2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

# Efficient Clothoid Tree-Based Local Path Planning for Self-Driving Robots

Minhyeong Lee and Dongjun Lee

***Abstract*— In this paper, we propose a real-time clothoid tree-based path planning for self-driving robots. Clothoids, curves that exhibit linear curvature profiles, play an important role in road design and path planning due to their appealing properties. Nevertheless, their real-time applications face considerable challenges, primarily stemming from the lack of a closed-form clothoid expression. To address these challenges, we introduce two innovative techniques: 1) an efficient and precise clothoid approximation using the Gauss-Legendre quadrature; and 2) a data-efficient decoder for interpolating clothoid splines that leverages the symmetry and similarity of clothoids. These techniques are demonstrated with numerical examples. The clothoid approximation ensures an accurate and smooth representation of the curve, and the clothoid spline decoder effectively accelerates the clothoid tree exploration by relaxing the problem constraints and reducing the problem size. Both techniques are integrated into our path planning algorithm and evaluated in various driving scenarios.**

## I. Introduction

Self-driving robots, including autonomous vehicles and delivery robots, have garnered substantial attention due to their potential to revolutionize transportation. A critical aspect of these self-driving robots is ensuring road safety and considering multi-modality, highlighting the importance of a fast and feasible sampling-based local path planner. Early efforts in this domain have relied on geometric primitives (e.g., straight lines and circular arcs) for path candidates [1]. However, these motion primitives, while valuable, are only $G^1$ continuous at their joints, thereby sacrificing the continuity of steering angles. Another sampling-based approach incorporates parametric curves, such as polynomials in the Frenét frame [2], [3] and B-splines [4]. Nonetheless, these methods often lack the requisite geometric interpretation for wheeled vehicles, requiring additional considerations regarding the curve attributes.

As an alternative, clothoids exhibit an advantageous feature of linear curvature profiles. This distinctive attribute enhances driving comfort and eases path tracking, rendering them suitable for self-driving robots. Moreover, clothoids offer a wide range of curve shapes while ensuring $G^2$ continuity. Owing to their appealing properties, clothoids find extensive applications in intersection, highway, and railway design. Refer to [5] for the computer-aided design (CAD) of clothoid splines and their applications in highway design.

This work is supported by Technology Innovation Program (20024355, Development of autonomous driving connectivity technology based on sensor-infrastructure cooperation) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea). Corresponding author: Dongjun Lee.

The authors are with the Department of Mechanical Engineering, Institute of Advanced Machines and Design (IAMD) and Institute of Engineering Research (IOER), Seoul National University, Seoul 08826, South Korea (e-mail: minhyeong@snu.ac.kr; djlee@snu.ac.kr).

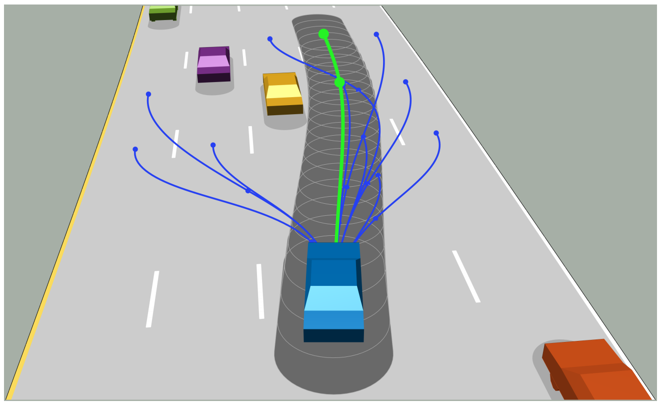


Fig. 1. Driving simulation using the clothoid tree-based path planning.

Motivated by the attractive properties of clothoids, we propose a real-time clothoid tree-based path planning for self-driving robots. For this, we introduce a $G^2$ continuous clothoid tree that addresses other feasibility constraints such as maximum curvature and collision avoidance. However, employing the clothoid tree poses considerable challenges resulting from the lack of a closed-form expression of clothoids. To address these challenges, we present two innovative techniques: 1) an efficient and precise clothoid approximation method using the Gauss-Legendre quadrature; and 2) a data-efficient decoder for clothoid splines that interpolates two given configurations, leveraging the symmetry and similarity of clothoids under translation, rotation, scale, and reflection transformations.

The effectiveness of these techniques is validated through numerical examples. The proposed clothoid approximation method guarantees smooth parameterization and small error of machine epsilon, approximately $2.22 \times 10^{-16}\mathrm{m}$ for a one-meter arc length, with reasonable computational complexity. The clothoid spline decoder effectively accelerates the tree exploration by relaxing the problem constraint and reducing the problem size while exhibiting millimeter-level accuracy, averaging $0.0048 \pm 0.0056\mathrm{m}$ error over a one-meter path. Both techniques are seamlessly integrated into our path planning algorithm and driving scenarios as in Fig. 1. The resultant planning time is $0.0179 \pm 0.0018\mathrm{s}$ without collision checks and $0.0535 \pm 0.0214\mathrm{s}$ with collision checks.

Like our approximation approach, prior studies employed numerical methods for clothoid calculation, falling into three major categories. The first category relies on well-established numerical integration methods like the midpoint and trapezoidal rules. These methods sacrifice continuous representations and demand a high computational cost for precision. Higher-order numerical quadrature is known to provide better accuracy, yet, to our knowledge, no prior works have utilized

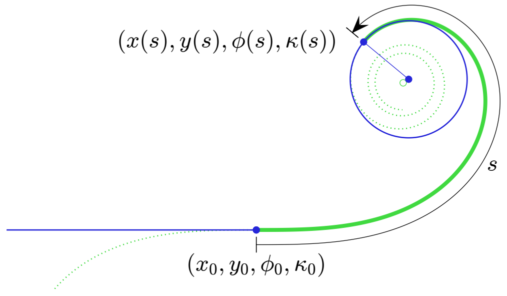
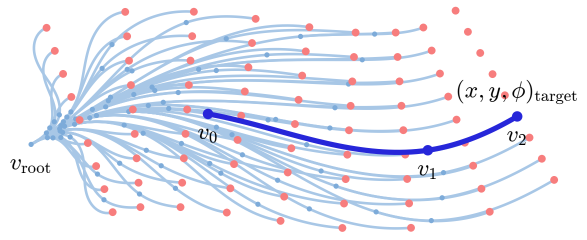

