# Optimal Model-Based Trajectory Planning With Static Polygonal Constraints

**Article** *in* IEEE Transactions on Control Systems Technology · July 2021

DOI: 10.1109/TCST.2021.3094617



**3 authors:**

Andreas Bell Martinsen
Norwegian University of Science and Technology
**16** PUBLICATIONS **62** CITATIONS
SEE PROFILE

Anastasios M. Lekkas
Norwegian University of Science and Technology
**45** PUBLICATIONS **806** CITATIONS
SEE PROFILE

Sebastien Gros
Norwegian University of Science and Technology
**173** PUBLICATIONS **1,630** CITATIONS
SEE PROFILE

# Optimal model-based trajectory planning with static polygonal constraints

Andreas B. Martinsen, Anastasios M. Lekkas, and Sébastien Gros

***Abstract*— The main contribution of this paper is a novel method for planning globally optimal trajectories for dynamical systems subject to polygonal constraints. The proposed method is a hybrid trajectory planning approach, which combines graph search, i.e. a discrete roadmap method, with convex optimization, i.e. a complete path method. Contrary to past approaches, which have focused on using simple obstacle approximations, or sub-optimal spatial discretizations, our approach is able to use the exact geometry of polygonal constraints in order to plan optimal trajectories. The performance and flexibility of the proposed method is evaluated via simulations by planning distance-optimal trajectories for a Dubins car model, as well as time-, distance- and energy-optimal trajectories for a marine vehicle.**

***Index Terms*— Autonomous vehicles, Marine vehicles, Mobile robots, Motion planning, Optimal control, Path planning, Trajectory optimization**

## I. Introduction

In robotics, motion planning is the process of finding a sequence of valid configurations, which move the robot safely from some initial configuration to a goal configuration. To be successful in the real world, the motion planner must be able to consider a variety of constraints such as *environment constraints*, including static and dynamic obstacles, and *differential constraints*, which arise from the system kinematics and dynamics and are modeled with differential equations. Due to a potentially large number of obstacles, actuators, as well as complex kinematics and dynamics, motion planning is in general a difficult problem that has led to a wide range of methods and a vast literature.

Trajectory planning pertains to finding a time-parametric continuous sequence of configurations, called a trajectory, which is obstacle-free and satisfies the differential constraints (i.e. a feasible trajectory). Optimal trajectory planning has the additional task of finding the ”best” feasible trajectory with respect to some performance measure, such as minimum energy, distance or time. The requirement of optimality is in general very demanding computationally since it requires an exhaustive search over the state space. One of many ways to categorize motion planning methods is to distinguish between *roadmap methods* and *complete path methods* [1]–[3].

The main goal of roadmap methods is to find a sequence of waypoints, which, when connected, result in an obstacle-free piecewise-linear path. The path can then be smoothed and turned into a feasible trajectory that complies with the vehicle dynamics. Roadmap methods can be further split into two distinct categories, namely, combinatorial methods and sampling-based methods. Combinatorial methods, divide the continuous space into structures that capture all spatial information needed to solve the motion planning using simple graph search algorithms. For many complex problems however, combinatorial methods may not be computationally feasible. For these problems, sampling based methods are often used instead. Sampling based methods, rely on using randomly sampled subset of states or actions. This creates a randomly sampled discretization of the continuous search space, and hence limits the computational complexity at the cost of accuracy and completeness of the discretization. Some notable combinatorial methods include coarse planning with path smoothing, in where a mesh, grid or potential field is used to plan a course path [4]–[7], and then a method using curve segments, splines or motion primitives is used to refine the trajectory [8]–[14]. Notable sampling based methods include probabilistic roadmap (PRM) [15], rapidly-exploring random tree (RRT) [16]–[18], and Random-walk planners [19], [20]

Complete path methods on the other hand, produce a continuous parameterized trajectory by explicitly taking into account the motion equations of the robot and the full continuous search space. As a result, these methods generate a trajectory that is both obstable-free and feasible, without further need of refinement/smoothing. Most complete path methods rely on some form of mathematical optimization. For some simple problems an analytical solution exists, as is the case for Dubins paths [21] and Reeds-Shepp [22]. In general, however, researchers must resort to numerical optimization, where handling complex constraints is challenging and getting stuck in local optima is not uncommon. Notable numerical methods include dynamic programming [23], particle swarm optimization (PSO) [24], [25], shooting methods [26], which are based on simulation, collocation methods [27], which are based on function approximation of low-level polynomials, and pseudospectral methods [28], which are based on function approximation of high-level polynomials.

In this paper we consider the problem of optimal motion

A. B. Martinsen is with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, NO-7491, Trondheim, Norway (e-mail: andreas.b.martinsen@ntnu.no).

A. M. Lekkas is with the Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, NO-7491, Trondheim, Norway (e-mail: anastasios.lekkas@ntnu.no).

S. Gros is with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, NO-7491, Trondheim, Norway (e-mail: sebastien.gros@ntnu.no).

planning for a particle-like vehicle, moving on a 2D surface with polygonal obstacles. To this end, we introduce a hybrid method, which combines graph search on a pre-computed mesh, with convex optimization for path refinement. The proposed method allows for planning a globally optimal trajectory for a dynamical system subject to static polygonal constraints. The main contributions is this paper is how we combine hybrid planning with polygonal constraints and triangulation based spatial discretization. With hybrid planning, we combine both roadmap and complete path methods. Contrary to other hybrid methods such as [29]–[31], where initial trajectories are planned using motion primitives and state space discretizations, and refined using numerical optimization, our method employs an iterative approach of planning and refinement. Polygonal constraints allow for complex constraints to be used in the planning algorithm. Very few optimization-based planning methods exist that are able to handle these types of constraints. Existing methods often lead to computationally expensive mixed integer optimization problems [32], rely on using inner approximations of the free space [33], [34], or non-convex elliptical approximations [2]. Our method relies on using a triangulation of the environment, similar to [4], [35] but instead of straight-line paths, it plans the path as a polynomial spline, similar to [36]. Combining the above concepts, our proposed method is able to efficiently plan globally optimal trajectories for a dynamical system subject to static polygonal constraints.

The rest of the paper is organized as follows: Section II outlines the method. Section III shows examples of distance-optimal paths for a simple kinematic car, as well as time-, distance- and energy-optimal paths for an unmanned surface vehicle. Finally Section IV concludes the paper.

## II. Method

The problem that we aim to solve in this paper, is that of planning optimal trajectories for dynamical systems in environments with static polygonal constraints. The proposed method is able to compute optimal time parameterized state trajectories:

$$\boldsymbol{x}(t), \quad t \in [t_0, t_f],$$

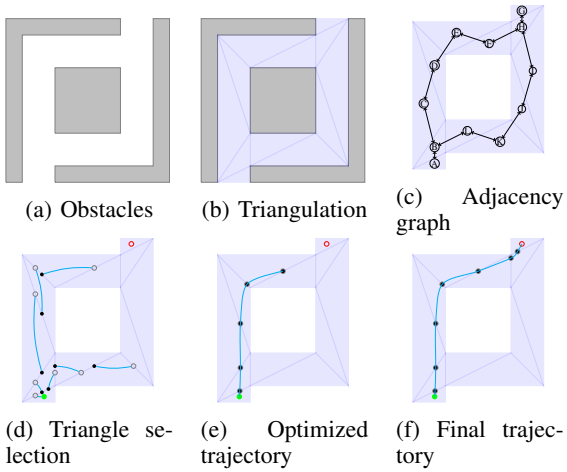
which connect some initial state $\boldsymbol{x}_0$ and final goal state $\boldsymbol{x}_f$ such that:

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{x}(t_f) = \boldsymbol{x}_f.$$

The trajectory is generated such that it satisfies the continuous time dynamics and kinematics of a given dynamical system on the form:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}),$$

which in general may be nonlinear and have additional constraints on the states and actions. The optimized trajectory, is found such that it avoids polygonal spatial constraints that are present in the environment. This is ensured by having the path travel through a sequence of neighbouring triangles $\mathcal{T}_i$, with the sequence denoted $[\mathcal{T}_0, \mathcal{T}_1, \dots \mathcal{T}_N]$, where the interior of each triangle is collision free. The proposed method for solving this problem can be divided into three distinct stages.



