Constrained Trajectory Planning for Wheeled Ground Robots in Unstructured Environments Modeled by B-Spline Approximation

Damien Hoareau and Alexandre Chapoutot

***Abstract*— Planning trajectories for wheeled unmanned ground vehicles (UGVs) in unstructured environments remains a significant challenge, particularly due to the discrete nature of most map representations. Traditional grid-based exploration algorithms, while widely used, frequently neglect critical factors such as dynamics and terrain traversability. Alternative methods, rooted in optimal control, are typically designed for structured environments and rely on collision constraints involving analytically defined or convexified shapes—limiting their applicability in real-world, cluttered scenarios. This work introduces a novel approach of the trajectory planning problem for unstructured environments. By leveraging cubic B-spline approximations of the map, we enable seamless integration into optimization solvers, combining flexibility with computational efficiency. The problem is framed within the SCvx algorithm, guaranteeing polynomial-time convergence and robust performance. A rigorous analysis of approximation errors is conducted to ensure the practical safety of the generated trajectories, with a focus on adherence to traversability constraints. The proposed approach is validated through comprehensive numerical simulations, demonstrating its effectiveness in complex, real-world conditions.**

I. INTRODUCTION

Trajectory planning for wheeled unmanned ground vehicles (UGVs) is one of the key challenges in autonomous robot navigation [1], especially in unstructured environments. Recent advances in optimization techniques and sensor fusion have significantly improved the performance and accuracy of trajectory planning algorithms, making them more effective in such complex conditions [2]. Additionally, the challenge of navigating complex and dynamic environments with cluttered and moving obstacles has garnered more attention, resulting in the development of novel methods [3], [4]. However, unstructured environments inherently increase the path planning difficulty, for example, where a UGV must adapt its speed to the ground surface profile while avoiding locally non-traversable zones and obstacles. Structured environments are often considered, leading to better performance and more reliable outcomes, especially when assuming a flat ground hypothesis. The representation of obstacles and the specific characteristics of the configuration space, such as traversability, must be reliably modeled.

In robotic navigation, the initial phase typically involves acquiring map data through perception. This data is then discretized and represented in a grid structure for further processing [5]. Other forms of map representation, such as topological maps and metric maps, also exist [6]. A variety of planning algorithms, including A*, Dijkstra's algorithm, and Rapidly-exploring Random Trees (RRT), employ heuristics or optimization techniques [7], [8] to determine paths in map-based environments. These planners are efficient for relatively simple environments, but their computational and memory requirements grow significantly in larger or more complex environments. In addition, the inclusion of constraints on system states, control inputs, and the environment adds complexity and reduces efficiency. An alternative solution is to formulate the problem as an optimization problem, which offers greater flexibility and allows for easier inclusion of various constraints. For example, Model Predictive Control (MPC) methods [9] can efficiently handle dynamic constraints such as velocity limits or non-holonomic restrictions. Additionally, methods such as sequential quadratic programming (SQP) [10], direct collocation, or successive convexification (SCvx) [11] can be employed to minimize objectives like energy consumption or travel time, while still respecting constraints imposed by the environment and the control system. The challenge lies in effectively modeling the various constraints, particularly in the context of unstructured environments.

Some works propose preprocessing the map of unstructured environments to define safe and convex zones for trajectory optimization. This can reduce the complexity of the optimization problem, ensuring more efficient and safer path planning. Han *et al.* [12] propose a method for decomposing unstructured environments by constructing free-space corridors using convex polygons. This approach ensures collision-free navigation and enhances safety along the path. However, the computational cost can be significant, especially in large or complex environments. Additionally, incorporating dynamic constraints, such as area-dependent speed limits caused by terrain irregularities, requires further modeling or preprocessing. To the best of our knowledge, this aspect has not been thoroughly investigated. Existing studies primarily focus on representing unstructured obstacles in the environment, while terrain irregularities remain largely unaddressed. The notable exception is [13], which considers a complex 3D model of a UGV and models terrain roughness as a cost to minimize during trajectory planning. The composite formulation of the framework, which combines an integrated potential field function, Voronoi-based preplanning, and a nonlinear optimal control model solved via *Interior Point Optimizer* (IPOPT) tool, inevitably increases the compu-

This work is supported by *Agence de l'Innovation de Défense* – AID – via *Centre Interdisciplinaire d'Études pour la Défense et la Sécurité* – CIEDS – (project – APRO) and by the French National Research Agency (ANR) under the "ANR-23-MOXE-0003" project.

Damien Hoareau and Alexandre Chapoutot are with the Computer Science and Systems Engineering Laboratory (U2IS), ENSTA, Institut Polytechnique de Paris, 91120 Palaiseau, France {hoareau, chapoutot}@ensta.fr

tational and implementation complexity. In contrast, the approach proposed in this paper utilizes a unified framework, coupled with an effective and straightforward formulation of the path planning problem. Additionally, convex optimization solvers are leveraged, providing significant efficiency in terms of computational cost.