$(x(s), y(s), \phi(s), \kappa(s))$
$s$
$(x_0, y_0, \phi_0, \kappa_0)$


Fig. 2. Left. A clothoid segment. The inverse of the local curvature is the radius of the osculating circle. Right. A clothoid tree. The adjacent clothoids are G$^2$ continuous. Light red markers are the target configurations, and the dark blue line is the new branch clothoid spline.

them due to the numerical complexity. Our adaptation of the Gauss-Legendre quadrature [6, Chapter 25.4] is the very first result along this line. The second category adopts easily computable curves for clothoid approximations, such as Bézier and B-spline curves [7], s-power series [8], and circular arc splines [9], imposing a considerable computational burden from the high-order curve optimization. The last category combines precomputed lookup tables with easy-to-compute curves, offering commendable speed and accuracy. For example, [10] utilizes circular arc interpolations, and [11] adopts Bézier curves. These numerical methods, however, have been focused on approximating clothoid segments and have not been integrated into comprehensive local path planning, as we attain in this paper.

On the other hand, there have been notable attempts to incorporate clothoids into local path planning. In [12], the combination of clothoids and circular arcs are exemplified for a lane change path optimization, while using rational approximations of Fresnel integrals [6, Chapter 7.3] for clothoid calculations. Authors in [13] use precomputed cornering paths consisting of two clothoids and one circular arc to control an Audi TTS at the limits. In [14], [15], clothoid tentacles are employed for local path planning. Despite all these efforts, the resultant path shape is restricted and may not be sufficient to address the intricate driving scenarios.

To address complex scenarios, clothoid tree-based methods have been introduced. In [16], a random clothoid tree is utilized to generate an optimal racing line, incorporating G$^1$ fitting with clothoids [17]. For autonomous parking, [18] introduces a target-oriented clothoid tree. These clothoid tree-based approaches focus solely on achieving G$^1$ continuity at their joints and do not address real-time path planning. While [19] suggests an interpolating clothoid spline with G$^2$ continuity, it is also unsuitable for real-time applications due to its nonlinear programming formulation. Importantly, our work stands out as the first to employ a G$^2$ continuous clothoid tree and a data-driven interpolating clothoid spline decoder for the real-time local path planning.

The remainder of this paper is organized as follows: Section II introduces the proposed path planning framework; Section III details the clothoid approximation method and presents its numerical examples; Section IV details the clothoid spline decoder and presents its numerical examples; Section V demonstrates the proposed algorithm in driving scenarios; and Section VI concludes the paper.

## II. Clothoid Tree-Based Path Planning

Our path planning algorithm relies on a tree structure that incorporates clothoids, referred to as a *clothoid tree*, to guide the robot towards the target configurations. This clothoid tree-based path planning consists of two steps: 1) expanding the tree from the initial configuration to the predefined target configurations; and 2) selecting the optimal path from the expanded tree with the minimum overall cost. The tree exploration and the path selection steps, as well as the definition of clothoids and clothoid trees, are detailed in the following subsections.

### A. Clothoid

Clothoids are characterized by their linear curvature $\kappa : \mathbb{R}_{\geq 0} \to \mathbb{R}$ and quadratic tangent angle $\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}$ s.t.

$$\kappa(s) = \kappa_0 + \sigma s \tag{1}$$

$$\phi(s) = \phi_0 + \kappa_0 s + \frac{\sigma}{2} s^2 \tag{2}$$

where $s \in \mathbb{R}_{\geq 0}$ is the arc length parameter, $\phi_0 \in \mathbb{R}$ is the initial heading angle, $\kappa_0 \in \mathbb{R}$ is the initial curvature, and $\sigma \in \mathbb{R}$ is the curve sharpness. Then the parametric representation of clothoids can be expressed as

$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \int_0^s \begin{bmatrix} \cos \\ \sin \end{bmatrix} \left(\phi_0 + \kappa_0 \xi + \frac{\sigma}{2} \xi^2\right) \mathrm{d}\xi \tag{3}$$

where $(x_0, y_0) \in \mathbb{R}^2$ is the initial position and $(\cos, \sin)(\cdot) : \mathbb{R} \to \mathbb{R}^2$ is the unit tangent vector operator. See Fig. 2 for the visualization of clothoids. It is important to note that an analytical solution of (3) is not available, necessitating the usage of numerical integration, as elaborated in Section III.

### B. Clothoid Tree

The clothoid tree is defined as a directed rooted tree, where each edge exhibits a linear curvature, and adjacent edges are G$^2$ continuous. The tree can be represented as a tuple:

$$\mathsf{CT} = (\mathcal{V}, \mathcal{E}, g_{\text{endpoints}}, g_l, g_x, g_y, g_\phi, g_\kappa) \tag{4}$$

where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ is a set of edges, and $g_\star$ denotes property functions associated with each vertex and edge. Specifically, $g_{\text{endpoints}} : \mathcal{E} \to \mathcal{V} \times \mathcal{V}$ provides the tail and head vertices of each edge, $g_l : \mathcal{E} \to \mathbb{R}_{\geq 0}$ provides the arc length of each edge, and $g_x, g_y, g_\phi, g_\kappa : \mathcal{V} \to \mathbb{R}$ provides the configuration of each vertex. The root vertex of the tree corresponds to the initial state, while the leaf vertices correspond to the predefined target configurations, see Fig. 2.

