Warm-Started Optimization-Based Collision Avoidance

Paal Arthur S. Thorseth
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, USA
pathorse@mit.edu

***Abstract*—We present a optimization based collision avoidance approach. Using recent results in optimization-based control, we show that the collision avoidance path planning problem can be reformulated to a smooth non-convex optimization problem. Since the presented optimization problem is numerically challenging to solve we present a warm-started OBCOLAV algorithm. The algorithm combines advantages from two planners, the A^* algorithm and optimization-based trajectory planning. The idea is that the A^* computes a globally and optimal collision free path used as an initial guess in order to warm-start the optimization problem. Our studies show that the proposed warm-started OBCOLAV algorithm is capable of finding dynamically feasible optimal solutions, while avoiding obstacles.**

I. Introduction

Autonomous vehicles depend heavily on their ability to exploit knowledge about their environment, and plan their actions accordingly. For underactuated surface vessels (USVs) an important challenge is to plan a path in an efficient and safe manner. A fundamental part of the planning is the USVs capability to avoid obstacles such as land, shallow waters and other USVs. Despite extensive research on the problem of generating collision-free trajectories in a cluttered enviroment, the task still remains difficult. The difficulties arise with the non-linear USV dynamics and the non-convex nature of the collision avoidance problem. In general, non-convex optimization problems are often NP-hard.

The path planning problem has several notable solutions in the existing litterature. The motivation for choosing this project has been to explore simple, yet capable, optimization based collision avoidance algorithms. This is done to familiarize myself with topics in the maritime autonomy environment, which I will be working with in the near future through DNV GL's student project ReVolt.

The following work is heavily inspired by [1], where a novel method for reformulating non-differentiable collision avoidance constraints into smooth linear constraints using the strong duality of convex optimization is presented. Moreover, the work presented in the paper has inspired others, such as [2], where a warm-started optimized trajectory planning for autonomous surface vehicles are presented. Both these papers serve as a fundamental basis for the presented material in this paper.

We can summarize the contributions of this paper as the following

- We show that if the controlled objective is described by a point mass and the obstacles described by convex polytopes, then the collision-avoidance constraint can effectively be exactly reformulated as a set of smooth, non-convex constraints. This is achieved by utilizing the strong duality of convex optimization when the collision-avoidance constraint is formulated using the minimum translation distance.
- We present a optimal control problem that solves the collision avoidance problem.
- We propose a hierarchical, 3-step, optimization-based collision avoidance algorithm, where the A^* algorithm is used to generate a global and optimal path between a start and goal position. More so, the resulting path is modified to be more dynamically feasible using Dubins path before being used as a initial guess for our optimal control problem.
- We demonstrate through simulation that the proposed algorithm works and are able to find dynamically feasible solutions in an efficient manner.

II. Theoretical background

A. System Dynamics

We model the USV dynamics using the simplified horizontal-plane model in surge, sway and yaw, i.e. the 3-Degree-of-Freedom (DOF) representation for marine crafts. This model is based on the assumption that the elements corresponding roll, pitch and heave is neglectable and takes the form [3]

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

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}\boldsymbol{\nu} = \boldsymbol{\tau}(\mathbf{u}) \tag{1b}$$

The pose vector $\boldsymbol{\eta} = [x^n, y^n, \psi]^T$ contains the USV's position and heading in the North-East-Down (NED) frame, whereas the velocity vector $\boldsymbol{\nu} = [u, v, r]^T$ contains the USV's body-fixed velocities in surge, sway and yaw. The rotation matrix $\mathbf{R}(\psi)$ is defined as

$$\mathbf{R}(\psi) := \mathbf{R}_{z,\psi} = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

and transform the body-fixed velocities to NED. The matrices $\mathbf{M}$, $\mathbf{C}(\boldsymbol{\nu})$ and $\mathbf{D}$ represents the system inertia, Coriolis and

Centripetal effects, and damping effects, respectively. The input vector $\mathbf{u} = [\tau_u, \tau_r]^T$ contains surge force and sway moment, and is mapped to the control force vector $\boldsymbol{\tau}(\mathbf{u}) = [\tau_u, 0, \tau_r]^T$.

The dynamics (1) of the USV is summarized in a compact form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} \mathbf{R}(\psi)\boldsymbol{\nu} \\ \mathbf{M}^{-1}(-\mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{D}\boldsymbol{\nu} + \boldsymbol{\tau}(\mathbf{u})) \end{pmatrix} \quad (3)$$

where $\mathbf{x} = [x^n, y^n, \psi, u, v, r]^T \in \mathbb{R}^6$ is the state vector, $\mathbf{u} \in \mathbb{R}^2$ is the control input, and $\mathbf{f} : \mathbb{R}^6 \times \mathbb{R}^2 \to \mathbb{R}^6$ describes the dynamics.