(a) Obstacles (b) Triangulation (c) Adjacency graph

(d) Triangle selection (e) Optimized trajectory (f) Final trajectory

Fig. 1: Given polygonal obstacles (a), the proposed algorithm finds the trajectory by creating a triangulation (b) and adjacency graph (c). Iteratively exploring different triangle sequences (d) where the refined trajectory is optimized as a spline (e). The exploration is performed until the goal is reached (f).

1) **Triangulation and adjacency graph** is the first stage, where a triangulation of the environment is generated based on the polygonal constraints (Figure 1b), and an adjacency graph is calculated based on neighbouring triangles (Figure 1c).
2) **Graph search** is the second phase, where a graph search algorithm is used to explore possible sequences of triangles in the triangulation (Figure 1c).
3) **Trajectory refinement** is the third phase, where a continuous trajectory is generated and optimized within the confinement of a sequence of triangles (Figure 1d and 1e).

### A. Triangulation and adjacency graph

In this step, the objective is to generate a triangulation of the environment, given polygonal spatial constraints. The resulting triangulation must include the edges of the polygons, which is referred to as *constrained triangulation*. The reason for segmenting the environment into triangles in this way, is that any triangle in this type of triangulation, is either fully inside of the polygonal constraint, or fully outside of the polygonal constraint. This results in an exact, and efficient decomposition of the environment. We can then use the triangles that are fully outside of the polygonal constraints in order to plan a sequence of triangles for the trajectory to pass through, which is guaranteed to be collision free.

In this work, the triangulation that we use, is a Constrained Delaunay Triangulation (CDT) [37]. A regular Delaunay triangulation (DT) [38] will maximize the minimum angle of all the angles of the triangles in the triangulation, and hence tend to avoid sliver triangles. With CDT, certain segments are forced into the triangulation. This is necessary in order to ensure that the triangles of the triangulation are either fully inside the polygonal spatial constraints, or fully outside the

spatial constraints. For the spatial constraints in Figure 1a, a constraint triangulation is given in Figure 1b.

After the triangulation is created, an adjacency graph is computed by connecting neighbouring triangles of the triangulation, where two triangles are considered neighbours if they share an edge. An illustration is shown in Figure 1c. The triangulation and adjacency graph are then used in the next phase for exploring and planning sequences of neighbouring triangles.

### B. Graph search

Graph search can in general only be used for planning in discrete environments. In order to extend it to the continuous domain, we propose using a trajectory refinement strategy, where the graph search is performed by planning a sequence of neighbouring triangles $[\mathcal{T}_0, \mathcal{T}_1, \dots \mathcal{T}_N]$, and a continuous time parameterized trajectory $\boldsymbol{x}(t)$, is planned within the constraints of the sequence of triangles.

Given a CTD, we can construct a graph, where each node represents a triangle, and edges are given by neighbouring triangles, this is illustrated in Figure 1c. The goal of the graph search is to plan a sequence of triangles $[\mathcal{T}_0, \dots \mathcal{T}_N]$, which optimizes a desired performance measure. In our case the goal is to optimize a time parameterized path integral on the form:

$$\int_{t_0}^{t_f} J(\cdot) d\tau, \tag{1}$$

where $J(\cdot)$ is a non-negative instantaneous cost. Given an initial starting point $\boldsymbol{x}_0$, the proposed graph search method, works by staring with the initial triangle sequence $[\mathcal{T}_0]$, such that $\boldsymbol{x}_0 \in \mathcal{T}_0$. It then iteratively extending the sequence of triangles $[\mathcal{T}_0, \dots \mathcal{T}_{N-1}]$, by adding new neighbouring triangels $\mathcal{T}_N$. This is performed until a feasible sequence of triangles $[\mathcal{T}_0, \dots \mathcal{T}_N]$, connecting the initial state $\boldsymbol{x}_0$ and final goal state $\boldsymbol{x}_f$, is found, and a termination condition is met. The order in which potential sequences are extended, is determined by a heuristics based lower bound on the path integral. This ensures that the potentially best paths are explored first, and hence reducing the number of triangle sequences that need to be explored.

### C. Trajectory refinement

In order to plan a continuous trajectory in an area divided into triangles, we can observe that the trajectory is constrained by the edge through which it enters, and the edge through which it leaves any given triangle. The point at which it leaves and enters a triangle is also the point at which the trajectory enters and leaves its neighbours respectively. It is therefore possible to plan a refined trajectory through each triangle, with a given entrance and exit point along the triangle boundary (see Figure 2). This means that the final optimal trajectory, which may travel through a non-convex polygon, consists of trajectory segments constrained to lie within individual convex triangles.

Given a dynamical system on the form:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{2}$$

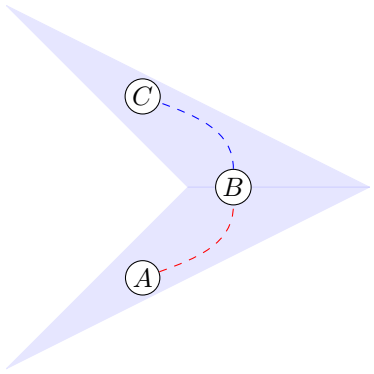



Fig. 2: The trajectory ($A \rightarrow C$) through two triangles can be planned as the trajectory through each individual triangle ($A \rightarrow B$ and $B \rightarrow C$), constrained to meeting somewhere along the neighbouring edge.

where $\boldsymbol{x}$ is the state vector, and $\boldsymbol{u}$ is the control vector. The optimal trajectory through a sequence of neighbouring triangles, denoted $[\mathcal{T}_0, \mathcal{T}_1, \dots \mathcal{T}_N]$, can be written as the following optimization problem.

$$V(\boldsymbol{x}_0, [\mathcal{T}_0, \mathcal{T}_1, \dots \mathcal{T}_N]) = \min_{\boldsymbol{x}, \boldsymbol{u}, t} \quad \sum_{i=0}^{N} \int_{t_i}^{t_{i+1}} J(\boldsymbol{x}, \boldsymbol{u}, \tau) d\tau \tag{3a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{3b}$$

$$\boldsymbol{x}(t) \in \mathcal{T}_i \quad \forall t \in [t_i, t_{i+1}] \tag{3c}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0. \tag{3d}$$

In the above optimization problem, (3b) ensures the trajectory is feasible with respect to the model, (3c) ensures each trajectory segment lies within its respective triangle, and (3d) gives the initial conditions for the optimization problem. Using the above formulation, we note that in the graph-search phase, the optimization problem is built by iteratively adding triangles to the triangle sequence $[\mathcal{T}_0, \mathcal{T}_1, \dots \mathcal{T}_N]$, and hence extending the horizon $N$. It should be noted that (3) can be extended to include additional state and input constraints. However, adding additional constraints may make the problem more difficult and time consuming to solve, and may lead to feasibility issues if the constraints are not chosen with care.

### D. Complete method

Given a trajectory $\boldsymbol{x}(t)$, starting at $\boldsymbol{x}_0$, and ending at $\boldsymbol{x}_f$, and going through a sequence of triangles $[\mathcal{T}_0, \mathcal{T}_1, \dots \mathcal{T}_N]$, we can define the value function of the sequence as the value that minimizes the cost along the optimal trajectory through the

sequence of triangles, with fixed start and endpoint:

$$Q(\boldsymbol{x}_0, [\mathcal{T}_0, \mathcal{T}_1, \dots \mathcal{T}_N], \boldsymbol{x}_f) = \min_{\boldsymbol{x},\boldsymbol{u},t} \quad \sum_{i=0}^{N} \int_{t_i}^{t_{i+1}} J(\boldsymbol{x}, \boldsymbol{u}, \tau) d\tau \tag{4a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{4b}$$

$$\boldsymbol{x}(t) \in \mathcal{T}_i \quad \forall t \in [t_i, t_{i+1}] \tag{4c}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{4d}$$

$$\boldsymbol{x}(t_{N+1}) = \boldsymbol{x}_f. \tag{4e}$$

Note, that this is the same optimization problem as in (3), but with the addition of the terminal constraint in (4e). Using this, we can get the result in Lemma 1.