### C. Clothoid Tree Expansion

Similar to other tree exploration algorithms, the clothoid tree expands iteratively by adding a new branch as illustrated in Fig. 2. During the iteration, the previous tree $(\mathcal{V}_{\text{prev}}, \mathcal{E}_{\text{prev}})$ and the current target configuration $(x, y, \phi)_{\text{target}}$ are given. The new branch, an $N$-segment clothoid spline to be added to the tree, is denoted by $(\mathcal{V}_{\text{new}}, \mathcal{E}_{\text{new}})$ where $\mathcal{V}_{\text{new}} = \{v_i\}_{i=0}^{N}$ is a sequence of vertices and $\mathcal{E}_{\text{new}} = \{e_i\}_{i=1}^{N}$ is a sequence of edges such that $g_{\text{endpoints}}(e_i) = (v_{i-1}, v_i)$. The tail vertex, or the parent vertex $v_0 \in \mathcal{V}_{\text{prev}} \cap \mathcal{V}_{\text{new}}$, is selected from the previous vertex set $\mathcal{V}_{\text{prev}}$. The head vertex $v_N$ is positioned to meet the target configuration by $(x, y, \phi)_N = (x, y, \phi)_{\text{target}}$. The resultant expanded tree is denoted by $(\mathcal{V}_{\text{next}}, \mathcal{E}_{\text{next}})$ with $\mathcal{V}_{\text{next}} = \mathcal{V}_{\text{prev}} \cup \mathcal{V}_{\text{new}}$ and $\mathcal{E}_{\text{next}} = \mathcal{E}_{\text{prev}} \cup \mathcal{E}_{\text{new}}$. Although the target curvature constraint $\kappa_N = \kappa_{\text{target}}$ is available, it has been observed that the position and heading constraints are adequate for the local path planning.

Then, each iteration of tree exploration can be formulated as a mixed-integer programming (MIP) problem:

$$\begin{aligned} \underset{\substack{v_0 \in \mathcal{V}_{\text{prev}} \\ \{(x,y,\phi,\kappa)_i\}_{i=1}^{N}, \{l_i\}_{i=1}^{N}}}{\text{minimize}} \quad & J_{\text{expand}}(v_N, \mathcal{E}_{\text{root}\to N}) \\ \text{subject to} \quad & (x, y, \phi)_N = (x, y, \phi)_{\text{target}} \\ & |\kappa_i| \le \kappa_{\max} \\ & \texttt{dist}(\mathcal{O}_{\text{ego}}(\mathcal{E}_{\text{new}}), \mathcal{O}_{\text{obs}}) \ge \epsilon \end{aligned} \tag{5}$$

where $v_0 \in \mathcal{V}_{\text{prev}}$ is the parent vertex to be selected from the previous tree, $\{(x, y, \phi, \kappa)_i\}_{i=1}^{N}$ and $\{l_i\}_{i=1}^{N}$ are the new vertex configurations and new edge lengths to be optimized, $\mathcal{E}_{\text{root}\to N} = \mathcal{E}_{\text{root}\to 0} \cup \mathcal{E}_{\text{new}} \subseteq \mathcal{E}_{\text{next}}$ is the edge sequence from the root vertex to the new head vertex, $J_{\text{expand}}(v, \mathcal{E}) = J_{\text{vert}}(v) + \sum_{e \in \mathcal{E}} J_{\text{edge}}(e)$ is the objective function with the vertex cost $J_{\text{vert}} : \mathcal{V} \to \mathbb{R}$ and the edge cost $J_{\text{edge}} : \mathcal{E} \to \mathbb{R}$, $\kappa_{\max} \in \mathbb{R}_{>0}$ is the maximum curvature of the ego vehicle, $\epsilon \in \mathbb{R}_{>0}$ is the safety margin, and $\texttt{dist}(\mathcal{O}_{\text{ego}}(\mathcal{E}_{\text{new}}), \mathcal{O}_{\text{obs}})$ is the distance between the obstacles and the ego vehicle following the new branch. The compact set $\mathcal{O}_\star$ can take various representations such as ellipsoids, rectangles, capsules, or occupancy grids, among other possibilities. Since iteratively solving (5) is time-demanding, a data-driven decoder for the new branch generation is proposed in Section IV.

### D. Clothoid Path Selection

The clothoid path selection is relatively straightforward. Essentially, the terminal vertex is chosen from the expanded tree that minimizes the overall cost. This path selection cost is defined as $J_{\text{select}}(v) = J_{\text{terminal}}(v) + J_{\text{expand}}(v, \mathcal{E}_{\text{root}\to v})$ where $J_{\text{terminal}} : \mathcal{V} \to \mathbb{R}$ is the terminal state objective such as lane keeping and travel distance in the road Frenét frame. Driving examples show the path selection in Fig. 5.

## III. Clothoid Approximation

A fast and efficient approximation for (3) is presented in this section. Let us consider a clothoid with the arc length $l \in \mathbb{R}_{\ge 0}$ and endpoint configurations $(x_0, y_0, \phi_0, \kappa_0)$, $(x_1, y_1, \phi_1, \kappa_1)$. To facilitate further expressions, the tangent angle is reformulated as $\underline{\phi}(\eta) = \phi_0 + \frac{l\kappa_0}{2}(\eta + 1) + \frac{l(\kappa_1 - \kappa_0)}{8}(\eta+1)^2$ where $\eta := \frac{2s-l}{l} \in [-1, 1]$ is the normalized arc length. Given $(x, y, \phi, \kappa)_0$ and $\kappa_1$, the terminal heading angle is obtained as $\phi_1 = \phi_0 + \frac{l}{2}(\kappa_1 + \kappa_0)$, and the terminal position is obtained as $(x_1, y_1) = (x_0, y_0) + \frac{l}{2}\int_{-1}^{1}(\cos, \sin) \circ \underline{\phi}(\eta)\mathrm{d}\eta$. Since a closed-form solution of $(x_1, y_1)$ is not available, the Gauss-Legendre quadrature is employed.

The Gauss-Legendre quadrature approximates the definite integral by a weighted sum of function evaluations as

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \approx \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \frac{l}{2}\sum_{i=1}^{n} w_i \begin{bmatrix} (\cos \circ \underline{\phi})(\eta_i) \\ (\sin \circ \underline{\phi})(\eta_i) \end{bmatrix} \tag{6}$$

where $n \in \mathbb{N}$ is the number of nodes, $\{\eta_i\} \in (-1, 1)^n$ is the set of nodes, and $\{w_i\} \in \mathbb{R}_{>0}^{n}$ is the set of weights. The nodes are chosen as the roots of the degree $n$ Legendre polynomial satisfying $P_n(\eta_i) = 0$, and the weights are given by $w_i := \frac{2}{(1-\eta_i^2)(P_n'(\eta_i))^2}$ with $P_n' = \frac{\mathrm{d}P_n}{\mathrm{d}\eta}$. The quadrature (6) precisely approximates the end position $(x_1, y_1)$, but it does not provide a continuous representation $(x(s), y(s))$.