The state and input constraints are described by

$$\mathbf{h}(\mathbf{x}, \mathbf{u}) \leq 0 \quad (4)$$

where $\mathbf{h} : \mathbb{R}^6 \times \mathbb{R}^2 \to \mathbb{R}^{n_h}$ is a smooth function, with n_h being the number of constraints.

B. Obstacle and USV modeling

We consider $M \geq 1$ obstacles $\mathbb{O}^{(1)}$, $\mathbb{O}^{(2)}$,..., $\mathbb{O}^{(M)} \subset \mathbb{R}^2$ described by its half-space representation (H-representation) as

$$\mathbb{O}^{(m)} = \{\mathbf{p} \in \mathbb{R}^2 \mid \mathbf{A}^{(m)}\mathbf{p} \leq \mathbf{b}^{(m)}\} \quad (5)$$

where $\mathbf{A}^{(m)} \in \mathbb{R}^{l_m} \times \mathbb{R}^2$ and $\leq \mathbf{b}^{(m)} \in \mathbb{R}^{l_m}$ with l_m being the number of facets of the m-th obstacle. We assume that the obstacles $\mathbb{O}^{(m)}$ are convex compact sets with non-empty relative interior. This is a fair assumption to make as most non-convex obstacles can be approximated or decomposed into convex unions of obstacles.

We denote the "space" occupied by the USV as $\mathbb{E}(\mathbf{x}) \subset \mathbb{R}^2$ for a given state in time $\mathbf{x}$. For the sake of simplicity we model $\mathbb{E}(\mathbf{x})$ as a point-mass model such that $\mathbb{E}(\mathbf{x})$ simply maps the state $\mathbf{x}$ to $\mathbf{p}$

$$\mathbb{E}(\mathbf{x}) = \mathbf{p} \quad (6)$$

where $\mathbf{p} = [x^n, y^n]^T$ is the NED position.

III. Optimization formulations

A. Optimization Problem

We discretize our dynamics (3a) using the Forward Euler Method with step length Δt such that

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \cdot \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \quad (7)$$

where $\mathbf{x}_k$ and $\mathbf{u}_k$ are state and input evaluated at time step k. Accordingly the discrete time state and input constraints simply becomes

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \quad (8)$$

We let N denote the number of time steps, such that $k \in \{0, 1, ..., N\}$, and we define the collection of all states $\boldsymbol{x} := [\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N]$ and all inputs $\boldsymbol{u} := [\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{N-1}]$.

The cost function is formulated so that it minimizes control input, i.e. energy usage, and are defined as

$$J(\boldsymbol{u}) := \sum_{k=0}^{N-1} \mathbf{u}_k^T \mathbf{Q} \mathbf{u}_k \quad (9)$$

where $\mathbf{Q}$ is a positive definite weighting matrix.

At last the collision avoidance constraint at time step k can be formulated, using the definitions presented in II-B, as

$$\mathbb{E}(\mathbf{x}_k) \cap \mathbb{O}^{(m)} = \emptyset \quad \forall \ m = 1, 2, ..., M \quad (10)$$

The finite-horizon optimal control problem can now be formulated using (7) - (10) and becomes

$$\begin{aligned} &\min_{\boldsymbol{x}, \boldsymbol{u}} J(\boldsymbol{u}) = \sum_{k=0}^{N-1} \mathbf{u}_k^T \mathbf{Q} \mathbf{u}_k \\ &\textit{subject to} \\ &\mathbf{x}_0 = \mathbf{x}_S, \quad \mathbf{x}_N = \mathbf{x}_G \\ &\left.\begin{aligned} &\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \cdot \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ &\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\ &\mathbb{E}(\mathbf{x}_k) \cap \mathbb{O}^{(m)} = \emptyset \end{aligned}\right\} \forall \begin{array}{l} k = 0, 1, ..., N-1 \\ m = 1, 2, ..., M \end{array} \end{aligned} \quad (11)$$

One of the major difficulties in solving (10) is the non-convex and non-differentiable collision constraint (10). In the following we will present a rewriting of the collision constraints for it to obtain both the property of being continious and differentiable. The rewriting will enable us to use efficient optimization algorithms for solving the problem.