This paper introduces a novel approach to trajectory planning in unstructured environments by formulating an *optimal control problem* (OCP) that addresses obstacle avoidance and velocity constraints while leveraging terrain traversability. The system constraints are modeled through a cubic B-spline approximation of the unstructured environment's traversability map, which enhances both the formulation and computational efficiency of the optimization problem. B-splines offer high numerical stability and can easily adapt to dynamic changes and irregularities, making them ideal for real-time trajectory optimization. Furthermore, B-spline approximation errors can be bounded and accounted for in constraints of OCP. The OCP is formulated using the SCvx algorithm, ensuring computational efficiency and leveraging the benefits of the convexification paradigm. Furthermore, recent advancements enable the leveraging of online embedded computations through code generation [14]. The constraints are formulated to align with the algorithm's structure, ensuring efficient implementation and effective optimization. While the proposed approach has not been extensively explored, numerical simulations are provided to demonstrate its effectiveness.

The paper is organized as follows. Section II explores the key concepts of B-spline approximation. Section III presents the method for generating unstructured environments traversability map used in the numerical simulations. Section IV states the problem formulation. Section V presents and discusses the results. Section VII concludes.

II. Preliminaries on B-splines

The Bivariate B-spline [15] approximation is given by:

$$S(x,y) = \sum_i \sum_j c_{ij} B_{i,n}(x) B_{j,n}(y), \tag{1}$$

where $B_{i,n}(x)$ and $B_{j,n}(y)$ are n-th order B-splines defined over the nodal grids $\{x_i\}$ and $\{y_j\}$, and c_{ij} are the control points associated with these grid points. The coefficients c_{ij} are determined by solving a system of linear equations based on the function values at selected points. The n-th order B-splines are recursively defined as follows:

$$B_{i,n}(x) = \frac{x - x_i}{x_{i+n-1} - x_i} B_{i,n-1}(x) + \frac{x_{i+n} - x}{x_{i+n} - x_{i+1}} B_{i+1,n-1}(x). \tag{2}$$

These B-splines can be computed using the Boor-Cox recursion formula, starting from the first-order B-splines defined as:

$$B_{i,1}(x) = \begin{cases} 1, & x_i \leqslant x < x_{i+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The function $S(x,y)$ provides a smooth and computationally efficient approximation in two dimensions. The partial derivatives with respect to x and y are computed as follows:

$$\frac{\partial S(x,y)}{\partial x} = \sum_i \sum_j c_{ij} \frac{\partial B_{i,n}(x)}{\partial x} B_{j,n}(y), \tag{4a}$$

$$\frac{\partial S(x,y)}{\partial y} = \sum_i \sum_j c_{ij} B_{i,n}(x) \frac{\partial B_{j,n}(y)}{\partial y}. \tag{4b}$$

The first derivative of an n-th order B-spline is given by:

$$B'_{i,n}(x) = \frac{B_{i,n-1}(x)}{x_{i+n-1} - x_i} - \frac{B_{i+1,n-1}(x)}{x_{i+n} - x_{i+1}}. \tag{5}$$

For cubic B-splines ($n = 3$), the resulting function $S(x,y)$ is globally C^2-continuous. In this work, we adopt a matrix-based formulation rather than the recursive Boor-Cox approach. This choice is particularly well-suited for structured grids, where the uniform knot spacing allows for a constant coefficient matrix. As a result, the evaluation of the B-spline surface becomes computationally efficient and avoids the overhead of recursive calculations. For cubic B-splines, the basis function matrix representation is as follows:

$$P(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{6}$$

where t is the parameter, and P_0, P_1, P_2, P_3 are the control points.

III. Generation of Traversability Map

In order to perform trajectory planning, a map generation workflow is designed to represent random traversability of unstructured environments. To this end, without loss of generality, Perlin noise [16] is used to generate a terrain-based traversability map (TM) that is both unstructured and coherent. This noise function guarantees smooth variations while maintaining randomness, making it particularly effective for simulating natural landscapes. The generated map can be customized using two key parameters—octaves and seed. The octaves parameter determines the level of detail in the map by specifying how many layers of noise are combined. Higher octave values result in finer, more detailed features. The seed parameter introduces variability, enabling the generation of a different map each time, even with identical octave values. Once generated, the map is normalized to the range $[0, 1]$, implicitly encoding various terrain features such as slopes and holes. Figure 1 illustrates several examples of normalized maps generated using Perlin noise.

