safety region Ω if $D_{\mathcal{T}} < r_{\Omega} - r_{\mathcal{T}}$ is satisfied, where $D_{\mathcal{T}}(t)$ is the distance from the ASV to the closest point in the collision region $\mathcal{T}$. The metric penalizes staying inside the safety region over time and also considers how far inside the safety region the ASV is.

Considering that the hybrid architecture consists of a deliberate method generating a global trajectory and a reactive method guided by it, an error between the desired and actual ASV trajectory is computed as:

$$e(t) = \left\| \boldsymbol{x}_d(t) - \boldsymbol{x}(t) \right\|_2, \tag{52}$$

where $\boldsymbol{x}(t) = [x(t), y(t)]^\top$ denotes the position of the ASV, and $\boldsymbol{x}_d(t)$ denotes the desired position of the ASV at time t. The integral of the absolute error is expressed as:

$$\text{IAE}(t) = \int_0^t \left|e(\sigma)\right| d\sigma. \tag{53}$$

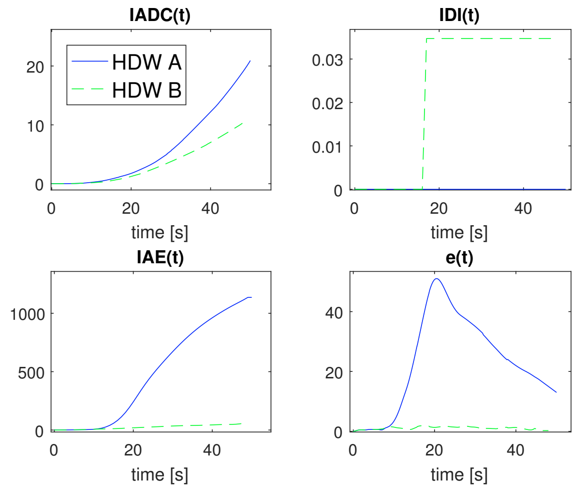

Fig. 5. Performance metrics from simulations in Figure 4.

5.2 Results

The simulation scenario contains both static and dynamic obstacles and is inspired by a narrow ferry passage around the island Bleikja in Hardangerfjorden, Norway. The HDW algorithm tracks a global trajectory generated by the RRT algorithm. The results in figures 4 and 5 illustrate how the HDW algorithm enables the ASV to follow the global trajectory and to catch up with it if falling behind. Furthermore, the results show how HDW A leaves the global trajectory to avoid entering the safety region. HDW B, however, tracks the global trajectory successfully at the cost of barely cutting through the safety region, as illustrated by the IAE and IDI metric in Figure 5. The $IADC$ metric shows how following the global trajectory without rerouting yields a lower actuator wear and tear. HDW B outperforms HDW A in all metrics except IDI, due to touching the safety region, and is concluded to have the best performance. The goal has a 15 meter acceptance radius, which is the reason why HDW A appears to have a significantly high e value when reaching goal.

For more simulation results including narrow paths, local minima, and other environments with moving obstacles, the interested reader is referred to (Serigstad, 2017).

6. CONCLUSION

We have modified the dynamic window (DW) algorithm to be suitable in a hybrid collision avoidance (COLAV) architecture. The hybrid dynamic window (HDW) algorithm is able to successfully track the global trajectory and catch up with it if falling behind, independent of the distance function. This allows the HDW algorithm to perform real-time COLAV without loosing track of the global trajectory. The HDW algorithm is compatible with any deliberate algorithm generating a trajectory. A branching of the DW trajectory predictions is applied, enabling the algorithm to change desired velocity during the trajectory prediction. The trajectory prediction branching generates trajectories that are more likely to align with the global trajectory, at the cost of moderately increased algorithm computational cost.

Applying a new distance function enables the HDW algorithm to track the global trajectory close to obstacles if necessary. However, it will only guide the autonomous surface vehicle (ASV) close to obstacles if the predicted vehicle trajectory passes the obstacle and stays clear of it. Based on simulation results and performance metrics, the distance function is believed to improve trajectory tracking and reduce actuator wear and tear, when operating close to obstacles.

For further development of the HDW algorithm, it should be thoroughly tested in full-scale ASV experiments.

ACKNOWLEDGEMENTS

This work was supported by the Research Council of Norway through project number 244116 and the Centres of Excellence funding scheme with project number 223254.

REFERENCES

Breivik, M. and Fossen, T.I. (2004). Path following for marine surface vessels. In *Proceedings of the OTO04, Kobe, Japan.*

Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *Proceedings 1999 IEEE International Conference on Robotics and Automation, Detroit, Michigan.*

Eriksen, B.-O.H. and Breivik, M. (2017). Modeling, identification and control of high-speed ASVs: Theory and experiments. In *Sensing and Control for Autonomous Vehicles: Applications to Land, Water and Air Vehicles*, 407–431. Springer International Publishing.

Eriksen, B.-O.H., Breivik, M., Pettersen, K.Y., and Wiig, M.S. (2016). A modified dynamic window algorithm for horizontal collision avoidance for AUVs. In *Proc. of 2016 IEEE Conference on Control Applications (CCA), Buenos Aires, Argentina.*

Eriksen, B.-O.H., Wilthil, E.F., Flåten, A.L., Brekke, E.F., and Breivik, M. (2018). Radar-based maritime collision avoidance using dynamic window. In *Proc. of IEEE Aerospace, Big Sky, Montana, USA.*

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. In *Int. J. Robot. Res.*, volume 17, 760–772.

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

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. In *IEEE Robotics and Automation Magazine*, volume 4(1), 23–33.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept., Iowa State University.

Loe, Ø.A.G. (2008). *Collision Avoidance for Unmanned Surface Vehicles.* Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Serigstad, E. (2017). *Hybrid Collision Avoidance for Autonomous Surface Vessels.* Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Ögren, P. and Leonard, N. (2005). A convergent dynamic window approach to obstacle avoidance. In *IEEE Transactions on Robotics*, volume 21(2), 188–195.