B. Reformulation of the Optimization Problem

The collision constaint (10) can be formulated using the notion of minimum translation distance, which is common in collision detection. The minimum translation distance between the two sets $\mathbb{E}(\mathbf{x})$ and $\mathbb{O}$ is defined as

$$\text{dist}(\mathbb{E}(\mathbf{x}), \mathbb{O}) = \min_{\mathbf{t}} \{\|\mathbf{t}\|_2 \mid \mathbb{E}(\mathbf{x} + \mathbf{t}) \cap \mathbb{O} \neq \emptyset\} \quad (12)$$

and is nonzero and positive when the sets are non-intersecting. Followingly the collision constraint (10) becomes

$$\text{dist}(\mathbb{E}(\mathbf{x}), \mathbb{O}) > 0 \quad (13)$$

Unfortunately directly enforcing (13) in our optimization doesn't solve the difficulties regarding convexity and differentiability.

To overcome the issues we turn to the reformulation presented in [1], where it is shown that by adding an additional decision, variable (13) can be reformulated to a smooth and exact collision constraint.

Theorem 1 ([1] Proposition 1.): Assume that the obstacle $\mathbb{O}$ and the controlled object $\mathbb{E}(\mathbf{x})$ are given as in (5) and (6), and let d_{min} be a non-negative desired safety distance between the obstacle and the object. Then we have

$$\begin{aligned} &\text{dist}(\mathbb{E}(\mathbf{x}), \mathbb{O}) > d_{min} \\ &\Updownarrow \\ &\exists\, \boldsymbol{\lambda} \geq 0 \ \mid \ (\mathbf{A}\mathbf{p} - \mathbf{b})^T \boldsymbol{\lambda} > d_{min}, \ \|\mathbf{A}^T \boldsymbol{\lambda}\|_2 \leq 1 \end{aligned} \quad (14)$$

Proof: The dual problem of (12) is given by $\max_{\boldsymbol{\lambda}} \{(\mathbf{A}\mathbb{E}(\mathbf{x}) - \mathbf{b})^T \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \geq 0, \ \|\mathbf{A}^T \boldsymbol{\lambda}\|_2 \leq 1\}$. Since the obstacles $\mathbb{O}$ are assumed to be convex and have non-empty interior, strong duality holds and $\text{dist}(\mathbb{E}(\mathbf{x}), \mathbb{O}) =$

$\max_{\boldsymbol{\lambda}}\{(\mathbf{A}\mathbb{E}(\mathbf{x})-\mathbf{b})^T\boldsymbol{\lambda} \mid \boldsymbol{\lambda}\geq 0,\ \|\mathbf{A}^T\boldsymbol{\lambda}\|_2 \leq 1\}$. From here it follows that for a non-negative scalar d_{min}, the collision constraint (13) is satisfied if, and only if, there exists $\boldsymbol{\lambda}\geq 0 \mid (\mathbf{A}\mathbb{E}(\mathbf{x})-\mathbf{b})^T\boldsymbol{\lambda}\geq d_{min},\ \|\mathbf{A}^T\boldsymbol{\lambda}\|_2\leq 1$

■

The optimal control problem (11) can now be reformulated using (13) and 1 as

$$\begin{aligned}
&\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\Lambda}} J(\boldsymbol{u}) = \sum_{k=0}^{N-1} \mathbf{u}_k^T\mathbf{Q}\mathbf{u}_k \\
&\textit{subject to} \\
&\mathbf{x}_0 = \mathbf{x}_S, \quad \mathbf{x}_N = \mathbf{x}_G \\
&\left.\begin{aligned}
&\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t\cdot\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k) \\
&\mathbf{h}(\mathbf{x}_k,\mathbf{u}_k)\leq 0 \\
&\boldsymbol{\lambda}_k^{(m)}\geq 0 \\
&(\mathbf{A}^{(m)}\mathbf{p}_k-\mathbf{b}^{(m)})^T\boldsymbol{\lambda}_k^{(m)}>0 \\
&\|\mathbf{A}^{(m)^T}\boldsymbol{\lambda}_k^{(m)}\|_2\leq 1
\end{aligned}\right\}\ \forall \quad \begin{aligned}&k=0,1,...,N-1\\&m=1,2,...,M\end{aligned}
\end{aligned} \tag{15}$$

where $\boldsymbol{\Lambda} = [\boldsymbol{\lambda}_0^{(1)},...,\boldsymbol{\lambda}_0^{(M)},.....,\boldsymbol{\lambda}_N^{(1)},...,\boldsymbol{\lambda}_N^{(M)}]$ denotes the dual variables, and $\mathbf{A}^{(m)}$, $\mathbf{b}^{(m)}$, $\mathbf{p}_k$ is as detailed in II-B, where again the subscript k represents the value evaluated at time step k. The other expressions remain as detailed in III-A. Moving forward we will refer to (15) as OBCOLAV, optimization-based collision avoidance.