In real-world applications, traversability maps are typically generated using geometric-based approaches [17]. These maps rely on *digital elevation models* (DEMs), which are constructed from high-resolution LiDAR measurements. The process involves two key steps: first, ground segmentation is performed to distinguish the terrain surface, and second,

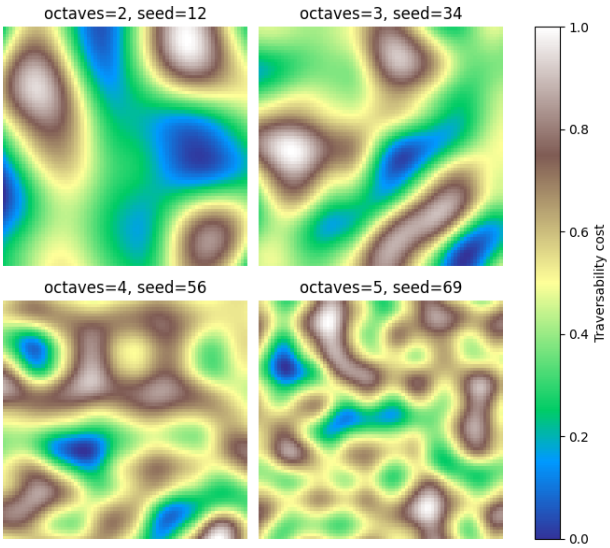

Fig. 1. Examples of normalized maps generated using Perlin noise.

obstacles are identified and segmented based on their height relative to the ground reference. To ensure the relevance of our method for practical deployment, we emulate this elevation mapping process using Perlin noise, which effectively replicates the variability and complexity observed in real LiDAR-derived terrain data as in [18]. This simulation strategy bridges the gap between theoretical development and real-system implementation, validating the applicability of our approach in unstructured environments.

IV. Trajectory Planning Problem

Let $P(x, y) : \mathbb{R}^2 \to \mathbb{R}$ denote a terrain irregularities score of an unstructured environment.

Remark 1. *Here, the term unstructured environment primarily refers to irregular variations in terrain elevation. Features such as holes, slopes, and bumps characterize such environments. More broadly, the spatial distribution of obstacles can also be considered part of an unstructured environment. In summary, the terrain irregularities score map can be derived from various assessments, depending on user considerations and system capabilities.*

A continuous mapping of the terrain irregularities allows for the construction of an associated traversability score map.

Assumption 1. *There exists an operator $\mathcal{T}$ such that:*

$$\mathcal{T} : P(x, y) \mapsto T(x, y),$$

where $T(x, y)$ represents the traversability score map derived from $P(x, y)$. The operator $\mathcal{T}$ is continuous, ensuring smooth transitions between different traversability regions.

Remark 2. *The traversability map (TM) is obtained by processing the terrain irregularities map, which is initially generated from discrete sensor measurements and preprocessing algorithms.*

We consider the case where the traversability map has already been obtained. Without loss of generality, we focus on maps derived from elevation assessments, primarily using Perlin noise, as presented in the previous section.

Since the traversability map $T(x_i, y_j)$ is inherently discrete, using it directly in optimization problems can be challenging. To address this issue, we apply a cubic bivariate B-spline approximation, denoted $\tilde{T}(x, y)$, to obtain a smooth, continuous representation of the TM by applying Equation (4) for $n = 3$. This approximation preserves the essential features of the original discrete map while enabling the smoothness required for subsequent calculations, such as dynamic optimization. Consequently, cubic B-splines provide a highly suitable method for approximating the traversability map. Since the knot positions are fixed in the chosen formulation (section II), the resulting B-spline surface $\tilde{T}(x, y)$ constitutes only an approximation of the discrete traversability map $T(x_i, y_j)$. Nevertheless, the approximation error can be bounded under suitable conditions, making $\tilde{T}(x, y)$ reliable for integration into subsequent analyses, including optimization and trajectory planning.

Assumption 2. *The traversability map $T(x, y)$ is sufficiently smooth locally such that the following error bound holds:*

$$|T(x, y) - \tilde{T}(x, y)| \leqslant C \cdot h^2 \cdot \max_{(x,y)\in[a,b]\times[c,d]} |D^2 T(x, y)|, \quad (7)$$

where h represents the grid step size, C is a constant depending on the spline construction and mesh regularity, $\tilde{T}(x, y)$ is the cubic B-spline approximation, and D^2 denotes second order partial derivatives.

a) Physical Interpretation and Assumption Validity: The local smoothness of the TM extracted from an elevation terrain map is considered valid in the presence of gradual positive and negative slopes. It is assumed that discontinuities are either nonexistent or sufficiently small to be negligible. In addition, the construction of a traversability map also depends on the system under study. For certain systems, terrain irregularities or some discontinuities may be perceived as non-impacting, thus extending the validity domain of this assumption.