*Lemma 1:* The fixed endpoint value function $Q(\cdot)$ will always be lower bounded by the free endpoint value function $V(\cdot)$:

$$Q(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N], \boldsymbol{x}_f) \geq V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N]) \tag{5}$$

*Proof:* The free endpoint value function $V(\cdot)$ where $\boldsymbol{x}_f$ is free can be expressed in terms of minimizing the fixed endpoint value function $Q(\cdot)$ as follows:

$$\begin{aligned} V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N]) &= \min_{\boldsymbol{x}_f \in \mathcal{T}_N} Q(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N], \boldsymbol{x}_f) \\ &\leq Q(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N], \boldsymbol{x}_f) \; \forall \boldsymbol{x}_f \in \mathcal{T}_N \end{aligned} \tag{6}$$

■

In order to determine the optimality of a sequence of triangles, we need to show that extending the sequence will not lower the cost of the trajectory. Using the value function definitions in (3) and (4), and the following assumption, we get the result in Lemma 2.

*Assumption 1:* The cost function $J(\cdot) \geq 0$ is a non-negative function. Meaning the integral of the cost can not decrease along the path.

*Lemma 2:* Given Assumption 1, the value function $V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N])$ is monotonically increasing with respect to the length of the sequence of triangles.

*Proof:*

$$\begin{aligned} V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N]) &= Q(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{N-1}], \boldsymbol{x}_N) + V(\boldsymbol{x}_N, [\mathcal{T}_N]) \\ &\geq V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{N-1}]) + V(\boldsymbol{x}_N, [\mathcal{T}_N]) \\ &\geq V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{N-1}]) \end{aligned}$$

■

*Definition 1 (Triangle sequence completeness):* We say that a sequence of triangles $[\mathcal{T}_0, \dots \mathcal{T}_N]$ is complete if the initial state is within the initial triangle $\boldsymbol{x}_0 \in \mathcal{T}_0$, and the final goal state is in the final triangle $\boldsymbol{x}_f \in \mathcal{T}_N$. Similarly, a sequence is incomplete if the initial state is within initial triangle $\boldsymbol{x}_0 \in \mathcal{T}_0$, and the final goal state is not within the last triangle $\boldsymbol{x}_f \notin \mathcal{T}_N$.

When searching sequences of triangles, it is useful to be able to approximate bounds on the cost to go, if the sequence is incomplete. In order to do this, we are using an admissible heuristic function $h(\boldsymbol{x}, \boldsymbol{x}_f)$ together with the optimization problem in (3) to estimate the cost to go from some state $\boldsymbol{x}$ to the terminal goal state $\boldsymbol{x}_f$. Using the heuristic, we can define the following function:

$$\begin{aligned} \underline{Q}(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}], \boldsymbol{x}_f) &= V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}]) \\ &\quad + h(\boldsymbol{x}_M, \boldsymbol{x}_f), \; \boldsymbol{x}_M = \boldsymbol{x}(t_M) \end{aligned} \tag{7}$$

which is a lower bound on possible complete sequences of triangles, that extend from an incomplete sequence. This result is summed up in Lemma 3.

*Assumption 2:* The heuristic function $h(\boldsymbol{x}_M, \boldsymbol{x}_f)$ is admissible. Hence the the heuristic will always underestimate the true cost or value function for any feasible sequence of triangles $[\mathcal{T}_M, \dots \mathcal{T}_N]$.

$$h(\boldsymbol{x}_M, \boldsymbol{x}_f) \leq Q(\boldsymbol{x}_M, [\mathcal{T}_M, \dots \mathcal{T}_N], \boldsymbol{x}_f),$$

*Lemma 3:* Given Assumption 2 and a triangle sequence $[\mathcal{T}_0, \dots \mathcal{T}_M, \dots \mathcal{T}_N]$, we have the following lower bound on the trajectory cost:

$$Q(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N], \boldsymbol{x}_f) \geq \underbrace{V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}]) + h(\boldsymbol{x}_M, \boldsymbol{x}_f)}_{:= \underline{Q}(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}], \boldsymbol{x}_f)} \tag{8}$$

where $\boldsymbol{x}_M = \boldsymbol{x}(t_M)$ is the end of the optimal free endpoint trajectory given by $V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}]$.

*Proof:*

$$\begin{aligned} Q(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N], \boldsymbol{x}_f) &= Q(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}], \boldsymbol{x}_M) \\ &\quad + Q(\boldsymbol{x}_M, [\mathcal{T}_M, \dots \mathcal{T}_N], \boldsymbol{x}_f) \\ &\geq V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}]) \\ &\quad + Q(\boldsymbol{x}_M, [\mathcal{T}_M, \dots \mathcal{T}_N], \boldsymbol{x}_f) \\ &\geq V(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}]) \\ &\quad + h(\boldsymbol{x}_M, \boldsymbol{x}_f) \\ &= \underline{Q}(\boldsymbol{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}], \boldsymbol{x}_f) \end{aligned}$$

■

Given the result from Lemma 3, where we have a lower bound $\underline{Q}(\cdot)$ for completing an incomplete sequence of triangles, we can use this to determine if completing an incomplete path will result in a complete sequence with a lower value $Q(\cdot)$, then some other completes sequence. This is summed up in Theorem 1.

*Theorem 1:* Given a complete sequence of triangles $\mathcal{S}^* = [\mathcal{T}_0, \dots \mathcal{T}_N]$, and an incomplete sequence $\mathcal{S}'$ satisfying:

$$Q(\boldsymbol{x}_0, \mathcal{S}^*, \boldsymbol{x}_f) \leq \underline{Q}(\boldsymbol{x}_0, \mathcal{S}', \boldsymbol{x}_f), \tag{9}$$

Then completing the incomplete sequence $\mathcal{S}'$ can not result in a trajectory with a lower value $Q(\cdot)$ then the sequence $\mathcal{S}^*$.

*Proof:* From Lemma 3, we have that extending any incomplete sequence $\mathcal{S}'$ to a complete sequence $\mathcal{S}$ will result in a higher cost, i.e:

$$\underline{Q}(\boldsymbol{x}_0, \mathcal{S}', \boldsymbol{x}_f) \leq Q(\boldsymbol{x}_0, \mathcal{S}, \boldsymbol{x}_f).$$

Given the condition in (9), we get the following result:

$$\begin{aligned} Q(\boldsymbol{x}_0, \mathcal{S}^*, \boldsymbol{x}_f) \leq \underline{Q}(\boldsymbol{x}_0, \mathcal{S}', \boldsymbol{x}_f) \quad &\Rightarrow \\ Q(\boldsymbol{x}_0, \mathcal{S}^*, \boldsymbol{x}_f) \leq Q(\boldsymbol{x}_0, \mathcal{S}, \boldsymbol{x}_f). & \end{aligned}$$

This means that all sequences $\mathcal{S}$ that can result from the incomplete sequences $\mathcal{S}'$ will have higher cost then the optimal sequence $\mathcal{S}^*$ if (9) holds. ■

Using the refined trajectory cost $V(\cdot)$, heuristic admissible cost $h(\cdot)$ and the search termination conditions given Theorem 1, we can derive the complete trajectory planning Algorithm 1. Where at each iteration, the trajectory is expanded into the triangle that minimizes the cost lower bound $\underline{Q}(\cdot)$. Until a complete sequence of triangles $\mathcal{S}^*$ is found, for which the termination condition in (9) is true for all sequences $\mathcal{S}'$, in the list of sequences to be searched (*open_list*). By ordering the (*open_list*) by the cost lower bound $\underline{Q}(\cdot)$, this reduces the termination to checking the termination condition (9) on only the first element in the (*open_list*). From Theorem 2 we can show that the proposed algorithm will find the optimal sequence of triangles, and hence the globally optimal trajectory, under the assumption that the resulting optimization problem is convex. This is the case if the dynamical system results in convex constraints, as the spatial constraints will be convex due to the triangulation.

*Lemma 4:* In Algorithm 1, the list of sequences to be searched (*open_list*) will always contain a sub-sequence $\mathcal{S}'$ of any possible complete path $\mathcal{S}$

*Proof:* Algorithm 1, changes the *open_list* by iterative removing incomplete sequences, and adding all feasible sequences that can be extended by one triangle from the sequence that is removed. Since any possible complete path must be extended from the sequence only containing the initial triangle $\mathcal{T}_0$. Then the list of sequences to be searched (*open_list*) will always contain a sub-sequence of any possible complete path. ■

*Theorem 2:* Algorithm 1 will find the optimal sequence of triangles $\mathcal{S}^*$, and hence the globally optimal trajectory.

