Continuous-Curvature Trajectory Planning

Submitted: 3rd May 2020; accepted: 14th March 2021

Jörg Roth

DOI: 10.14313/JAMRIS/1-2021/2

Abstract: *Continuous-curvature paths play an important role in the area of driving robots: as vehicles usually cannot change the steering angle in zero-time, real trajectories must not have discontinuities in the curvature profile. Typical continuous-curvature paths are thus built of straight lines, arcs and clothoids. Due to the geometric nature of clothoids, some questions in the area of trajectory planning are difficult the answer – usually we need approximations here. In this paper we describe a full approach for continuous-curvature trajectory planning for mobile robots – it covers a maneuver-based planning with Viterbi optimization and geometric approximations required to construct the respective clothoid trajectories.*

Keywords: *Mobile Robots, Trajectory Planning, Continuous-curvature Paths, Clothoids*

1. Introduction

Trajectory planning is a fundamental function of a mobile robot. When executing tasks such as transporting goods, the robot has to drive complex trajectories that meet certain measures of optimality. For this, a cost function may consider driving time, energy consumption, mechanical wear or buffer distance to obstacles.

Whereas a geometric route planning tried to find a line string with minimal costs that does not cut an obstacle (with respect to the robot's driving width), the trajectory planning also considers nonholonomic constraints such as curve angles or orientations. Non-holonomic constraints prohibit, e.g., to move sideways or to rotate the body while driving straight. An important constraint is related to curvature: real motion systems can change the steering angle and thus the current curvature only with finite speed. As a result, the change of, e.g., straight driving to driving a curve is not possible instantly. The key approach to create continuous-curvature trajectory sequences is to incorporate clothoid segments – they linearly change the curvature and thus are able to, e.g., connect straight driving and arc trajectories.

Fig. 1 shows an example: we see a trajectory sequence and the curvature κ of the driving distance s. Straight driving (L) have zero and arcs (A) have a constant non-zero curvature. Clothoids (C) have a constant derivate of κ. We are able to create sequences of these three primitive trajectories L, A, C without any discontinuity in curvature. We could think of more complex primitive trajectories that steadily change the curvature, however, the clothoid is the most 'natural' trajectory with this property.

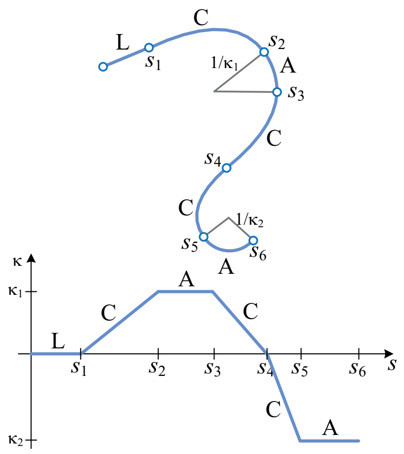


Fig. 1. A continuous-curvature trajectory sequence

We have to face the following challenges when we want to compute continuous-curvature paths based on clothoids:

- The planning complexity increases, when we increase the configuration dimensions. For continuous-curvature paths we have start, target and intermediate configurations with four parameters (x, y, θ, κ).
- We want to consider arbitrary cost functions, not only the length or maximum curvature. Ideally the cost function is represented as functional black box, queried at runtime to compute cost values. In addition, paths have to consider obstacles that already have been perceived by the robot.
- In contrast to straight driving and arcs, clothoids are not described by simple, closed formulas. For direct functions (e.g., the position for a driving distance), there exist well-known approximations. For reverse functions (e.g., given a position, what is the driving distance to this position) approximations are often not known.

Our approach to compute continuous-curvature paths covers the whole range from basic approximation functions up to the overall planning of trajectories. We consider arbitrary cost functions, passed as runtime function. We use a maneuver approach with Viterbi optimization to compute obstacle-free paths with least costs. We suggest different runtime-optimized approximations to compute the required clothoid functions.

2. Related Work

We can split related work in approaches that deal with the general problem of trajectory planning and specific problems related to continuous-curvature trajectory planning. A general approach may also create continuous-curvature paths, if the curvature is part of the configuration and the derivate of curvature can be considered in the planning algorithm.

Early work investigated shortest paths for vehicles that are able to drive straight forward and circular curves [3, 5, 18]. Without obstacles, we can connect two configurations with only three primitive trajectories, the so-called Dubins paths. Dubins path have discontinuities in the curvature as they instantly change between left and right arcs or arcs and linear driving.