Trajectory planning based on traversability specifications is subsequently carried out under the system's dynamic constraints, which can be classified into the following categories:

- **Non-Traversable Area (NTA)**: Regions that cannot be traversed.
- **Constrained Traversable Area (CTA)**: Regions that are traversable but subject to dynamic restrictions.
- **Traversable Area (TA)**: Regions without any restrictions on the system's dynamics.

Hence, the dynamic constraints associated with the traversability specifications can be formally defined. In this study, the following constraints are taken into account.

For the NTA, a threshold value τ_1 is defined and is given by:

$$\tilde{T}(x, y) \leqslant \tau_1 \quad (8)$$

Its value is determined by the user, it can be extracted from the robot dynamics or from operation constraints. This constraint can be seen as a collision avoidance constraint. For example, if a robot cannot go over an obstacle higher than 30cm then $\tau_1 = 0,3$. For both TA and CTA, a threshold value τ_0 is defined, which leads to the following constraint on the maximum velocity:

$$v_{\max} = \frac{\text{max\_speed}}{1 + K \max\left(\tilde{T}(x,y) - \tau_0, 0\right)} \tag{9}$$

where $v_{\max}$ denotes the system's maximum admissible linear speed, max_speed represents the system's unconstrained maximum velocity, K is a constant velocity gain, and τ_0 is the threshold associated with the TA and CTA specifications. For example, a robot can go over obstacles at full speed when obstacles are not higher than 10cm so in this case $\tau_0 = 0.1$. When $\tilde{T}(x,y)$ is below τ_0, no speed restriction is applied. We choose to incorporate the velocity constraint in this problem using the proposed formulation, which includes a user-defined threshold τ_0. This parameter provides a specific dynamic behavior for the maximum velocity, while remaining adaptable by the user.

Example 1. *Consider the kinematic model of a Dubin's car:*

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_1 \cos(\theta) \\ u_1 \sin(\theta) \\ u_2 \end{bmatrix}$$

where x and y are the position coordinates of the vehicle in the plane, and θ is the orientation angle. The control inputs are given by u_1, which represents the linear velocity, and u_2, which is the angular velocity. More elaborate dynamics may be considered such as [19] which are variations of Dubin's model. Indeed many four-wheel ground robots are based on differential kinematic model.

The optimal control problem for trajectory generation is formulated as:

$$\begin{aligned} \min_{u,\mathbf{x}} \quad & \sum_{k=0}^{N-1} \left(u_{1,k}^2 + u_{2,k}^2 + \tilde{T}(x_k, y_k) \right) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, 1, \ldots, N-1 \\ & 0 \leqslant u_{1,k} \leqslant v_{\max}^k, \quad |u_{2,k}| \leqslant \omega_{\max} \\ & \tilde{T}(x_k, y_k) \leqslant \tau_1, \quad \forall k \\ & \mathbf{x}_0 = \mathbf{x}_{start}, \quad \mathbf{x}_N = \mathbf{x}_{\text{end}} \\ & \mathbf{u}_{1,0} = \mathbf{u}_{2,0} = 0, \quad \mathbf{u}_{1,N} = \mathbf{u}_{2,N} = 0 \\ & x_{\min} \leqslant x_k \leqslant x_{\max}, \quad y_{\min} \leqslant y_k \leqslant y_{\max}, \quad \forall k \end{aligned} \tag{10}$$

where the dynamics of the robot is given by discrete-time non-linear function $\mathbf{f}$, the state vector is given by:

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}, \quad \mathbf{u}_k = \begin{bmatrix} u_{1,k} \\ u_{2,k} \end{bmatrix}$$

and $\mathbf{x}_0$ and $\mathbf{x}_N$ represent the initial and final positions, respectively.

Algorithm 1 SCvx Algorithm

Input : Initial trajectory $(\mathbf{x}^{(1)}, \mathbf{u}^{(1)})$, tolerance ϵ_{tol}
Output: Optimized trajectory $(\mathbf{x}^{(i+1)}, \mathbf{u}^{(i+1)})$

Data: Initialize trust region radius $\eta^{(1)} > 0$
Repeat
  Linearize the dynamics around the current trajectory $(\mathbf{x}^{(i)}, \mathbf{u}^{(i)})$ to obtain matrices $A_k^{(i)}$, $B_k^{(i)}$, $C_k^{(i)}$, $D_k^{(i)}$, and $E_k^{(i)}$
  Update the convex sub-problem parameters using the linearized dynamics
  Solve the convex sub-problem to obtain the optimal solution $(\mathbf{x}^{(i+1)}, \mathbf{u}^{(i+1)})$
  if improvement in cost function **then**
    Accept the solution and update the trust region radius $\eta^{(i+1)}$
  end
  else
    Reduce the trust region radius $\eta^{(i+1)}$ and recompute
  end
EndRepeat cost reduction $\leqslant \epsilon_{tol}$;
return Optimized trajectory $(\mathbf{x}^{(i+1)}, \mathbf{u}^{(i+1)})$