*Proof:* Given that Algorithm 1 terminates with the optimal sequence $\mathcal{S}^*$. If we assume there exists a better sequence $\tilde{\mathcal{S}}^*$. such that:

$$Q(\boldsymbol{x}_0, \tilde{\mathcal{S}}^*, \boldsymbol{x}_f) < Q(\boldsymbol{x}_0, \mathcal{S}^*, \boldsymbol{x}_f)$$

Then from Lemma 4, a sub-sequence $\tilde{\mathcal{S}}'$ of $\tilde{\mathcal{S}}^*$ must exist in the list of possible sequences to be extended (*open_list*). Given the result in Lemma 3 we get that:

$$\underline{Q}(\boldsymbol{x}_0, \tilde{\mathcal{S}}', \boldsymbol{x}_f) \leq Q(\boldsymbol{x}_0, \tilde{\mathcal{S}}^*, \boldsymbol{x}_f) < Q(\boldsymbol{x}_0, \mathcal{S}^*, \boldsymbol{x}_f).$$

This contradicts the termination condition in (9), and hence no sequence $\tilde{\mathcal{S}}^*$ that is better then $\mathcal{S}^*$ can exist. ■

### E. Implementation considerations

In order to implement the optimization problem given in (3) and (4), we need to formulate the constraint in (3c) and (4c) as a linear inequality constraint. The most straightforward way of doing this is to use the half-space representation of the triangle. Given a 2D triangle $\mathcal{T}_i$ with vertices $\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \boldsymbol{v}_{i,3}$, as illustrated in Figure 3, the half-space representation of a triangle gives a set of linear inequality constraints on the form:

$$\boldsymbol{A}_i \boldsymbol{p} \leq \boldsymbol{b}_i.$$

Where $\boldsymbol{A}_i \in \mathbb{R}^{3\times 2}$ and $\boldsymbol{b}_i \in \mathbb{R}^{3\times 1}$ is the matrix and vector making up the halfspace, and $\boldsymbol{p} = [x, y]^\top$ is a position. Using this, we can check if a position $\boldsymbol{p}$ lies within the triangle $\mathcal{T}_i$, as follows:

$$\boldsymbol{A}_i \boldsymbol{p} \leq \boldsymbol{b}_i \quad \Leftrightarrow \quad \boldsymbol{p} \in \mathcal{T}_i. \tag{10}$$

**Algorithm 1** Optimal trajectory planning. Note that evaluating $Q(\cdot)$ and $\underline{Q}(\cdot)$ involves solving the optimization problem in (3) and (4) using numerical optimization.

**Require:** Adjacency graph of triangulation, initial state $\boldsymbol{x}_0$, and goal state $\boldsymbol{x}_f$.
$\mathcal{S}^* = [\,]$
$\mathcal{S} = [\mathcal{T}_0]$ where $\boldsymbol{x}_0 \in \mathcal{T}_0$
$open\_list = \{\mathcal{S}\}$
**while** $open\_list$ is not empty **do**
  $\mathcal{S}$ = pop sequence from $open\_list$ with smallest $\underline{Q}(\boldsymbol{x}_0, \mathcal{S}, \boldsymbol{x}_f)$
  **if** $\mathcal{S}^*$ is not empty, and $\underline{Q}(\boldsymbol{x}_0, \mathcal{S}, \boldsymbol{x}_f) \geq Q(\boldsymbol{x}_0, \mathcal{S}^*, \boldsymbol{x}_f)$ **then**
    **return** Optimal triangle sequence $\mathcal{S}^*$
  **end if**
  **for** Triangle $\mathcal{T}_n$ in $neighbours(\mathcal{S})$ **do**
    $\mathcal{S}_n = \text{extend}(\mathcal{S}, \mathcal{T}_n)$
    **if** $\boldsymbol{x}_f \in \mathcal{T}_n$ **then**
      **if** $Q(\boldsymbol{x}_0, \mathcal{S}_n, \boldsymbol{x}_f) < Q(\boldsymbol{x}_0, \mathcal{S}^*, \boldsymbol{x}_f)$ **then**
        $\mathcal{S}^* = \mathcal{S}_n$
      **end if**
    **else**
      append $\mathcal{S}_n$ to $open\_list$
    **end if**
  **end for**
**end while**

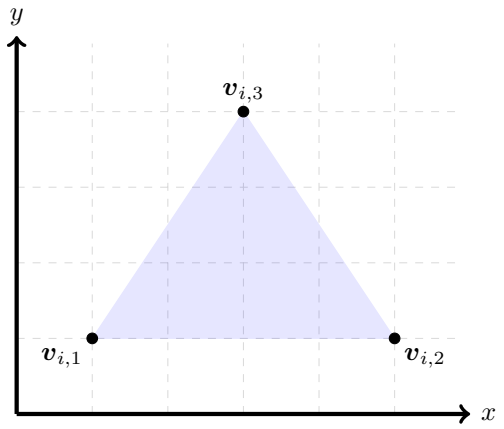



Fig. 3: Triangle $\mathcal{T}_i$, with vertices $\boldsymbol{v}_{i,1}$, $\boldsymbol{v}_{i,2}$, $\boldsymbol{v}_{i,3}$

The matrix $\boldsymbol{A}_i$, and vector $\boldsymbol{b}_i$ can be computed using the triangle vertices $\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \boldsymbol{v}_{i,3}$ as follows:

$$\begin{aligned} \boldsymbol{A}_i &= \begin{bmatrix} (\boldsymbol{v}_{i,2} - \boldsymbol{v}_{i,1})^\top \boldsymbol{R}^\top \\ (\boldsymbol{v}_{i,3} - \boldsymbol{v}_{i,2})^\top \boldsymbol{R}^\top \\ (\boldsymbol{v}_{i,1} - \boldsymbol{v}_{i,3})^\top \boldsymbol{R}^\top \end{bmatrix} \\ \boldsymbol{b}_i &= \begin{bmatrix} (\boldsymbol{v}_{i,2} - \boldsymbol{v}_{i,1})^\top \boldsymbol{R}^\top \boldsymbol{v}_{i,1} \\ (\boldsymbol{v}_{i,3} - \boldsymbol{v}_{i,2})^\top \boldsymbol{R}^\top \boldsymbol{v}_{i,2} \\ (\boldsymbol{v}_{i,1} - \boldsymbol{v}_{i,3})^\top \boldsymbol{R}^\top \boldsymbol{v}_{i,3} \end{bmatrix} \end{aligned} \tag{11}$$

Where the matrix $\boldsymbol{R}$ is given as the $\pm 90°$ rotation matrix, when the triangle vertices are given in a in a clockwise/counter clockwise direction. In the example in Figure 3, the vertices are given in a counter clockwise direction, giving the following

rotation matrix:

$$\boldsymbol{R} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

While the above linear inequality can be used to ensure the different path segments stay within the desired triangle, we propose a slight modification to this approach. The modification involves using a local triangle-centered coordinate system instead of a global coordinate system for optimizing the position within the triangle. Defining the following objects:

$$\begin{aligned} \boldsymbol{C}_i &= [\boldsymbol{v}_{i,2} - \boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,3} - \boldsymbol{v}_{i,2}] \\ \boldsymbol{d}_i &= \boldsymbol{v}_{i,1}, \end{aligned} \tag{12}$$

we define the transformation between the position $\boldsymbol{p} = [x, y]^\top$ in the global coordinate system, and the position $\boldsymbol{p}' = [p'_1, p'_2]^\top$ in the local triangle coordinate system as follows:

$$\boldsymbol{p} = \boldsymbol{C}_i \boldsymbol{p}' + \boldsymbol{d}_i. \tag{13}$$

Using the triangle transformation in (13), the position $\boldsymbol{p}$ is constrained within the triangle when the following inequality constraints are satisfied:

$$\begin{aligned} 0 \leq \boldsymbol{p}' &\leq 1 \\ p'_1 - p'_2 &\leq 0, \end{aligned} \tag{14}$$

which can be implemented directly into the optimization problem as the triangle constraints in equation (3c) and (4c). The reason for using this coordinate transformation is to help normalize the variables in the optimization problem as well as simplify the triangle constraints. This helps improve the conditioning of the optimization problem, and gives better performance when solving the problem.