Thus, we develop a smooth representation of the curve. Similar to (6), we choose the parameterization:

$$\begin{bmatrix} \underline{x}^{\text{gl}}(\eta) \\ \underline{y}^{\text{gl}}(\eta) \end{bmatrix} := \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \frac{l}{2}\sum_{i=1}^{n} \underline{w}_i(\eta) \begin{bmatrix} (\cos \circ \underline{\phi})(\eta_i) \\ (\sin \circ \underline{\phi})(\eta_i) \end{bmatrix} \tag{7}$$

where $\underline{w}_i : [-1, 1] \to \mathbb{R}_{\ge 0}$ is the weight functions. From the fact that the quadrature weight representations $w_i \equiv \int_{-1}^{1} \ell_i(\eta)\mathrm{d}\eta$ are equivalent, $\underline{w}_i(\eta)$ can be expressed as

$$\underline{w}_i(\eta) = \int_{-1}^{\eta} \ell_i(\xi)\mathrm{d}\xi = \int_{-1}^{\eta} \prod_{j=1, j\ne i}^{n} \frac{\xi - \eta_j}{\eta_i - \eta_j}\mathrm{d}\xi \tag{8}$$

where $\{\ell_i(\cdot)\}_{i=1}^{n}$ is the degree $n-1$ Lagrange polynomials. Directly using (8), however, can be computationally demanding and numerically unstable, especially when $n$ is large.

Instead, we observe that the weight function (8) is a degree $n$ polynomial, which can be reconstructed by $n+1$ function evaluations. Then $\underline{w}_i(\eta)$ can be reformulated as

$$\underline{w}_i(\eta) = \sum_{j=0}^{n} w_{ij}\ell_j^{+}(\eta) = \sum_{j=0}^{n} w_{ij} \prod_{k=0, k\ne j}^{n} \frac{\eta - \eta_k}{\eta_j - \eta_k} \tag{9}$$

where $\{w_{ij}\}$ is the evaluated weights to be computed below and $\{\ell_j^{+}(\cdot)\}_{j=0}^{n}$ is the degree $n$ Lagrange polynomials with the initial node $\eta_0 = -1$. To efficiently obtain $\{w_{ij}\}$, we arrange the derivatives $\frac{\mathrm{d}}{\mathrm{d}\eta}\underline{w}_i(\eta_k) = \ell_i(\eta_k) \equiv \sum_{j=0}^{n} w_{ij}\frac{\mathrm{d}}{\mathrm{d}\eta}\ell_j^{+}(\eta_k)$, where $1 \le i \le n$ in rows and $0 \le k \le n$ in columns, organizing them into a matrix form $\big[\ell_i(\eta_k)\big] = \big[w_{ij}\big]\big[\frac{\mathrm{d}}{\mathrm{d}\eta}\ell_j^{+}(\eta_k)\big]$ with $\big[\ell_i(\eta_k)\big] \in \mathbb{R}^{n\times n+1}$, $\big[w_{ij}\big] \in \mathbb{R}^{n\times n}$, and $\big[\frac{\mathrm{d}}{\mathrm{d}\eta}\ell_j^{+}(\eta_k)\big] \in \mathbb{R}^{n\times n+1}$. The index $j = 0$ is omitted because $w_{i0} = 0$ are known. Since the differentiation matrix $\big[\frac{\mathrm{d}}{\mathrm{d}\eta}\ell_j^{+}(\eta_k)\big]$ has a closed-form solution and has a rank of $n$, the weights can be obtained by

$$\big[w_{ij}\big] = \texttt{pinv}\big(\big[\tfrac{\mathrm{d}}{\mathrm{d}\eta}\ell_j^{+}(\eta_k)\big]\big)\big[\ell_i(\eta_k)\big] \tag{10}$$

where $\texttt{pinv}(\cdot)$ is the pseudo-inverse operator. The nodes $\{\eta_i\}$ and weights $\{w_{ij}\}$ can be precomputed to reduce the real-time computation efforts. This approach ensures a smooth and continuous representation of clothoids while maintaining the accuracy of the Gauss-Legendre quadrature.