Remark 3. *Note that a fixed-time problem with a specified end time t_f is being considered.*

A. SCvx Algorithm

The SCvx algorithm is a *sequential convex programming* (SCP) method designed to solve a sequence of locally convex approximations of an optimization problem. At each SCvx iteration, the original non-convex problem is convexified by linearizing the non-convexities. The resulting convex sub-problem is then discretized with respect to time, typically using a *first-order hold* (FOH) interpolation method. However, the linearization introduces artifacts such as artificial infeasibility and unboundedness. To address these issues, the linearized equations are augmented by adding a virtual control term, modelled by matrix E in Algorithm 1, and a trust region, as described in [20], [21]. At each iteration, the convex sub-problem is solved using an *second order cone programming* (SOCP) solver. The obtained solution is then used to update the SCvx algorithm parameters. The stopping criterion is based on the difference between the current iteration's trajectory solution and the previous one, with a small difference indicating that further improvements are unlikely. Algorithm 1 summarizes the SCvx structure.

For further details on the SCvx algorithm, we refer the reader to [21]. For the purpose of this work, we focus on two key aspects of the algorithm: the *outer loop*, where the non-convexities are linearized at each iteration, and the *inner loop*, where the convex sub-problem is solved using an SOCP solver.

B. Constraints Linearization

In its current form, the problem in (10) is not convex with respect to the constraints and requires linearization. The focus is placed on the linearization of the dynamic constraints in (8) and (9). The linearization of the other constraints is well-documented in the literature, and interested readers can refer to [21] for further details. To address this, the first-order Taylor expansion is applied. This is made easy thanks to the good derivation property of B-Spline, see Equation (5). Thus, the constraint defined in Equation (8) is transformed as follows:

$$\tilde{T}(x_k, y_k) = \tilde{T}(x_k^p, y_k^p) + \frac{\partial \tilde{T}}{\partial x}(x_k^p, y_k^p)(x_k - x_k^p) + \frac{\partial \tilde{T}}{\partial y}(x_k^p, y_k^p)(y_k - y_k^p) \leqslant \tau_1 + \alpha \quad (11)$$

where $(.)^p$ denotes the values obtained from the previous iteration $(i-1)$, α is a positive relaxation variable.

The constraint (9) can be rewritten by exploiting only the values from the previous iteration:

$$0 \leqslant u_{1,k} \leqslant \frac{\text{max\_speed}}{1 + K \max\left(\tilde{T}(x_k^p, y_k^p) - \tau_0, 0\right)}. \quad (12)$$

Hence, the right-hand side term now depends only on known parameters, and the nonconvexity with respect to x_k and y_k has been relaxed.

C. Error Bound and Step Size Constraint

To ensure that the constraints are physically respected, it is essential to guarantee that the map approximation is sufficiently accurate, while also correcting any potential biases that could lead to violations of these constraints.

Proposition 1. *Let $T(x_i, y_i)$ be the original discrete traversability map, approximated by a cubic B-spline $\tilde{T}(x, y)$ at the control points (x_i, y_j). If the maximum error at the control points is*

$$E_{\max} = \max_{(x_i, y_j)} |T(x_i, y_j) - \tilde{T}(x_i, y_j)|,$$

then the error over the entire domain $[a, b] \times [c, d]$ satisfies

$$\max_{(x,y)\in[a,b]\times[c,d]} |T(x, y) - \tilde{T}(x, y)| \leqslant E_{\max}.$$

Proof. By Assumption 2, $T(x, y)$ is C^2 and its cubic B-spline approximation $\tilde{T}(x, y)$ is smooth with local support. The B-spline basis forms a convex partition of unity, so no new extrema are introduced between control points. Hence the maximum error over the domain is bounded by the maximum error at the control points, $E_{\max}$. □

Corollary 1. *Using the bound provided by Assumption 2, the grid step size h must satisfy*

$$C \cdot h^2 \cdot \max_{(x,y)\in[a,b]\times[c,d]} |D^2 T(x, y)| \leqslant E_{\max}$$

to guarantee that the error over the entire domain does not exceed $E_{\max}$.