Another consideration when solving (3) and (4), is how to perform the integration of the cost function (3a) and (4a), and system dynamics (3b) and (4b). In order to do this we propose using a multiple shooting collocation based scheme [39], for which the trajectory in each triangle is approximated by a polynomial of degree $d$. This results in an optimization problem, where the objective is to find a spline where each triangle contains a polynomial representing the trajectory through the triangle (Figure 1d), the trajectories are then constrained to being connected between neighbouring triangles (Figure 1e), while at the same time satisfy the system dynamics. It is worth noting that the trajectory within each triangle will differ in length due to the size and shape of the triangle. This means the a free time variable must be used for each triangle in order to ensure the trajectory is constrained within the triangle.

In the graph search phase, some additional assumptions were made, in order to prune and reduce the search space.

*Assumption 3:* The optimal path will only pass through any given triangle $\mathcal{T}$ once.

Assumption 3, allows us to not extend a sequence of triangles into a given triangle if it already appears in the sequence. This results in a significantly smaller search space, when searching for the optimal triangle sequence. It should be noted that Assumption 3 is not strictly necessary, as the proposed method will in theory work without it. It does however significantly reduce the search space, and helps make the method computationally feasible. The downside of this assumption, is that the proposed planner will not allow for maneuvers which reenter triangles, which may be optimal for certain classes of problems.

*Assumption 4:* If two initial starting points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{T}$ are sufficiently close:

$$||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2 \leq \epsilon.$$

Then the optimal sequences of triangles $\mathcal{S}^*$ to the goal will be the same for both trajectories, and the difference between values of the trajectories is bounded.

$$||Q(\boldsymbol{x}_1, \mathcal{S}^*, \boldsymbol{x}_f) - Q(\boldsymbol{x}_2, \mathcal{S}^*, \boldsymbol{x}_f)|| \leq \delta$$

Given two different triangle sequences $\mathcal{S}_1$ and $\mathcal{S}_2$, that both end in the same triangle $\mathcal{T}$, and the same endpoints $\boldsymbol{x}_1 = \boldsymbol{x}_2, \quad \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{T}$, where:

$$V(\boldsymbol{x}_0, \mathcal{S}_1) \leq V(\boldsymbol{x}_0, \mathcal{S}_2),$$

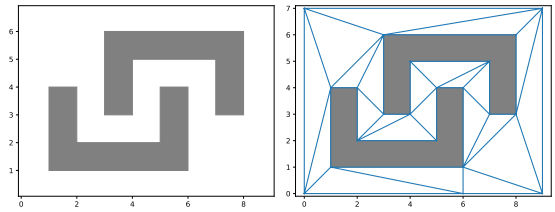
we only need to continue the search from the sequence $\mathcal{S}_1$, and hence can prune the sequence $\mathcal{S}_2$. Using Assumption 4, we can extend the above argument to say that we can prune sequences if the states are sufficiently close. Unfortunately, computing the exact bounds would require completing the trajectory, which defeats the purpose of pruning. In stead we use the following heuristic for evaluating if two endpoints $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are sufficiently close:

$$(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top \boldsymbol{W} (\boldsymbol{x}_1 - \boldsymbol{x}_2) \leq \epsilon, \quad \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{T}$$

where $\boldsymbol{W}$ is a positive definite weighting matrix, and $\epsilon$ is a sufficiently small threshold. This is a relaxation of the exact condition for pruning, where $\boldsymbol{x}_1 = \boldsymbol{x}_2, \quad \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{T}$, and where the conditions are exactly the same in the limit as $\epsilon \to 0$. It should be noted that pruning potential sequences is not strictly necessary. It is however added in order to further reduce the search space, and hence improve the computational complexity.

When running Algorithm 1, there may be certain triangle sequences for which no feasible trajectory can be found due to the constraints imposed by the vehicle dynamics, as well as additional state and input constraints. When an infeasible triangle sequence is found, no feasible path through the given sequence exists, and the sequence is discarded. In practise, numerical optimization tools are used to find the optimal trajectories and detect infeasible solutions. In certain situations, the numerical optimization tools may not return a feasible solution even though a feasible solution theoretically does exist. This can typically be mitigated using good initial guesses for the trajectory. For the proposed planner, the optimal trajectory from the previous triangle sequence can be used as one such initial guess. In addition to improving feasibility, using such an initial guess will typically also improve the computation time, as a good initial guess will result in fewer iterations when solving the optimization problem in (3) and (4).

Given algorithm 1, we can note that it is possible to paralellize the exploration of new triangle sequences. This is possible, as the exploration of possible sequences is not dependant on other sequences, however it requires some extra considerations in the termination criteria. For our implementation, this property was exploited in order to explore multiple



(a) Constraints, without triangulation. (b) Constraints, with triangulation.

Fig. 4: Polygonal spatial constraints, and the resulting CDT

sequences in parallel. It should be noted that if an exact heuristic function is known, the paralellization will not give a speedup, as the optimal sequence of triangles will always be the first to be explored. If however a poor heuristic function is used, parallelization will typically give a speedup, as it allows for multiple triangle sequences to be explored simultaneously.

## III. Examples

In order to validate the method, we test it on a simple kinematic car model in in a confined environment, and compare to a Rapidly-exploring Random Tree based approach. To further prove the versatility of the method we also show it on a test scenario in trajectory planning for marine vessels in the Trondheim fjord, for which we use it to plan trajectories that minimize time, distance as well as energy.

### A. Trajectory planning for a simple kinematic car model

*1) Simple kinematic car model:* In order to verify the proposed method we will in this section show how it can be applied to planning distance optimal paths for a simple kinematic car model on the form:

$$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix}\cos(\psi)v\\ \sin(\psi)v\\ r\end{bmatrix}}_{f(\boldsymbol{x},\boldsymbol{u})} \tag{15}$$

where $x$, $y$ is the position, $\psi$ is the heading, $v$ is the velocity, and $r$ is the turning rate. Using the constant speed $v = 1$, we are left with an under actuated system where the turning rate is the control variable $\boldsymbol{u} = r$. This type of model is often used robotics and control theory when planning paths for wheeled robots, airplanes and underwater vehicles. As the model offers a simple geometric approximation of the maneuvering capabilities of these types vehicles.

*2) Spatial constraints:* In order to validate the proposed method, the simple set of spatial constraints, seen in Figure 4a, were devised. Given the polygonal representation, the CDT was computed, giving the triangulation in Figure 4b

*3) Optimization objective:* The objective for the optimization problem, is to find the shortest path between two points. The instantaneous is then given by the path integral as follows:

$$\begin{aligned} J(\boldsymbol{x},\boldsymbol{u},t) &= \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \\ &= \sqrt{(\cos(\psi)v)^2 + (\sin(\psi)v)^2} \\ &= |v| \\ &= 1. \end{aligned} \tag{16}$$

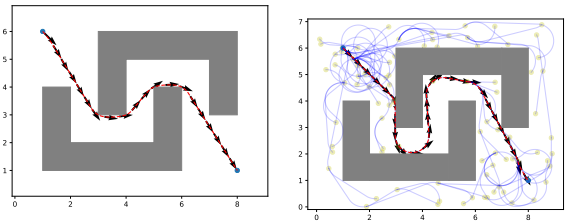
It should be noted, that given a maximum turning rate, and the vehicle traveling at constant speed, the distance optimal path from one point to an other can be shown to be a Dubins path [21], which consists of straight lines and circles segments of maximum curvature.