IV. OBCOLAV Algorithm

The optimization problem (15) is a non-convex and non-linear problem, which is computationally difficult to solve in general. Such problems introduce a number of local optimums, and can rarely be solved to global optimality. In practice this leads to the case where one often has to satisfy oneself with a local optima. The convergence to these local optimums heavily depend on the initial guess that is presented to the solver. Different initial guesses will tend the solution towards different local optimas. The process of starting the optimization with an initial guess is often referred to as "warm-starting" the optimization, and serves a key role in convergence and computational efficiency for non-convex optimization problems.

In the following sections we propose a hierarical, 3-step, OBCOLAV algorithm for USVs. We first use the A^* algorithm to compute a global shortest collision-free path between the start and goal position, and then we make the computed path more dynamically feasible by creating a trajectory using Dubins Path. The resulting trajectory is at last used to warm-start the OBCOLAV (15).

A. A^* Algorithm

The $A*$ algorithm war originally presented in [4], and is widely known. The search algorithm looks for the shortest collision-free path between nodes in a uniformly decomposed grid, where Δd is the chosen grid resolution and Euclidean distance is used as cost and heuristic function.

The current choices are made for the sake of simplicity and one could boost performance using a different grid decomposition. It is also worth noting that the grid resolution Δd heavily affects the run-time, and must therefore be chosen carefully to achieve acceptable run-times, without restricting the algorithm from finding desired solutions, e.g. optimal routes through narrow passages.

B. Trajectory Generation

The path generated by the A^* algorithm provides a poor warm-start as it fails to take into account the USV's dynamics. To address this issue we utilize the famous result of [5], and can be summarized as *The shortest path (minimum-time) between two configurations $[x, y, \psi]$ of a craft moving at constant speed is a path formed by straight lines and circular arc segments*.

The path generated by the A^* Algorithm may contain many unnecessary waypoints, and can be reduced using a simple way-point reduction algorithm that backtracks the waypoints, only adding the ones needed to avoid collisions. The waypoints in the reduced path are connected using straight line segments and circle arcs to increase the geometric feasability, where by choosing the turning radius R_{turn} larger than the minimum turning radius for the USV R_{min} enables tight following of the path. We can now construct positional and directional functions parameterized by length along the path, and let the total path length be denoted by L_{path}. Since the desired behavior is to travel at a nominal speed along the path we sample along the path, where the step length is given as $\Delta L = L_{path}/N$. Here N again denotes the number of steps in our horizon.

The sampled trajectory can now be considered as a collection of initial guesses $\boldsymbol{s}_{guess} = [\boldsymbol{\eta}_{g_0}, \boldsymbol{\eta}_{g_1}, ..., \boldsymbol{\eta}_{g_N-1}]$, where $\boldsymbol{\eta}_{g_k} = [x_{g_k}^n, y_{g_k}^n, \psi_{g_k}]^T$ is the pose guess evaluated step k.

C. Warm-started OBCOLAV

Finally the OBCOLAV problem (15) is solved with the initial guess from IV-B. Since the OBCOLAV (15) is a smooth optimization problem it can be solved using a off-the-shelf nonlinear solver, e.g. IPOPT and SNOPT.

D. Algorithm Summary

The hierarchical, 3-step, OBCOLAV algorithm can be summarized by the following piece of pseudocode

Algorithm 1 Warm-started OBCOLAV algorithm

Input: Initial state $\mathbf{x}_S$, Final state $\mathbf{x}_G$, Step length Δt, Total number of steps N and Obstacles $\mathbb{O}^{(1)},...,\mathbb{O}^{(M)}$

Output: Optimal trajectory $\boldsymbol{x}^*$ and input sequence $\boldsymbol{u}^*$

1: Compute the shortest collision-free path using A^*
2: Compute initial guesses $\boldsymbol{\eta}_{g_k}$ using Dubins Path
3: Solve (15) using $\boldsymbol{\eta}_{g_k}$ as initial guesses in state $[x_k^n, y_k^n, \psi_k]^T$

V. Simulation Results

In this section, we provide simulation results for two maneuvering scenarios, see figure 1 - 2. The Python-based source code can be found at `https://github.com/Pathorse/OB_COLAV_USVs`.