More related to our approach is work that investigates longer paths that go through an environment of obstacles. As the space of possible trajectory sequences gets very large, probabilistic approaches are a suitable method to find at least a suboptimal solution [8, 10, 11]. They randomly connect configurations by primitive trajectories and are able to search on the respective graph to plan an actual path. Further work used potential fields [1] or visibility graphs [16]. With the help of geometric route planners, the overall problem of trajectory planning can be reduced. In [9], the route planning step and a local trajectory planning step are recursively applied.

Random sampling can also be used to improve generated trajectories. E.g., CHOMP [25] uses functional gradient techniques based on Hamiltonian Monte Carlo to iteratively improve the quality of an initial trajectory. The approach in [15] represented the continuous-time trajectory as a sample from a Gaussian process generated by a linear time-varying stochastic differential equation. Then gradient-based optimization technique optimizes trajectories with respect to a cost function.

Further planning approaches are based on the state lattice idea introduced in [17]. State lattices are discrete graphs embedded into the continuous state space. Vertices represent states that reside on a hyperdimensional grid, whereas edges join states by trajectories that satisfy the robot's motion constraints. The original approach is based on equivalence classes for all trajectories that connect two states and perform inverse trajectory generation to compute the result trajectory. [7] introduced a two-step approach, with coarse planning of states based on Dynamic Programming, and a fine trajectory planning that connects the formerly generated states.

Further work investigated the mathematics behind clothoids and characteristics of clothoids in paths. [2] discovered an important property: if we only consider the path length as costs, then optimal paths may have an infinite number of switching points, i.e., points that change between different primitive trajectories. Apart from trivial paths, sequences with minimal lengths may have an infinite number of clothoid arcs. This finding is important, because real paths (with a finite number of primitive trajectories) are thus usually not optimal. This is contrarily to Dubins paths, where we do not consider curvature. [2] does not consider obstacles or cost functions other than the length.

[24] minimized the maximum curvature when constructing clothoids. Up to a certain point their approach provided closed formulas, however, when considering the so-called Fresnel integral (see later), they switched to approximations with fractions of quadratic and cubic polynoms. With this, maneuvers of type CA (see later) can be computed.

[13] focused on certain geometric problems when construction clothoids, e.g., clothoids that connect circles. They provide approximation functions to compute clothoid parameters. [14] approximated clothoids by arc splines that are more suitable for some geometric questions. Instead of clothoids, also cubic spirals or splines were considered [4, 12].

[6] took existing planned trajectory sequences and converted them to continuous-curvature sequences. The input sequence only contains linear driving and turning in place. Their approach: each turn in place is transformed to a pair of clothoids that result in the same target configuration. This approach has a certain benefit: planning of the input trajectory sequence with the help of turn-in-place trajectories is very simple, thus this special continuous-curvature sequence can efficiently planned as well. However: replacing turn in place by clothoids may cause large loops that move the robot far away from the planned path. As a result, the robot may collide with obstacles.

[22] introduced so-called Simple Continuous Curvature Paths (SCC) and constructed every larger path from SCCs. SCCs contain 8 primitive trajectories. In our notion they are of type $CAC_{-0}LCAC_{-0}$ (see later).

An observation: related work either focused on short clothoid paths without the consideration of different cost functions, or tried to construct a complex paths of a small set of what we later call maneuvers. We strongly believe, the cost function may not only consider the length or maximum curvature, but may be more complex. In particular, we have to deal with obstacles and maybe want to integrate a safety distance to the cost function. We also believe that suitable paths may contain numerous combinations of primitive trajectories, not only a small set of maneuvers.

3. The Trajectory Planning Approach

3.1. The Planning Architecture

The goal is to provide a mechanism to move a robot from one configuration (x, y, θ, κ) to a target configuration, meanwhile holding the constraint of continuous curvature. We start with the overall architecture of the motion planning and execution (Fig. 2).

The application specifies the motion tasks to move to a certain target. A target may define only a position, but also a target orientation or curvature. The *Navigation* component provides a point-to-point route planning in the workspace. This component does not consider non-holonomic constraints. It computes a line string of minimal costs that avoids obstacles.

The *Trajectory Planning* computes a drivable sequence of trajectories between configurations and considers non-holonomic constraints. As the Navigation already detected a collision-free line string, this component does not have to check for *any* trajectory sequence between the configurations, but looks for trajectories that connect the turning points. This two-phase approach significantly reduces the overall computation complexity.