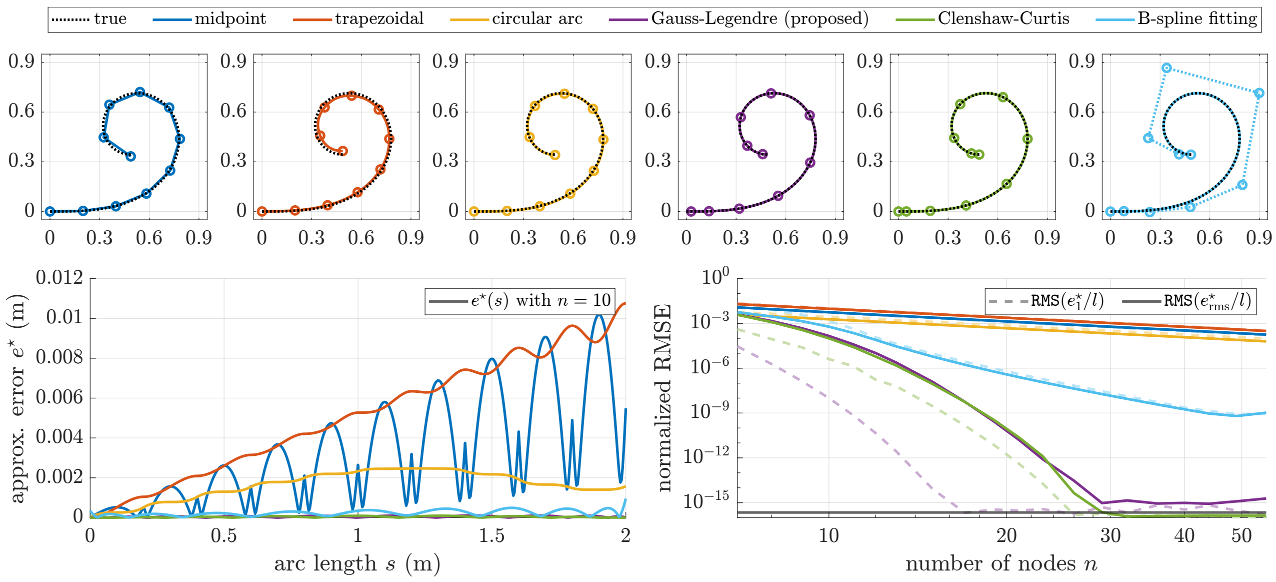


Fig. 3. Top. An example of clothoid approximations with $\kappa_0 = 0$, $\sigma = \pi$, and $n = 10$. Bottom-left. Approximation error $e^\star(s)$ with $\kappa_0 = 0$, $\sigma = \pi$, and $n = 10$. Bottom-right. Normalized root-mean-square of the endpoint errors $\texttt{RMS}(e_1^\star/l)$ (dashed lines) and smooth parameterization errors $\texttt{RMS}(e_{\text{rms}}^\star/l)$ (solid lines) with random curvatures $\kappa_0, \kappa_1 \in [-2\pi, 2\pi]$ and varying number of nodes $7 \le n \le 55$. The machine epsilon is approximately $2.22 \times 10^{-16}$.

### *Numerical Examples*

The approximation error is defined as the Euclidean distance $e^\star(s) := \|(x^\star(s), y^\star(s)) - (x(s), y(s))\|_2$ where $\star$ indicates the approximation method. The endpoint error is denoted by $e_1^\star = e^\star(l)$, and the root-mean-squared error (RMSE) of the smooth parameterization is denoted by $e_{\text{rms}}^\star = (\frac{1}{l}\int_0^l (e^\star(\xi))^2 \mathrm{d}\xi)^{1/2}$. The true clothoid $(x(s), y(s))$ is obtained by an adaptive quadrature until the estimated error falls below the machine epsilon.

The performance is compared to other numerical methods, including the midpoint and trapezoidal rules, piecewise circular arcs, the Clenshaw-Curtis quadrature, and quintic B-spline fitting. The midpoint and trapezoidal rules divide the curve into $n$ piecewise linear and quadratic functions, respectively. The circular arc approximation uses $n$ circular arcs, whose curvatures are determined based on the segment midpoint. The Clenshaw-Curtis quadrature is another class of numerical integration method known to converge at a rate similar to the Gauss-Legendre quadrature. Its smooth interpolation can be obtained by the Chebyshev polynomials of the first kind. The quintic B-spline fitting involves regression to the true clothoid data points rather than real-time curve integration. It optimizes $n$ control points minimizing $e_{\text{rms}}^{\text{bspline}}$.

Clothoid approximation results are visualized in Fig. 3. The initial position and heading are set to $(x_0, y_0, \phi_0) = 0$ because the shape of the curve is invariant to the translation and rotation, and the length is set to $l = 2$. Our proposed method achieves the smallest error, even outperforming the quintic B-spline fitting obtained by the time-consuming optimization. The smooth parameterization of the Clenshaw-Curtis quadrature is very close in performance to the proposed method. However, our proposed method has a much smaller endpoint error $e_1^{\text{gl}}$, which is crucial for the vertex position constraints. The endpoint error of the proposed method reaches the machine epsilon of the double precision data type at $n = 17$. It is worth noting that if we use (8) directly, the error $e_{\text{rms}}^{\text{gl}}$ could diverge at $n \ge 19$ due to the numerical instability. In terms of computational efficiency, the midpoint, trapezoidal, and circular arc methods necessitate more than $1 \times 10^7$ nodes for error saturation, resulting in computation times exceeding 0.4s per curve. Conversely, the proposed method maintains computational efficiency (e.g., requiring less than $5 \times 10^{-4}$s at $n < 30$) without compromising performance.

## IV. Interpolating Clothoid Spline Decoder

The interpolating clothoid spline decoder is detailed in this section. It generates clothoid splines that interpolate the parent and target configurations $(x, y, \phi, \kappa)_0$, $(x, y, \phi)_{\text{target}}$, relaxing the target constraint $(x, y, \phi)_N = (x, y, \phi)_{\text{target}}$ and reducing the problem size of (5). To simplify the problem, we first consider $N = 2$, and assume that the parent vertex configuration $(x, y, \phi, \kappa)_0$ is given. Then the clothoid spline between $v_0$ and $v_2$ can be fully described by the arc lengths $l_1, l_2$ and the curvatures $\kappa_1, \kappa_2$ using (1) to (3). Therefore, when $N = 2$, the number of continuous decision variables for the clothoid spline is 4, while the target constraint is three-dimensional. This implies that when the configurations $(x, y, \phi, \kappa)_0$ and $(x, y, \phi)_2 = (x, y, \phi)_{\text{target}}$ are given, the shape of two-segment clothoid spline $(\mathcal{V}_{\text{new}}, \mathcal{E}_{\text{new}})$ can be represented by a one-dimensional latent variable.