A. Simulation Setup

The USV dynamics (3a) is described by matrices $\mathbf{M}$, $\mathbf{C}(\boldsymbol{\nu})$ and $\mathbf{D}$, which are given as

$$\mathbf{M} = \begin{pmatrix} 263.93 & 0 & 0 \\ 0 & 306.44 & 7.00 \\ 0 & 7.03 & 322.15 \end{pmatrix}$$

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{pmatrix} 0 & 0 & -207.56v + 7.00r \\ 0 & 0 & 250.07u \\ 207.56v - 7.00r & -250.07u & 0 \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} 50.66 & 0 & 0 \\ 0 & 601.45 & 83.05 \\ 0 & 83.10 & 268.17 \end{pmatrix}$$

The A^* algorithm uses a grid size $\Delta d = 4\ [m]$, where collision is checked by ensuring that the controlled object (6) does not lie within a obstacle (5). The obstacles are decomposed using a resolution of $0.1\ [m]$.

The maneuvering area is constrained for both test-cases, with scenario 1 being limited in x and y directions by $[x_{lb}, x_{ub}] = [-10, 200]$ and $[y_{lb}, y_{ub}] = [0, 100]$, and scenario 2 limited by $[x_{lb}, x_{ub}] = [-10, 450]$ and $[y_{lb}, y_{ub}] = [0, 200]$. The starting position for both scenarios is fixed at $[x_s^n, y_s^n] = [10, 10]$, while the goal position is $[x_g^n, y_g^n] = [180, 45]$ and $[x_g^n, y_g^n] = [430, 95]$ for scenario 1 and 2, respectively. All values are given as metres $[m]$.

The obstacles for scenario 1 is designed to represent a tight harbour passage where the USV have to navigate through the passage to arrive at the goal position. The passage is $20[m]$ wide, and the obstacles are modeled as squares, thus clearly satisfying (5). For scenario 2 the USV also have to travel through a passage, with two additional obstacles are added to raise path difficulty towards the goal. The obstacles are here again modeled by squares, with a further extension by adding a parallelogram. Again all obstacles satisfy (5).

All of the following simulations are performed on a 2019 MacBook Pro with an six core Intel Core i7 processor clocked at $2.6\ GHz$. Our simulations were implemented in Python, and the OBCOLAV problem (15) was set up using the Drake toolbox [6]. The optimization was solved using the non-linear solver SNOPT, but it is worth noting that the solver IPOPT could just as easily have been used as it is also supported in Drake.

B. Results

To evaluate the proposed algorithm IV we run the two maneuvering scenarios presented in V-A, with both the Warm-started OBCOLAV and the Cold-started OBCOLAV problems being simulated.

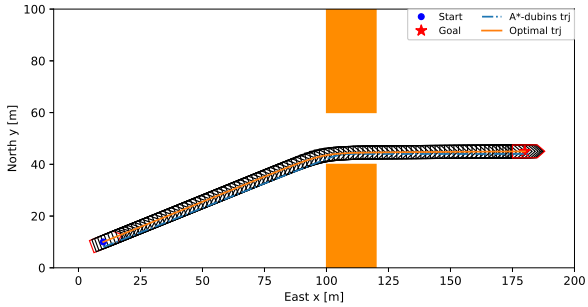

Fig. 1. Maneuvering Scenario 1

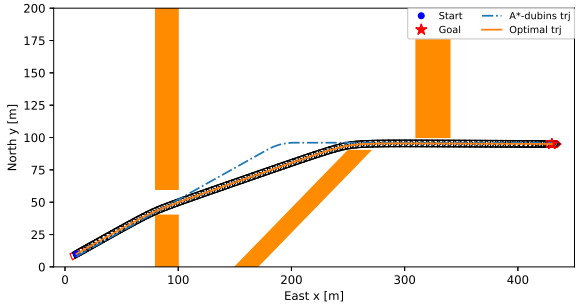

Fig. 2. Maneuvering Scenario 2

The trajectories generated by the Warm-started OBCOLAV for scenario 1 and scenario 2 is displayed in 1 and 2, respectively.