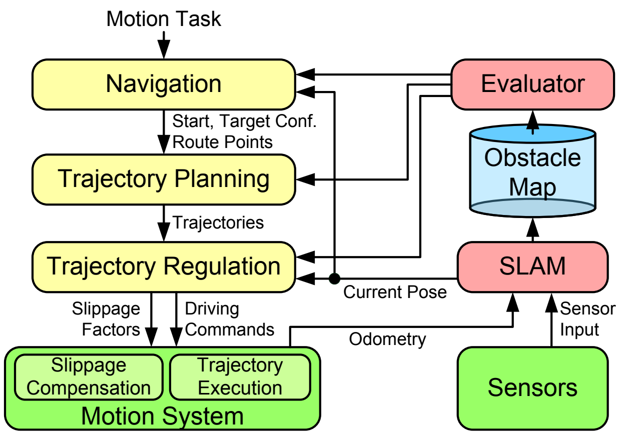


Fig. 2. Planning and execution data flow

Fig. 2. Planning and execution data flow

The *Evaluator* computes costs of routes and trajectories based on the obstacle map and the desired properties. Cost values may take into account the path length, expected energy consumption or the amount back-driving. Also, the distance to obstacles could be considered, if, e.g., we want the robot to keep a safety distance where possible. As an important decision: the planning approach is fully separated from the evaluation approach that even may change its behaviour at runtime.

The lower components are not focus of this paper: the *Trajectory Regulation* permanently tries to hold the planned trajectories, even if the position drifts off. *Simultaneous Localization and Mapping* (SLAM) constantly observes the environment and computes the most probable own location and location of obstacles by motion feedback and sensors (e.g., Lidar or camera). The current error-corrected configuration is passed to all planning components. Observed and error-corrected obstacle positions are stored in an *Obstacle Map* for further planning tasks. The *Motion System* finally is able to execute and supervise driving commands by formalized trajectories.

3.2. Trajectories and Maneuvers

We now assume the robot drives in the plane in a workspace $\mathcal{W}$ with positions (x, y). The configuration space $\mathcal{C}$ covers additional dimensions for orientation angle θ and curvature κ, i.e., our configuration space is of (x, y, θ, κ).

The goal is to find a collision-free sequence of trajectories that connects two configurations, meanwhile minimizes a cost function. The overall problem has many degrees of freedom. Whereas even small distances can be connected by an infinite number of trajectories, the problem gets worse for larger environments with many obstacles. We thus introduce the following concepts:

- A route planning solely operates on workspace $\mathcal{W}$ and computes a sequence of collision-free lines of sight (with respect to the robot's width) that minimize the costs.
- As the route planning only computes route points in $\mathcal{W}$, we have to specify additional values in $\mathcal{C}$. From the infinite assignments, we only consider a small finite set.
- From the infinite set of trajectories between two route configurations, we only consider a finite set of *maneuvers*. Maneuvers are trajectories, for which we know formulas that derive the respective parameters (e.g., curve radii) from start and target configurations.
- Even though these concepts reduce the problem space to a finite set of variations, this set would by far be too large for a complete iteration. We thus apply a Viterbi-like approach that significantly reduces the number of checked variations to find an optimum.

We carefully separated the cost function from all planning components. We assume there is a mapping from a route or trajectory sequence to a cost value according to two rules: first, we have to assign a single, scalar value to a trajectory sequence that indicates its costs. If costs cover multiple attributes (e.g., driving time and battery consumption), the cost function has to weight these attributes and create a single value. Second, a collision with obstacles has to result in infinite costs.

The basic capabilities of movement are defined by a set of *primitive trajectories*. The respective set can vary between different robots. E.g., the Carbot [21] is able to execute the following primitive trajectories:

- $L(\ell)$: linear (straight) driving over a distance ℓ;
- $A(\ell, r)$: circular arc with radius r over a distance ℓ;
- $C(\ell, \kappa_s, \kappa_t)$: clothoid over a distance ℓ with given start and target curvatures.

Further primitive trajectories may be possible, e.g., to turn in place, however, we are only interested in continuous-curvature trajectories in this paper.

We are able to map primitive trajectories directly to driving commands that are natively executed by the robot's motion system. Implicitly, they specify functions that map two configurations c_s to c_t. Due to non-holonomic constraints, it usually is not possible to reverse this mapping. I.e., for given $c_s, c_t \in \mathcal{C}$ there is in general no primitive trajectory that maps c_s to c_t.