Proof. By Proposition above and the C^2 bound from Assumption 2, the error is bounded by $C\, h^2 \max_{(x,y)} |D^2 T(x, y)|$. To ensure this does not exceed $E_{\max}$, the inequality above must hold. □

D. Approximation-Based Safety Constraints

Finally, to ensure safe physical collision avoidance and respect the traversability specifications, the equations (11) and (12) are adapted as follows.

$$\tilde{T}(x_k, y_k)_p r = \tilde{T}(x_k^p, y_k^p) + \frac{\partial \tilde{T}}{\partial x}(x_k^p, y_k^p)(x_k - x_k^p) + \frac{\partial \tilde{T}}{\partial y}(x_k^p, y_k^p)(y_k - y_k^p) \leqslant \tau_1 - \varepsilon + \alpha, \quad (13)$$

$$0 \leqslant u_{1,k} \leqslant \frac{\text{max\_speed}}{1 + K \max\left(\tilde{T}(x_k^p, y_k^p) + \varepsilon - \tau_0, 0\right)}, \quad (14)$$

with the new condition:

$$\varepsilon := \min_h E_{max} \quad (15)$$

The parameter ε ensures that the constraints are physically respected, allowing for a margin of error, which may result in global over-estimation of $\tilde{T}(x, y)$. This parameter should be chosen to be as small as possible by decreasing the discretization step h.

V. Numerical simulations

In this section, the simulation results obtained for the considered example are presented. The simulation parameters used are $\mathbf{x}_{\text{start}} = [-4, -7, 0]^\top$, $\mathbf{x}_{\text{end}} = [4, 5, 0]^\top$, $N = 40$, $w_{\max} = \pi/3$, max_speed $= 20/3.6$, $x_{\min} = y_{\min} = -7.5$, $x_{\max} = y_{\max} = 7.5$, $h = 0.2$, $K = 5$, $\tau_0 = 0.2$, $\tau_1 = 0.9$, and $\varepsilon = 0.05$. We considered different value of final time $t_f = [8, 9, 10, 11]$ s. The SCvx convex sub-problem is solved using ECOS solver [22], with convergence achieved after 15 iterations for $t_f = 8$ s (Ubuntu 20.04.6 LTS, Intel® Core™ i9-9900K CPU @ 3.6 GHz). Figure 2 shows the planned trajectory for the Dubin's car dynamics on a map generated with octave $= 4$ and seed $= 10$.

All the constraints are satisfied at the k discretization points. For each final time t_f, the algorithm successfully minimized the traversability score and the controls along the trajectories. In addition, all the constraints are satisfied. Because of the nonlinear nature of the problem the obtained solutions are only locally optimal. Hence, starting from different initial trajectories may provide different solutions.

Figure 3 presents the velocity evolution subject to the traversability velocity constraints. As observed, the CTA specification is respected at the discretization points. Furthermore, u_1 is equal to the maximal value as the optimizer found a trajectory allowing the maximal value. Figure 4 presents the path cost along the optimal trajectory.

For the entire trajectory, the cost remains below the threshold value τ_1. Figure 5 presents the angular velocity evolution along the optimal trajectory.

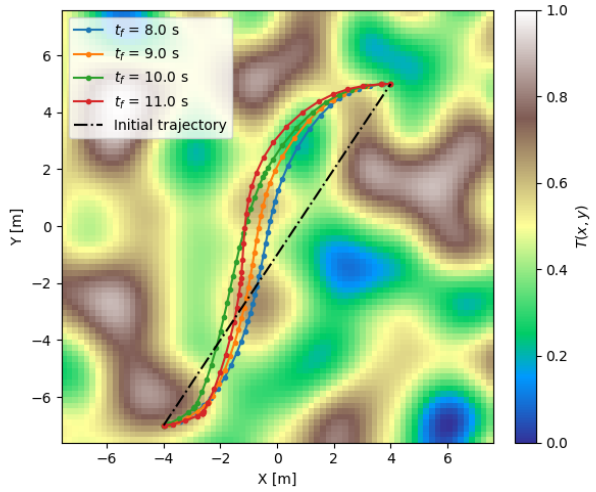

Fig. 2. Optimized trajectories corresponding to different final times t_f (colored lines). The initial trajectory is shown in black.

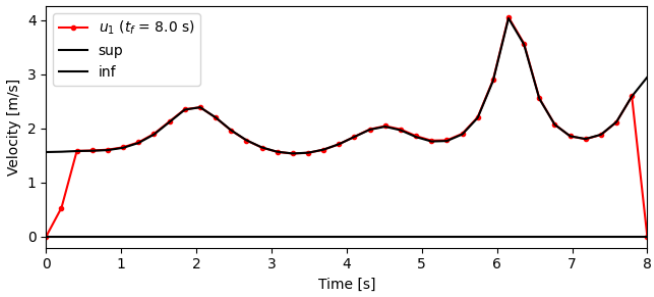

Fig. 3. Evolution of the control u_1.