*4) Results:* As the optimal path is known to be a Dubins path, a Dubins based RRT method [40] is used for comparison, as RRT based methods are the most commonly used approaches for motion planning for robotic applications when faced with spatial constraint. Given the spatial constraint in Figure 4, we get the resulting planned path in Figure 5. From the results we see that one of the major flaws of the Dubins based RRT method is that it performance is highly dependant on the randomly sampled nodes, which are used to select way points. For RRT, finding a feasible path is fairly quick, and it is possible to continue to optimize the path by generating new nodes. Further optimizing the path can often be very time consuming, as the RRT path can only guarantee converge to the optimal path as the number of sampled nodes approaches infinity [41]. For our proposed approach however, if a feasible path is found it is guaranteed to be optimal. This is verified in the results, where we can observe that our approach generates a path which is very similar to a Dubins path, and finds the shortest path that gets close to, but does not intersect the spatial constraints. In real time planning, where finding a feasible path in a short period of time is important, sampling based methods such as RRT will still be the better choice. However, if the goal is to find the optimal path in a finite amount of time, our proposed approach will be the better option, with our implementation of the optimization based planner using an average of 3.24 seconds for planning the optimal trajectory in Figure 5a.

### B. Trajectory planning for an autonomous surface vessel

In the field of marine robotics, motion planning is an important problem, which has seen a lot of interest. Given the complex vessel dynamics, as well as complex non-convex spatial constraints, the motion planning problem becomes very difficult. Because of this, most existing solutions heavily rely on simplifying the problem, this however results in loss of accuracy and optimality of the final solution. In this example we will show how our proposed planning algorithm can be used for optimal trajectory planning for an Autonomous Surface Vessel (ASV) in the Trondheim harbour.

*1) Vessel model:* As a model for the trajectory optimization, we will use a vessel model, where we assume the vessel moves on the ocean surface in a relatively large range of possible velocities. In addition to this, we assume that the effects of the roll and pitch motions of the vessel are negligible, and hence



(a) Our approach. The red dashed line shows the final optimized path, while the arrows show the direction of travel, when moving between the start and end point marked with blue dots.

(b) Dubins based RRT. The red dashed line is the final path, yellow dots show sampled nodes, while the blue lines show explored dubins paths between the sampled nodes.

Fig. 5: Results for path generated by our proposed approach and Dubins based RRT.

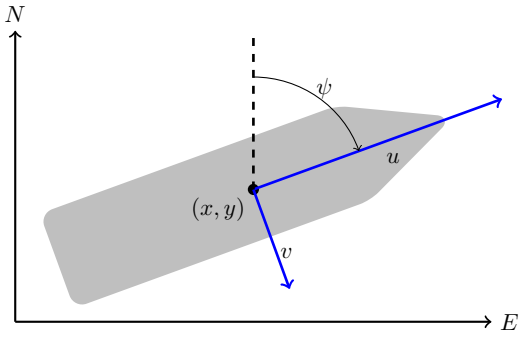



Fig. 6: 3-DOF vessel centered at $(x, y)$, with surge velocity $u$, sway velocity $v$, heading $\psi$ in a North-East-Down (NED) reference frame.

have little impact on the surge, sway, and yaw of the vessel. The mathematical model used to describe the system can then be kept reasonably simple by limiting it to the planar position and orientation of the vessel. The motion of a surface vessel can be represented by the pose vector $\boldsymbol{\eta} = [x, y, z_r, z_i]^\top \in \mathbb{R}^4$, and the velocity vector $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$. Here, $(x, y)$ describe the Cartesian position in the earth-fixed reference frame, $(z_r, z_i)$ is a complex number of unit length $|z| = |z_r + i \cdot z_i| = 1$ describing the vessel orientation, where $\psi = \text{atan2}(z_i, z_r)$ is yaw angle, $(u, v)$ is the body fixed linear velocities, and $r$ is the yaw rate, an illustration is given in Figure 6. Using the notation in [42] we can describe a 3-DOF vessel model as follows

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{17}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \tag{18}$$

where $\boldsymbol{M}, \boldsymbol{C}(\boldsymbol{\nu}), \boldsymbol{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, $\boldsymbol{\tau} \in \mathbb{R}^3$ and $\boldsymbol{J}(\boldsymbol{\eta})$ are the inertia matrix, coriolis matrix, dampening matrix, control input vector, and transformation matrix respectively. The transformation matrix $\boldsymbol{J}(\boldsymbol{\eta})$ is given by

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{bmatrix} z_r & -z_i & 0 \\ z_i & z_r & 0 \\ 0 & 0 & -z_i \\ 0 & 0 & z_r \end{bmatrix}, \tag{19}$$

(a) Map, without triangulation.

(b) Map, with triangulation.

Fig. 7: Map of the Trondheim fjord, based on polygons representing land masses.

and is the transformation from the body frame to the earth-fixed reference frame. Using the unit complex numbers in stead of a heading angle allows the the dynamics to avoid the angle wraparound problem, which avoids local optima when performing trajectory optimization. For the model dynamics $\boldsymbol{M}, \boldsymbol{C}(\boldsymbol{\nu}), \boldsymbol{D}(\boldsymbol{\nu})$, parameters for a simplified model of the milliAmpere experimental platform was used, where:

$$\boldsymbol{M} = \begin{bmatrix} 2138 & 0 & 0 \\ 0 & 2528 & 0 \\ 0 & 0 & 3942 \end{bmatrix} \tag{20}$$

$$\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \begin{bmatrix} 10.3u + 114.6|u|u - 2528vr \\ 13.0v + 200.8|v|v + 2138ur \\ 201.0r + 424.1|r|r + 390uv \end{bmatrix}. \tag{21}$$

For the thrust configuration, one rotatable azimuth thruster is assumed, giving the following thrust vector:

$$\boldsymbol{\tau} = \begin{bmatrix} u_1 \cos(u_2) \\ u_1 \sin(u_2) \\ -2u_1 \sin(u_2) \end{bmatrix}, \tag{22}$$

Where $0 \leq u_1 \leq 400$ is the thruster force, and $-45° \leq u_2 \leq 45°$ the thruster angle.

*2) Spatial constraints:* Using a map where landmasses are represented by polygons, a CDT is created, where all edges of the polygons are treated as constraints. Doing this ensures that the resulting triangulation has triangles that do not intersect land. The resulting triangulation mesh is shown in Figure 7. While the whole map of the Trondheim fjord is used, for the example, only a small portion of the map was relevant as the start and goal positions were selected within the Trondheim harbour.

*3) Optimization objective:* Depending on the use-case, any optimization objective satisfying Assumption 1 can be selected. In this paper we will show three commonly used objectives, namely time minimization, distance minimization, and energy minimization.

*Minimum time:* In terms of instantaneous cost, the time minimization is the simplest optimization objective. where:

$$J(\boldsymbol{x}, \boldsymbol{u}, t) = 1. \tag{23}$$

This gives the path integral optimization problem as follows:

$$\int_{t_0}^{t_N} 1 \, dt. \tag{24}$$

Minimizing the above expression then equates to minimizing the total time, $t_N - t_0$, of the the trajectory, with boundary conditions given by the initial and final state.

For the heuristic function of the minimum time, we chose the time taken traveling in a straight line from the given state $\boldsymbol{x}_N$ to the desired terminal state $\boldsymbol{x}_f$, at the maximum vessel speed $U_{\max}$. Giving the following heuristic function:

$$h(\boldsymbol{x}_N, \boldsymbol{x}_f) = \frac{\sqrt{(x_N - x_f)^2 + (y_N - y_f)^2}}{U_{\max}}. \tag{25}$$

Intuitively, we can see that this is an admissible heuristic, as it represents the time of traveling the shortest possible path, at the highest speed possible, hence it will always underestimate the time of a feasible trajectory.