Here, we choose the ratio of the arc lengths $\gamma := \frac{l_1}{l_1 + l_2}$ as the latent variable, but generating the clothoid spline from $\gamma$ is non-trivial. Therefore, a data-driven two-segment interpolating clothoid spline decoder is designed as

$$D(\gamma \mid (x, y, \phi, \kappa)_0, (x, y, \phi)_2) = (l_{02}, \phi_1) \tag{11}$$

where $l_{02} = l_1 + l_2$ is the total arc length. Using the decoder output, the parameters $l_1, l_2, \kappa_1, \kappa_2$ can be computed analytically as $l_1 = \gamma l_{02}$, $l_2 = (1 - \gamma) l_{02}$, $\kappa_1 = \frac{\phi_1 - \phi_0}{l_1}$,

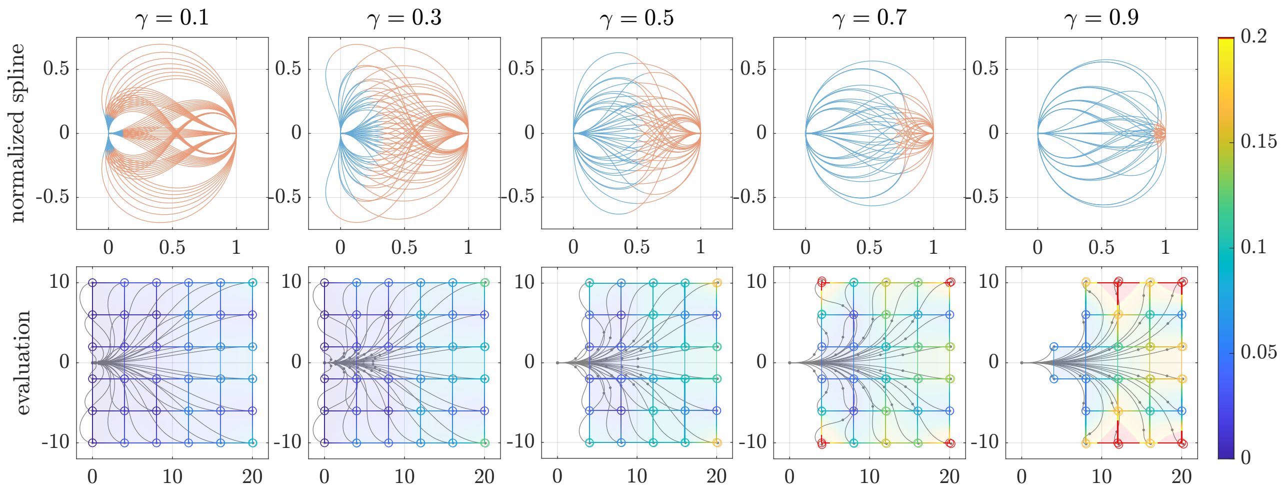


Fig. 4. Top. Normalized two-segment clothoid splines and their reflections with varying latent variable $\gamma \in (0,1)$. The light blue curve is the first segment, and the light orange curve is the second segment of the spline. Bottom. Evaluation of the interpolating clothoid spline decoder. Two-segment clothoid splines are generated by the decoder using the latent variable $\gamma$ and endpoint constraints. The evaluation error is depicted using a color map, with points exceeding an error of 0.2m indicated in red.

and $\kappa_2 = \frac{\phi_2 - \phi_1}{l_2}$. However, the decoder $D$ still requires high dimensional input data including $\gamma$, $(x, y, \phi, \kappa)_0$, and $(x, y, \phi)_2$. To reduce the complexity, we leverage the property that the shape of the curve remains invariant under translation, rotation, scale, and reflection transformations. To utilize the curve shape invariance, the normalized spline is defined by $(\overline{x}_0, \overline{y}_0) = (0, 0)$ and $(\overline{x}_2, \overline{y}_2) = (1, 0)$ with the following rotation and scale transformations:

$$\overline{\phi}_\star \equiv \phi_\star - \phi_{02} \pmod{2\pi}, \quad \overline{\kappa}_\star = \kappa_\star d_{02}, \quad \overline{l}_\star = \frac{l_\star}{d_{02}} \tag{12}$$

where $\star \in \{0, 1, 2\}$ is the vertex index of the clothoid spline, $\phi_{02} := \arctan(y_2 - y_0, x_2 - x_0)$ is the relative heading, and $d_{02} := \sqrt{(y_2 - y_0)^2 + (x_2 - x_0)^2}$ is the endpoint distance. The reflection over the $x$-axis further reduces the domain of the normalized spline by $(\overline{\phi}_\star, \overline{\kappa}_\star) \mapsto -(\overline{\phi}_\star, \overline{\kappa}_\star)$. Since the endpoint positions of the normalized splines are fixed, the decoder can be redefined as

$$\underline{D}(\gamma \mid \overline{\kappa}_0, \overline{\phi}_0, \overline{\phi}_2) = (\overline{l}_{02}, \overline{\phi}_1) \tag{13}$$

where $\underline{D}$ is the decoder for the normalized splines. From the decoder output $(\overline{l}_{02}, \overline{\phi}_1)$, the original clothoid spline can be reconstructed using the inverse of (12).

From the decoder (13), the tree expanding problem (5) can be approximated as

$$\begin{aligned} \underset{v_0 \in \mathcal{V}_{\text{prev}}, \gamma \in (0,1)}{\text{minimize}} \quad & J_{\text{expand}}(v_N, \mathcal{E}_{\text{root}\to N}) + J_D(v_N) \\ \text{subject to} \quad & |\kappa_i| \le \kappa_{\max} \\ & \texttt{dist}(\mathcal{O}_{\text{ego}}(\mathcal{E}_{\text{new}}), \mathcal{O}_{\text{obs}}) \ge \epsilon \end{aligned} \tag{14}$$

which selects the parent vertex $v_0 \in \mathcal{V}_{\text{prev}}$ and optimizes the arc length ratio $\gamma \in (0, 1)$. The decoder cost $J_D : \mathcal{V} \to \mathbb{R}$ is to evaluate the decoder model $\underline{D}$, because it may contain some error. The relaxation using the decoder can be extended to $N > 2$, but $N = 2$ suffices for our problem formulation. If the target curvature constraint $\kappa_N = \kappa_{\text{target}}$ is used in the tree expanding problem, $N = 3$ is required, and the latent variable becomes two-dimensional.