VI. Dicussion

By exploiting the SCvx paradigm and reusing the previous iteration values at each convexification step, some problem variables can be treated as parameters, thereby reducing the numerical complexity. Using cubic B-spline interpolation, we were able to formulate dynamic environment constraints on the system state and control variables more naturally. While the approximation error can be bounded under certain assumptions, its value is incorporated into the problem constraints to ensure practical feasibility of the obtained solutions. Hence, as presented, we are able to obtain locally optimal solutions to the original complex control problem.

a) Limitations: Due to the nonlinear nature of the problem, local minima exist within the solution set. Within the SCvx paradigm, feasibility can still be achieved even when starting from an infeasible initial trajectory that passes through these minima; however, it is not guaranteed. The algorithmic framework can be further enhanced to ensure solution feasibility. Oguri *et al.* [23] proposed the SCvx* algorithm, highlighting its strong global convergence to a feasible local optimum. The feasibility is ensured using an augmented Lagrangian approach. By combining this method with the ideas discussed before, the robustness of the algorithm can be further improved.

Moreover, inter-sample constraint violations may occur in

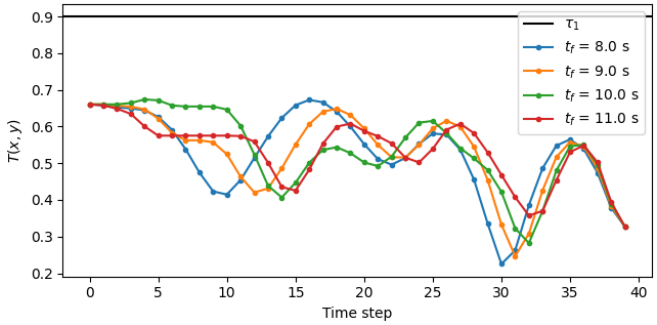

Fig. 4. Traversability cost along the optimal trajectory.

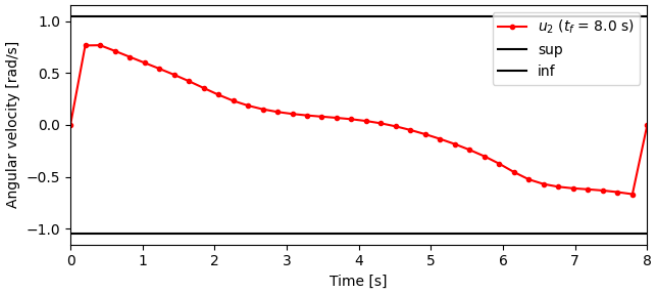

Fig. 5. Evolution of the control u_2

the proposed approach. While this aspect is not explored in this paper for the sake of brevity, readers can refer to [24] for further details. Kamath *et al.* [25] proposed a time-interval dilation procedure to address this issue. This more advanced SCvx variant can be employed instead of the simpler version used in the proposed approach, and its impact will be examined in future work.

Finally, since the problem formulation assumes a smooth traversability map, the overall feasibility may appear limited. Nonetheless, this limitation can be mitigated by using local approximations of the map based on continuous piecewise segments, which provide a more flexible representation of complex terrains.

b) Extended capabilities: In its current form, the optimal control problem can be easily extended by introducing additional constraints. In particular, inequality constraints can be added to represent physical obstacles. In the context of robotic perception, these obstacles can be derived, for instance, from semantic map segmentation. For example, the approach proposed in [26] can be used to approximate the ground surface for the construction of the traversability map (TM). Subsequently, semantic segmentation can identify non-traversable obstacles, which are then incorporated into the framework. This approach highlights the versatility of the proposed formulation, demonstrating its ability to accommodate a variety of constraints within a unified framework.

Furthermore, as the local subproblem to be solved is a Second-Order Cone Program (SOCP), the approach remains computationally efficient and well-suited for embedded systems. Finally, by adapting the SCvx algorithm, it is possible to reduce the formulation to a Linear Program (LP), which

opens promising opportunities for verified computations [27].

VII. Conclusion

The current work focuses on trajectory planning in unstructured environments using discrete maps. The proposed formulation is straightforward, promoting ease of implementation and integration while enhancing modularity. A smooth cubic B-spline approximation efficiently provides a continuous representation of the discrete map. Obstacle avoidance and velocity constraints are incorporated based on traversability specifications. The problem is formulated within the SCvx framework and solved using a SOCP solver, leveraging sequential convex programming. Approximation errors are accounted for to ensure practical safety. The results demonstrate the effectiveness of the proposed approach. Future directions aim at further improving the robustness of the proposed framework, which already demonstrates efficiency and accessible implementation process, making it well-suited for embedded systems.

References

[1] Y. Guo, Z. Guo, Y. Wang, D. Yao, B. Li, and L. Li, "A survey of trajectory planning methods for autonomous driving—part i: Unstructured scenarios," *IEEE Transactions on Intelligent Vehicles*, 2023.

[2] Z. Han, P. Chen, B. Zhou, and G. Yu, "Real-time navigation of unmanned ground vehicles in complex terrains with enhanced perception and memory-guided strategies," *IEEE Trans. on Vehicular Technology*, 2025.