*Minimum distance:* In terms of minimizing distance, we can observe that the instantaneous cost of a trajectory given by the north, and east coordinates $x(t)$ and $y(t)$ respectively, is given as the instantaneous arc length:

$$\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}. \tag{26}$$

From the kinematics we note that the square of the instantaneous cost can be rewritten as:

$$\begin{aligned} \dot{x}^2 + \dot{y}^2 =& \cos(\psi)^2 u^2 + \sin(\psi)^2 v^2 - \cos(\psi)\sin(\psi)uv \\ &+ \sin(\psi)^2 u^2 + \cos(\psi)^2 v^2 + \cos(\psi)\sin(\psi)uv \\ =& (\cos(\psi)^2 + \sin(\psi)^2)(u^2 + v^2) \\ =& u^2 + v^2, \end{aligned} \tag{27}$$

giving the following instantaneous cost.

$$J(\boldsymbol{x}, \boldsymbol{u}, t) = \sqrt{u^2 + v^2} \tag{28}$$

This gives the path integral optimization problem as follows:

$$\int_{t_0}^{t_N} \sqrt{u^2 + v^2}\, dt. \tag{29}$$

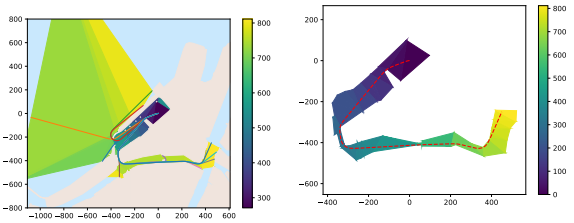
Theoretically optimizing the above problem should give the shortest path, however for most optimization algorithms, the objective must be smooth and continuously differentiable, which is not the case when the square root is used. In order to ensure the function is continuously differentiable, a small positive number $\epsilon > 0$ is added, giving the following smooth approximation of the path integral:

$$\int_{t_0}^{t_N} \sqrt{u^2 + v^2 + \epsilon}\, dt. \tag{30}$$

For the heuristic function of the minimum distance, we simply chose the euclidean distance from the given state $\boldsymbol{x}_N$ to the desired terminal state $\boldsymbol{x}_f$.

$$h(\boldsymbol{x}_N, \boldsymbol{x}_f) = \sqrt{(x_N - x_f)^2 + (y_N - y_f)^2} \tag{31}$$

Intuitively, we can see that this is an admissible heuristic, as represents the straight line path, which is the shortest possible path between two points. This means that it will always underestimate the length of a feasible trajectory.



(a) Space of searched triangles (colored by the value function lower bound $\underline{Q}(\cdot)$), with time optimal trajectories to the fringes given as solid lines.

(b) Sequence of triangles (colored by the free endpoint value function $V(\cdot)$) for the minimum time trajectory, within which the trajectory refinement is performed.

Fig. 8: Search space and triangulation for minimum time trajectory.

*Minimum energy:* In many problems, it is often useful to minimize the energy usage. In the case of marine vessels, minimizing energy usage, equates to better fuel efficiency, and less pollution. For moving objects, the quantity of work over time (power) is integrated along the trajectory of the point of application of the force. This gives the instantaneous power as the scalar product of the force/torque and the linear/angular velocity.

$$\boldsymbol{\tau}^\top \boldsymbol{\nu} \tag{32}$$

In general, power regeneration and recapture is not possible for marine vessels, In order to account for this we in stead use the absolute instantaneous power, giving the following instantaneous cost:

$$J(\boldsymbol{x}, \boldsymbol{u}, t) = |X \cdot u| + |Y \cdot v| + |N \cdot r|, \tag{33}$$

where the thrust vector is given as $\boldsymbol{\tau} = [X, Y, N]^\top$, and velocity vector is given as $\nu = [u, v, r]$. This gives the path integral optimization problem as follows:

$$\int_{t_0}^{t_N} |X \cdot u| + |Y \cdot v| + |N \cdot r|\, dt. \tag{34}$$

Similarly to the minimum distance formulation, the absolute value is none smooth and the derivative not defined at 0, in order to avoid this problem, we again use an approximation of the absolute value giving the following path integral to be optimized.

$$\int_{t_0}^{t_N} \sqrt{(X \cdot u)^2 + \epsilon} + \sqrt{(Y \cdot v)^2 + \epsilon} + \sqrt{(N \cdot r)^2 + \epsilon}\, dt. \tag{35}$$

For the heuristic function of the minimum energy, it is difficult to find a good estimate for the cost to go from a given state $\boldsymbol{x}_N$ to the desired terminal state $\boldsymbol{x}_f$. Hence the heuristic:

$$h(\boldsymbol{x}_N, \boldsymbol{x}_f) = 0 \tag{36}$$

is chosen. This is in general a poor estimate of the cost to go, and will result in a larger number of triangles being explored, but it is an admissible heuristic and hence satisfies Assumption 2.

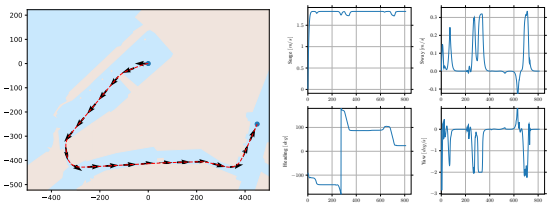


Fig. 9: Minimum time path

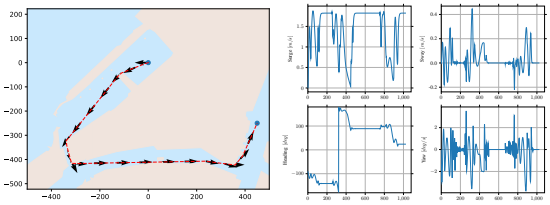


Fig. 10: Minimum distance path

*4) Results:* To visualize the proposed algorithm during the search phase, the value functions are shown in Figure 8. For the three different optimization objectives, the resulting trajectories from the trajectory planner are given in Figures 9, 10 and 11 for the time, distance and energy minimization problems respectively. For the energy minimization problem, it is important to note that the any actuation of the control surfaces will result in energy being used, hence the optimal action would be to not move. In order to fix this, a terminal constraint was added on the time, in order to ensure the trajectory would be complete within 1200 seconds.

From the performance measure comparison in Table I, we can see that the different optimization objectives perform as expected, as they each minimize their respective objectives. For the minimum time objective, we can see that the speed in the surge direction is close to the maximum for most of the duration of the trajectory, this is what results in the minimum time trajectory, but comes at the cost of a slightly longer trajectory in terms of distance, and a significantly larger energy consumption. For the minimum distance trajectory we see a more erratic behaviour, especially in the surge direction. This pattern of speeding up and slowing down, is what allows the vessel to take tight corners, and hence minimize the distance, however due to the trajectory dynamics, the resulting distance is only slightly shorter than that of the other two optimization objectives. For the minimum energy trajectory, the behaviour is similar to that of the minimum time objective, with the main difference being a lower surge speed. This behaviour is due to the nonlinear drag, which makes lower speeds more energy efficient.

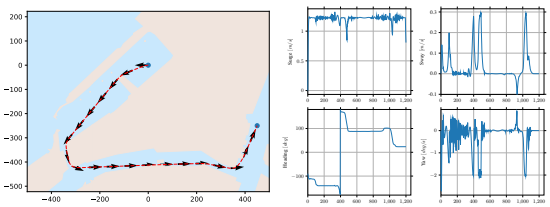


Fig. 11: Minimum energy path

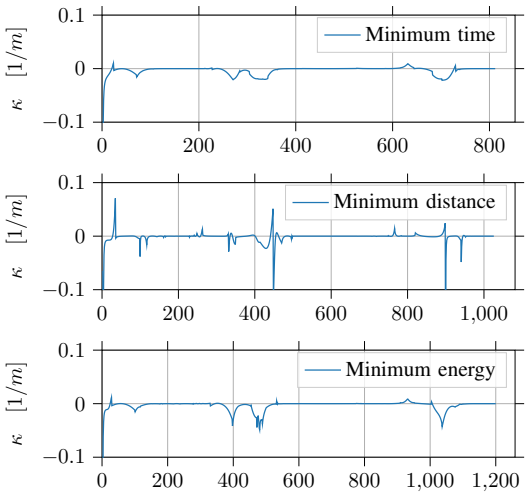



Fig. 12: Trajectory curvature resulting from the different optimization objectives.

A useful tool for evaluating a the feasibility of a trajectory, is the trajectory curvature $\kappa$.

$$\kappa = \frac{\dot{x} \cdot \ddot{y} - \dot{y} \cdot \ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{37}$$

One of the reasons for the curvature being used as a way of evaluating trajectory feasibility, is that most vessels have a limit on the maximum possible path curvature. This has lead to the widespread use of Dubins paths [21] which consist of straight line segments and circle arcs with maximum curvature, giving path with piecewise constant curvature. These paths have been shown to be the shortest path for a vehicle that only travels forward, and has a constraint on max curvature. The Dubins path however does not consider the underlying system dynamics, hence a dubins path is no longer optimal once the dynamics are considered. This is illustrated in Figure 12 where the curavature is continuous, similar to [8]. From the curvature results it is worth noting the difference in curvature between the different optimization objectives. For the minimum time objective a higher speed is desired, hence the curvature is small allowing for taking turns at higher speeds. For the minimum distance trajectory, we can observe spikes of very high curvature, which is what we expect as the shortest path will consist only of straight line segments. For the minimum energy trajectory, we see similar results to that of the minimum time path, however the peak curvature is slightly higher, as a result of the velocities being lower.