At this point, we introduce *maneuvers*. Maneuvers are small sequences of primitive trajectories (usually up to 10 elements) that *are* able to map between given $c_s, c_t \in \mathcal{C}$. More specifically:

- A maneuver is defined by a sequence of primitive trajectories (e.g., denoted ALA or LCA) and further constraints. Constraints may relate or restrict the respective primitive trajectory parameters.
- For given $c_s, c_t \in \mathcal{C}$ there exist formulas that specify the parameters of the involved primitive trajectories, i.e., ℓ for L, A and C, r for A, κ_s, κ_t for C.
- Usually, the respective equations are underdetermined. As a result, multiple maneuvers of a certain type (sometimes an infinite number) map c_s to c_t. Thus, we need further parameters we call *free parameters* to get a unique maneuver.
- We further have maneuvers that do not consider all start and target configuration parameters. E.g., some maneuvers drive to a certain target position, but the target orientation cannot be specified beforehand. We call the specified configuration parameters for a maneuver *start* and *target parameters*. At least (x, y, θ) must be start parameters and workspace dimensions (x, y) must be target parameters. In addition, continuous-curvature maneuvers must accept (x, y, θ, κ) as start parameters.

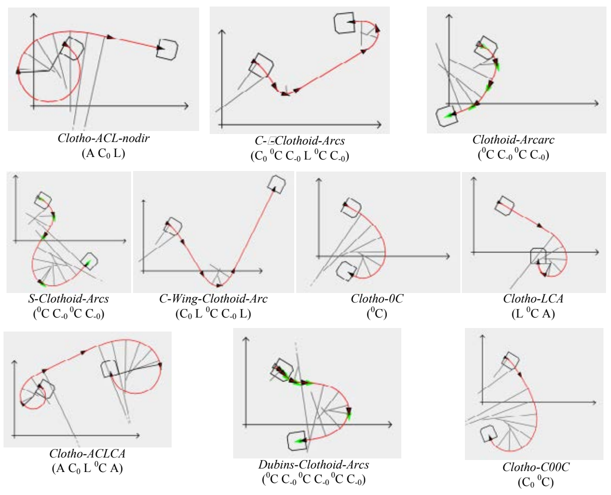


Fig. 3. Sample continuous-curvature maneuvers

Until now we identified about thirty maneuvers. Fig. 3 shows ten of them that support continuous-curvature paths. The challenge is to set up formulas for the trajectory parameters. For the non-continuous curvature planning, the formulas usually are a result of solving a linear or quadratic equation system – for all we get closed solutions. For continuous-curvature planning, however, we have clothoid functions that results in approximations. We come back to this point later.

3.3. Notation of Maneuvers

In order to systematically describe the numerous maneuvers, we introduced a special notation. This is not only useful to precisely describe the maneuver's structure to developers – is can be used to check basic properties without to know the underlying formulas. These checks cover:

- Checks when constructing maneuvers, whether they are properly defined and integrated into the overall planning system.
- Checks that support the developer of a robot-application to select appropriate maneuvers for a certain situation or vehicle.
- Runtime checks, whether two subsequent maneuver *match*, i.e., can be driven after each other, while certain driving properties are fulfilled.

For this, we developed the notion of *structure patterns*. They are built from the primitive trajectory (A, L and C) and modifiers that set a primitive trajectory in relation to its neighbour. Table 1 shows all elements. For A we introduced the modifiers $A_\sim$ and $A_=$ that relate the arc's turning orientation to a former arc. This is useful to indicate maneuvers that first drive a right curve, then left curve or vice versa ($A_\sim$) or twice right or twice left curve ($A_=$).

For clothoids we know more modifiers. First we may indicate a zero-curvature at start (^0C) or termination (C_0). Second, we may indicate a symmetric clothoid as described later in section 4.2 (C_-) that also may have zero-curvature at termination (C_0).

Tab. 1. Maneuver notation elements

Notation	Meaning
L	Linear
A	Arc
$A_\sim$	Arc with opposite turning orientation to last A
$A_=$	Arc with same turning orientation to last A
C	Clothoid
^0C	Clothoid with zero start curvature
C_0	Clothoid with zero end curvature
C_-	Exact symmetric clothoid to last C
C_{-0}	Exact symmetric clothoid to last C with zero end curvature

The maneuvers we found so far are listed in Table 2. Besides the structure patterns, the table presents start, target and free parameters. For start parameters we can distinguish the following cases:

- (x, y, θ): maneuvers that do not predefine a start curvature: such maneuvers usually change the curvature at start, thus are not suitable for continuous curvature planning;
- $(x, y, \theta, 0)$: maneuvers that assume a zero-curvature at start: such maneuvers can continue a trajectory that ends with zero-curvature;