### *Numerical Examples*

Normalized two-segment clothoid splines can be generated by optimizing $\overline{l}_{02}, \overline{\phi}_1$ subject to given $\gamma$, $\overline{\kappa}_0$, $\overline{\phi}_0$, and $\overline{\phi}_2$. CasADi and IPOPT are employed here, while the optimization benefits by the proposed clothoid approximation (7), which offers small endpoint error with reasonable complexity. The data is gathered across a four-dimensional grid with $\gamma = 0.1, 0.2, \ldots, 0.9$, $\overline{\phi}_0 = 0, 0.1\pi, \ldots, 0.5\pi$, $\overline{\kappa}_0 = -2\pi, -1.9\pi, \ldots, 2\pi$, and $\overline{\phi}_2 = -0.5\pi, -0.4\pi, \ldots, 0.5\pi$ which has 24 354 data points in total. The reflection transformation reduces the data and the decoder domain to $\overline{\phi}_0 \ge 0$. See Fig. 4 for the example of the normalized clothoid splines and their reflections.

The input space of the decoder has been notably reduced by the spline normalization (12) and reflection, making it feasible to employ a linear interpolation of the grid data. Alternative methods, including regressions, Gaussian processes, and neural networks, remain available for future explorations. The performance of the interpolation-based decoder (13) is evaluated in Fig. 4. The decoder demonstrates an average error of $0.0048 \pm 0.0056$m over a one-meter path. Evaluations with a high arc length ratio $\gamma$ tend to exhibit large errors (e.g., around 0.28m over a 24.6m path). It occurs because the high $\gamma$ causes many endpoint constraints infeasible for two-segment clothoid splines, leading to limited available data. Moreover, longer clothoid splines amplify evaluation errors since the decoder is trained with normalized splines. To address the decoder error, the tree expansion problem (14) accounts for the decoder model cost, effectively preventing the selection of decoded splines with substantial errors.

On the other hand, using an inaccurate decoder, such as one with coarse grid data, can lead to increased deviations in the target node constraints. Then, the resulting tree may exhibit undesirable behaviors, including lane violations, and may not adequately explore the target configurations for effective path planning. Therefore, it is crucial to reduce the decoder error by employing a sufficient amount of data.

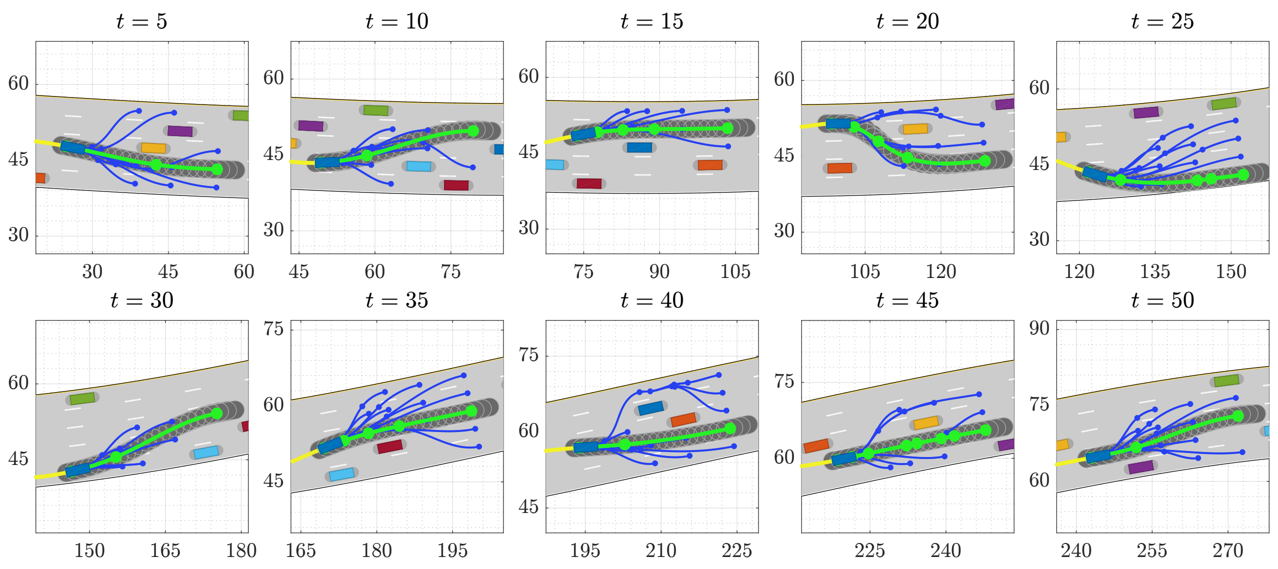


Fig. 5. Snapshots of the driving simulation. The proposed method selects the best path while considering the maximum curvature and the collision avoidance. The safety margin of the selected path is visualized by the capsules. The simulation video is provided at `https://youtu.be/g2UBF7bnQ74`

## V. Simulation Results

The proposed framework is validated using MATLAB on a Windows 10 machine with AMD Ryzen 5 3600X CPU and 16GB RAM. The driving scenario is generated by MATLAB Automated Driving Toolbox with static obstacles. The proposed algorithm is tested on various roads approximately $800$ to $900\mathrm{m}$ long and $18\mathrm{m}$ width with $5$ lanes. The ego vehicle maintains a constant speed $5\mathrm{m/s}$, and the maximum curvature of the ego vehicle is $0.3\mathrm{m}^{-1}$. For the tree expansion, the $20$ target configurations are drawn from the Frenét frame of the road, leading the clothoid tree to have maximum $41$ vertices. The relaxed MIP is solved in a grid-based manner, i.e., all possible combinations of the tail vertices and the latent variables $\gamma = 0.3, 0.4, \ldots, 0.7$ are tested. Capsules with a safety margin are employed for collision checks. The planning frequency is $10\mathrm{Hz}$. The planning algorithm has the potential for extensions to address dynamic obstacles, higher ego speeds, or the adaptation of driving strategies based on traffic conditions. However, these extensions lie beyond the focus of the paper.

As demonstrated in Fig. 5, our clothoid tree-based path planning successfully finds the path with $\mathrm{G}^2$ continuity and feasibility constraints. The tree shape varies by the road curvature and obstacles while incorporating the multi-modality of the driving scenarios. The resultant planning time without collision checks is $0.0179 \pm 0.0018\mathrm{s}$, and the planning time with collision checks is $0.0535 \pm 0.0214\mathrm{s}$. Note that the computational complexity of the algorithm solely depends on the number of target configurations and obstacles, not vehicle speed or road shape. Some prior works may find the feasible path faster, but they lack the ability to generate path candidates of varying shapes with proper geometric interpretations.