The computational times for the Warm-started OBCOLAV is reported in I, whereas the times for the Cold-started OBCOLAV is reported in II. The computation time is decomposed with respect to computing the A^* path presented in IV-A, solving the OBCOLAV problem presented IV-C and Total Run-Time for Algorithm 1. Note that the process of generating a more dynamically feasible initial guess presented in IV-B is not included in itself as it has a low run-time, it is however counted as part of the Total Run-Time. The two scenarios was run 10 times for the Warm-started OBCOLAV and 5 times for the Cold-started OBCOLAV in order to produce a min, max and mean value.

TABLE I
COMPUTATION TIME FOR A^* (IV-A), WARM-STARTED OBCOLAV (IV-C) AND TOTAL RUN-TIME (ALGORITHM 1)

	min	max	mean
Scenario 1			
A^*	0.1683	0.2194	0.1901
OBCOLAV	15.3617	16.1830	15.5251
Total Run-Time	15.0843	16.6205	15.9352
Scenario 2			
A^*	$0,7869$	$0,9191$	$0,8534$
OBCOLAV	$59,4785$	$63,9529$	$62,5268$
Total Run-Time	$60,7634$	$65,3023$	$63,8633$

* All values are given in seconds $[s]$.

TABLE II
COMPUTATION TIME FOR COLD-STARTED OBCOLAV (III-B)

	min	max	mean
Scenario 1			
Total Run-Time	64.5627	$68,9658$	67.1118

* All values are given in seconds $[s]$.

C. Discussion

We see from fig 1 and 2 that the proposed Warm-started OBCOLAV Algorithm is able to find dynamically feasible solutions that successfully navigate the USV from a start position to a goal position while avoiding obstacles.

1) Comparing the Warm- and Cold-started OBCOLAV: Table I and II shows the run-times for the warm-started OBCOLAV is significantly better when comparing to the cold-started run-times. In fact we see an improvement of 76% when comparing the two. This shows that the run-time costs of generating an initial guess is well worth it.

Moreover, we notice that for the cold-started OBCOLAV no run-times for scenario 2 is presented. This comes as a consequence of the solver being unable to produce a feasible solution, which again shows the value of providing a initial guess.

We close this section out by pointing out that the results suggest that not only does warm-starting the OBCOLAV provide improved run-times, it also helps finding feasible optimal solutions in situations where the cold-started OBCOLAV fails. This is an extremely important feature when dealing with non-convex optimization.

2) Computational Time: The computational times presented in table I shows that the mean total run-time becomes 15.9352 for scenario 1, and 63.8631 for scenario 2, with solving the OBCOLAV being the major run-time contributor. It therefore goes to show that by providing a more dense environment of obstacles one drastically increase run-times.

The presented run-times would clearly not be feasible for use in a Model Predictive Control Scheme (MPC), and would be better off when being stabilized using a controller, e.g. LQR or PID-controller.

VI. CONCLUSION

In this paper a Warm-started OBCOLAV Algorithm is proposed, implemented and tested. The proposed algorithm effectively combines the global and optimal path planning properties for A^* with the OBCOLAV's ability to compute dynamically feasible trajectories while avoiding obstacles. The results presented in V show that the warm-started OBCOLAV finds collision-free dynamically feasible solutions. Furthermore, the results show that the warm-started OBCOLAV improves both run-time and optimality when compared to the cold-started OBCOLAV.

The experiments have uncovered several points for improvements and expansions, and creates a foundation for future work.

- Improve the initial guess by providing adding more dynamical information. As of now the only dynamical information added is to the pose, there are still a lot of dynamical information left in the velocity.
- Improve the A^* grid decomposition. The current grid decomposition is a simple uniformly distributed decomposition, which can be altered to enhance performance.
- Implement the algorithm in C++ or Julia. By implementing the algorithm in a different programming language, where focus lies towards run-time efficiency one can improve the run-times.
- Implement the RRT^* algorithm as an alternative to A^* and compare performances. This can be desirable as RRT^* is better suited for scaling.
- Implement a simple controller in order to test that the USV is capable of tracking produced trajectories.
- Extend the controlled object to include a whole object, rather than the simplified point mass.

REFERENCES

[1] X. Zhang, A. Liniger, and B. Francesco, "Optimization-based collision avoidance," 2018.

[2] G. Bitar, V. N. Vestad, A. M. Lekkas, and M. Breivik, "Warm-started optimized trajectory planning for asvs," 2019.

[3] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 1st ed. John Wiley Sons, Ltd., 2011.

[4] P. E. H. Hart, N. J. Nilsson, and R. Bertram, "A formal basis for the heuristic determination of minimum cost paths," 1968.

[5] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[6] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2020. [Online]. Available: https://drake.mit.edu