For the implementation of Algorithm 1 used to solve the trajectory planning problem, we achieved the algorithm running time given in Table II. The timing shows the results for running the algorithm sequentially, as well as the performance when running the algorithm with 4 and 8 parallel workers. In theory, increasing the number of workers, should not lead to slower running times. In practise however, there is a overhead associated with each additional worker. This is reflected in the results for the minimum time and minimum distance objectives, where the sequential approach outperforms multiple workers. For the minimum energy approach however, we see

| Trajectory | Time $[s]$ | Distance $[m]$ | Energy $[kJ]$ |
|---|---|---|---|
| Minimum Time | **811.81** | 1460.97 | 584.82 |
| Minimum Distance | 1025.82 | **1450.58** | 484.70 |
| Minimum Energy | 1200.00 | 1456.20 | **269.96** |

TABLE I: Performance measure

| | Sequential | 4 workers | 8 workers |
|---|---|---|---|
| Minimum Time | **4min 28s** | 6min 1s | 7min 19s |
| Minimum Distance | **6min 40s** | 6min 49s | 8min 36s |
| Minimum Energy | 18min 36s | 15min 4s | **13min 5s** |

TABLE II: Time required for solving the different problems using the sequential approach, as well as 4 and 8 parallel workers.

that increasing the number of workers improves the solution time. This is due to the poor choice of heuristic function, resulting in having to search a larger part of the search space, and hence the ability to evaluate multiple sequences simultaneously, outweighs the overhead of having multiple workers. It should be noted that the timing result in Table II, will vary greatly with implementation and hardware, and a more optimized implementation is likely to significantly improve the running time.

## IV. Conclusion

In this paper, we have proposed a method for planning and optimizing trajectories in an environment with static polygonal obstacles, and where the trajectories must be feasible with respect to model dynamics. Under some mild assumptions, we show that the method is able to plan globally optimal trajectories, even when faced with highly non-convex obstacles. The proposed method does however have some drawbacks. The main drawback being computational requirements, which is due to each iteration of the search phase requiring the solution of a numerical optimization problem. As well as the number of decision variables for the optimization problems increasing linearly with the number of triangles the trajectory passes through. Another important limitation of the proposed method is that the dynamics of the system is approximated by a single polynomial within each triangle, this can cause problems for large triangles and complex dynamical models, where the polynomial is not sufficiently rich to accurately capture the dynamics. Despite these limitations, the proposed method shows great promise based on simulation results. Offering great flexibility both in terms of environment complexity, model complexity, as well as optimization objective.

For future work, one of the main concerns would be to improve the computational efficiency. Some potential methods for doing so, include fixing the trajectory after a certain number of triangles in order to reduce the number of decision variables at later stages, or developing better heuristics to reduce or limit the search space. Work can also be done on how to best select a numerical integration scheme to better balance accuracy, flexibility, and computational efficiency. Similarly, methods for further decomposing the triangulation may also be used to improve accuracy, especially in large triangles, or when performing complex maneuvers. It may also be interesting to add additional environmental disturbances to the problems. This would be especially useful in the case of vessel motion planning, where wind and current may greatly impact the performance.

## References


[1] A. Wolek and C. A. Woolsey, "Model-based path planning," in *Sensing and Control for Autonomous Vehicles*. Springer, 2017, pp. 183–206.

[2] G. Bitar, M. Breivik, and A. M. Lekkas, "Energy-optimized path planning for autonomous ferries," *IFAC-PapersOnLine*, vol. 51, no. 29, pp. 389–394, 2018.

[3] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[4] M. Kallmann, "Path planning in triangulations," in *Proceedings of the IJCAI workshop on reasoning, representation, and learning in computer games*, 2005, pp. 49–54.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[6] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, "A voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Engineering Practice*, vol. 61, pp. 41–54, 2017.

[7] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE transactions on systems, man, and cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[8] A. M. Lekkas, A. R. Dahl, M. Breivik, and T. I. Fossen, "Continuous-curvature path generation using fermat's spiral," *Modeling, Identification and Control*, vol. 34, no. 4, p. 183, 2013.

[9] P. Jacobs and J. Canny, "Planning smooth paths for mobile robots," in *Nonholonomic Motion Planning*. Springer, 1993, pp. 271–342.

[10] S. Fleury, P. Soueres, J.-P. Laumond, and R. Chatila, "Primitives for smoothing mobile robot trajectories," *IEEE transactions on robotics and automation*, vol. 11, no. 3, pp. 441–448, 1995.

[11] K. Judd and T. McLain, "Spline based path planning for unmanned air vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2001, p. 4238.

[12] J. Pan, L. Zhang, D. Manocha, and U. Hill, "Collision-free and curvature-continuous path smoothing in cluttered environments," *Robotics: Science and Systems VII*, vol. 17, p. 233, 2012.

[13] A. M. Lekkas and T. I. Fossen, "Integral los path following for curved paths based on a monotone cubic hermite spline parametrization," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 6, pp. 2287–2301, 2014.

[14] C. L. Bottasso, D. Leonello, and B. Savini, "Path planning for autonomous vehicles by trajectory smoothing using motion primitives," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1152–1168, 2008.

[15] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[16] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[17] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[18] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[19] S. Carpin and G. Pillonetto, "Motion planning using adaptive random walks," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 129–136, 2005.

[20] ——, "Merging the adaptive random walks planner with the randomized potential field planner," in *Proceedings of the Fifth International Workshop on Robot Motion and Control, 2005. RoMoCo'05.* IEEE, 2005, pp. 151–156.

[21] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[22] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific journal of mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[23] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[24] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4. Citeseer, 1995, pp. 1942–1948.
[25] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, vol. 5. IEEE, 1997, pp. 4104–4108.
[26] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.
[27] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.
[28] F. Fahroo and I. M. Ross, "Direct trajectory optimization by a chebyshev pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 160–166, 2002.
[29] O. Ljungqvist, N. Evestedt, M. Cirillo, D. Axehill, and O. Holmer, "Lattice-based motion planning for a general 2-trailer system," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 819–824.
[30] K. Bergman, O. Ljungqvist, J. Linder, and D. Axehill, "An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 5283–5290.
[31] G. Bitar, V. N. Vestad, A. M. Lekkas, and M. Breivik, "Warm-started optimized trajectory planning for asvs," *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 308–314, 2019.
[32] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.
[33] T. Schoels, P. Rutquist, L. Palmieri, A. Zanelli, K. O. Arras, and M. Diehl, "CIAO$^*$: Mpc-based safe motion planning in predictable dynamic environments," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6555–6562, 2020.
[34] A. B. Martinsen, A. M. Lekkas, and S. Gros, "Autonomous docking using direct optimal control," *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 97–102, 2019.
[35] H. Yan, H. Wang, Y. Chen, and G. Dai, "Path planning based on constrained delaunay triangulation," in *2008 7th World Congress on Intelligent Control and Automation*. IEEE, 2008, pp. 5168–5173.
[36] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2182–2189, 2017.
[37] L. P. Chew, "Constrained delaunay triangulations," *Algorithmica*, vol. 4, no. 1-4, pp. 97–108, 1989.
[38] F. P. Preparata and M. I. Shamos, *Computational geometry: an introduction*. Springer-Verlag, 1985.
[39] T. Tsang, D. Himmelblau, and T. F. Edgar, "Optimal control via collocation and non-linear programming," *International Journal of Control*, vol. 21, no. 5, pp. 763–768, 1975.
[40] D. Živojević and J. Velagić, "Path planning for mobile robot using dubins-curve based rrt algorithm with differential constraints," in *2019 International Symposium ELMAR*. IEEE, 2019, pp. 139–142.
[41] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
[42] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.