The cost functions are described below. The vertex cost is defined as $J_{\mathrm{vert}}(v) = \frac{k_{\mathrm{vert},\kappa}}{2}(g_\kappa(v) - \kappa_{\mathrm{target}})^2$ with $k_{\mathrm{vert},\kappa} = 1$ which tries to match the target curvature during the tree expansion. The target curvature $\kappa_{\mathrm{target}}$ is obtained from the road shape. The position and heading error are not considered due to the target constraint in (5). The edge cost is defined as $J_{\mathrm{edge}}(e) = \int_0^{g_l(e)} \big(\frac{k_{\mathrm{edge},\kappa}}{2}\kappa^2(s) + \frac{k_{\mathrm{edge},\sigma}}{2}\sigma^2(s)\big)\mathrm{d}s$ with $k_{\mathrm{edge},\kappa} = 1$ and $k_{\mathrm{edge},\sigma} = 5$ where $\kappa, \sigma : \mathbb{R}_{\geq 0} \to \mathbb{R}$ are the curvature and sharpness along the edge. The edge cost makes the curvature of each branch more uniform. The decoder evaluation cost is defined as $J_D(v) = \frac{k_D}{2}\|g_{(x,y)}(v) - (x,y)_{\mathrm{target}}\|^2$ with $k_D = 0.1$ which regulates the decoder model error. The decoder cost also does not consider heading error since $\phi_N = \phi_{\mathrm{target}}$ is ensured by $\underline{D}$.

## VI. Conclusion

In this paper, we propose a clothoid tree-based path planning for self-driving robots. The clothoid tree is a directed rooted tree, where each edge exhibits a linear curvature, and adjacent edges maintain $\mathrm{G}^2$ continuity. The clothoid tree is employed efficiently with the following two techniques: 1) the efficient and accurate clothoid approximation method that ensures a smooth curve representation; and 2) the data-efficient decoder that generates two-segment interpolating clothoid splines, improving the tree generation time. The clothoid approximation achieves an error of machine epsilon over a one-meter arc length. The interpolating clothoid spline decoder exhibits millimeter-level error over a one-meter path. In the driving simulations, the tree shape varies by the road curvature and obstacles, and the planning time remains $\leq$0.1s even with collision checks using capsules. Our approach contributes to curve approximation and path optimization methods, offering enhanced path planning and design possibilities for self-driving robots. Some possible future research directions are: considering dynamic obstacles and speed change of the ego vehicle; expanding the clothoid tree to different terrains and environmental conditions; incorporating the dynamics-based feasibility checks [20]; and real hardware experiments.

## REFERENCES

[1] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[2] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame," in *2010 IEEE International Conference on Robotics and Automation*, pp. 987–993, IEEE, 2010.

[3] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2012.

[4] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[5] D. J. Walton and D. S. Meek, "A controlled clothoid spline," *Computers & Graphics*, vol. 29, no. 3, pp. 353–363, 2005.

[6] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, vol. 55. US Government Printing Office, 1968.

[7] L. Wang, K. T. Miura, E. Nakamae, T. Yamamoto, and T. Wang, "An approximation approach of the clothoid curve defined in the interval $[0, \pi/2]$ and its offset by free-form curves," *Computer-Aided Design*, vol. 33, no. 14, pp. 1049–1058, 2001.

[8] J. Sánchez-Reyes and J. M. Chacón, "Polynomial approximation to clothoids via s-power series," *Computer-Aided Design*, vol. 35, no. 14, pp. 1305–1313, 2003.

[9] D. S. Meek and D. J. Walton, "An arc spline approximation to a clothoid," *Journal of Computational and Applied Mathematics*, vol. 170, no. 1, pp. 59–77, 2004.

[10] M. Brezak and I. Petrović, "Real-time approximation of clothoids with bounded error for path planning applications," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 507–515, 2013.

[11] Y. Chen, Y. Cai, J. Zheng, and D. Thalmann, "Accurate and efficient approximation of clothoids using Bézier curves for path planning," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1242–1247, 2017.

[12] D. K. Wilde, "Computing clothoid segments for trajectory generation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2440–2445, IEEE, 2009.

[13] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Bessler, and B. Huhnke, "Up to the limits: Autonomous Audi TTS," in *2012 IEEE Intelligent Vehicles Symposium*, pp. 541–547, IEEE, 2012.

[14] C. Alia, T. Gilles, T. Reine, and C. Ali, "Local trajectory planning and tracking of autonomous vehicles, using clothoid tentacles method," in *2015 IEEE Intelligent Vehicles Symposium*, pp. 674–679, IEEE, 2015.

[15] H. Mouhagir, R. Talj, V. Cherfaoui, F. Guillemard, and F. Aioun, "A Markov decision process-based approach for trajectory planning with clothoid tentacles," in *2016 IEEE Intelligent Vehicles Symposium*, pp. 1254–1259, IEEE, 2016.

[16] M. Frego, P. Bevilacqua, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, "Trajectory planning for car-like vehicles: A modular approach," in *2016 IEEE Conference on Decision and Control*, pp. 203–209, IEEE, 2016.

[17] E. Bertolazzi and M. Frego, "G1 fitting with clothoids," *Mathematical Methods in the Applied Sciences*, vol. 38, no. 5, pp. 881–897, 2015.

[18] M. Kim, J. Ahn, and J. Park, "TargetTree-RRT*: Continuous-curvature path planning algorithm for autonomous parking in complex environments," *IEEE Transactions on Automation Science and Engineering*, 2022.

[19] E. Bertolazzi and M. Frego, "Interpolating clothoid splines with curvature continuity," *Mathematical Methods in the Applied Sciences*, vol. 41, no. 4, pp. 1723–1737, 2018.

[20] S.-Y. Jeon, R. Chung, and D.-J. Lee, "Planned trajectory classification for wheeled mobile robots to prevent rollover and slip," *IEEE Robotics and Automation Letters*, 2023.