[3] A. Manjunath and Q. Nguyen, "Safe and robust motion planning for dynamic robotics via control barrier functions," in *Conference on Decision and Control (CDC)*, pp. 2122–2128, IEEE, 2021.

[4] H. Chen, S. Feng, Y. Zhao, C. Liu, and P. A. Vela, "Safe hierarchical navigation in crowded dynamic uncertain environments," in *Conference on Decision and Control (CDC)*, pp. 1174–1181, IEEE, 2022.

[5] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Engineering Science and Technology, an International Journal*, vol. 40, p. 101343, 2023.

[6] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, 2023.

[7] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, "A review of global path planning methods for occupancy grid maps regardless of obstacle density," *Journal of intelligent & robotic systems*, vol. 84, pp. 829–858, 2016.

[8] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119310–119342, 2021.

[9] S. Yu, M. Hirche, Y. Huang, H. Chen, and F. Allgöwer, "Model predictive control for autonomous ground vehicles: A review," *Autonomous Intelligent Systems*, vol. 1, pp. 1–17, 2021.

[10] Y. Jiang, Z. Liu, D. Qian, H. Zuo, W. He, and J. Wang, "Robust online path planning for autonomous vehicle using sequential quadratic programming," in *Intelligent Vehicles Symposium (IV)*, pp. 175–182, IEEE, 2022.

[11] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe, "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," *IEEE Control Systems Magazine*, vol. 42, no. 5, pp. 40–113, 2022.

[12] Z. Han, Y. Wu, T. Li, L. Zhang, L. Pei, L. Xu, C. Li, C. Ma, C. Xu, S. Shen, *et al.*, "An efficient spatial-temporal trajectory planner for autonomous vehicles in unstructured environments," *IEEE Trans. on Intelligent Transportation Systems*, vol. 25, no. 2, pp. 1797–1814, 2023.

[13] J. Hu, Y. Hu, C. Lu, J. Gong, and H. Chen, "Integrated path planning for unmanned differential steering vehicles in off-road environment with 3d terrains and obstacles," *IEEE Trans. on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5562–5572, 2022.

[14] D. Berrah, A. Chapoutot, and P.-L. Garoche, "SCvxPyGen: Autocoding SCvx algorithm," in *IEEE Conference on Decision and Control (CDC)*, pp. 5086–5093, 2024.

[15] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing. i. theory," *IEEE transactions on signal processing*, vol. 41, no. 2, pp. 821–833, 1993.

[16] K. Perlin, "An image synthesizer," in *Proc. of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH'85, (New York, NY, USA), p. 287–296, Association for Computing Machinery, 1985.

[17] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, *Robot-Centric Elevation Mapping with uncertainty estimates*, pp. 433–440.

[18] M. Thoresen, N. H. Nielsen, K. Mathiassen, and K. Y. Pettersen, "Path planning for UGVs based on traversability hybrid A*," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1216–1223, 2021.

[19] L. Caracciolo, A. de Luca, and S. Iannitti, "Trajectory tracking control of a four-wheel differentially driven mobile robot," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 4, pp. 2632–2638 vol.4, 1999.

[20] Y. Mao, M. Szmuk, and B. Acikmese, "Successive convexification of non-convex optimal control problems and its convergence properties," in *Conference on Decision and Control (CDC)*, IEEE, 2016.

[21] Y. Mao, M. Szmuk, X. Xu, and B. Acikmese, "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," 2019.

[22] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *European control conference (ECC)*, pp. 3071–3076, IEEE, 2013.

[23] K. Oguri, "Successive convexification with feasibility guarantee via augmented lagrangian for non-convex optimal control problems," in *IEEE Conference on Decision and Control (CDC)*, pp. 3296–3302, 2023.

[24] S. Uzun, P. Elango, A. G. Kamath, T. Kim, and B. Açıkmeşe, "Successive convexification for nonlinear model predictive control with continuous-time constraint satisfaction," *IFAC-PapersOnLine*, vol. 58, no. 18, pp. 421–429, 2024.

[25] A. G. Kamath, P. Elango, Y. Yu, S. Mceowen, G. M. Chari, J. M. Carson III, and B. Açıkmeşe, "Real-time sequential conic optimization for multi-phase rocket landing guidance," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3118–3125, 2023.

[26] W. Zhang, J. Qi, P. Wan, H. Wang, D. Xie, X. Wang, and G. Yan, "An easy-to-use airborne LiDAR data filtering method based on cloth simulation," *Remote Sensing*, vol. 8, no. 6, 2016.

[27] D. Berrah, D. Hoareau, and A. Chapoutot, "SCvx-Frank-Wolfe algorithm," in *European Control Conference*, 2025.