Tab. 2. Maneuvers

Name	Structure Pattern	Start Params	Target Params	Free Params	Continuous Curvature	Suitable for Continuous Curvature Planning
C-Bow	A	(x, y, θ)	(x, y)	-	yes	
Clothoid-Bow	C C$_{-}$	(x, y, θ)	(x, y)	-	yes	
J-Bow	L A	$(x, y, \theta, 0)$	(x, y, θ)	-		
J-Clothoid-Bow	L ^{0}C C$_{-0}$	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	-	yes	yes
C-J-Clothoid-Bow	C$_0$ L ^{0}C C$_{-0}$	(x, y, θ, κ)	$(x, y, \theta, 0)$	ℓ of 1st C	yes	yes
J-Bow2	A L	(x, y, θ)	$(x, y, \theta, 0)$	-		
J-Clothoid-Bow2	^{0}C C$_{-0}$ L	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	-	yes	yes
C-J-Clothoid-Bow2	C$_0$ ^{0}C C$_{-0}$ L	(x, y, θ, κ)	$(x, y, \theta, 0)$	ℓ of 1st C	yes	yes
∫-Arcs	A L A	(x, y, θ)	(x, y, θ)	r_1 of 1st arc, r_2 of 2nd A		
∫-Clothoid-Arcs	^{0}C C$_{-0}$ L ^{0}C C$_{-0}$	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	r_1 of 1st A, r_2 of 2nd A (*)	yes	yes
C-∫-Clothoid-Arcs	C$_0$ ^{0}C C$_{-0}$ L ^{0}C C$_{-0}$	(x, y, θ, κ)	$(x, y, \theta, 0)$	ℓ of 1st C, r_1 of 1st A, r_2 of 2nd A (*)	yes	yes
Arcarc	A A$_{=}$	(x, y, θ)	(x, y, θ)	r_1 of 1st A, $r_{2\min}$... $r_{2\max}$ of 2nd A		
Clothoid-Arcarc	^{0}C C$_{-0}$ ^{0}C C$_{-0}$	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	r_1 of 1st A, $r_{2\min}$... $r_{2\max}$ of 2nd A (*)	yes	yes
S-Arcs	A A$_{\sim}$	(x, y, θ)	(x, y, θ)	-		
S-Clothoid-Arcs	^{0}C C$_{-0}$ ^{0}C C$_{-0}$	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	-	yes	yes
Wing-Arc	L A L	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	r of A		
Wing-Clothoid-Arc	L ^{0}C C$_{-0}$ L	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	r of A (*)	yes	yes
C-Wing-Clothoid-Arc	C$_0$ L ^{0}C C$_{-0}$ L	(x, y, θ, κ)	$(x, y, \theta, 0)$	ℓ of 1st C, r of A (*)	yes	yes
Snake	A A$_{\sim}$ L	(x, y, θ)	$(x, y, \theta, 0)$	r of 1st A (=r of 2nd A)		
Snake-Clothoid-Arcs	^{0}C C$_{-0}$ ^{0}C C$_{-0}$ L	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	r of 1st A (=r of 2nd A) (*)	yes	yes
Snake2	L A A$_{\sim}$	$(x, y, \theta, 0)$	(x, y, θ)	r of 1st A (=r of 2nd A)		
Snake2-Clothoid-Arcs	L ^{0}C C$_{-0}$ ^{0}C C$_{-0}$	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	r of 1st A (=r of 2nd A) (*)	yes	yes
Dubins-Arcs	A A$_{\sim}$ A$_{\sim}$	(x, y, θ)	(x, y, θ)	r of 1st A (=r of 2nd and 3rd A)		
Dubins-Clothoid-Arcs	^{0}C C$_{-0}$ ^{0}C C$_{-0}$ ^{0}C C$_{-0}$	$(x, y, \theta, 0)$	$(x, y, \theta, 0)$	r of 1st A (=r of 2nd and 3rd A) (*)	yes	yes
Clotho-0C	^{0}C	$(x, y, \theta, 0)$	(x, y)	-	yes	yes
Clotho-C00C	C$_0$ ^{0}C	(x, y, θ, κ)	(x, y)	ℓ of 1st C	yes	yes
Clotho-C	C	(x, y, θ, κ)	(x, y)		yes	yes
Clotho-CL	C$_0$ L	(x, y, θ, κ)	$(x, y, 0)$		yes	yes
Clotho-ACL-nodir	A C$_0$ L	(x, y, θ, κ)	$(x, y, 0)$		yes	yes
Clotho-ACL	A C$_0$ L	(x, y, θ, κ)	$(x, y, \theta, 0)$		yes	yes
Clotho-LCA	L ^{0}C A	$(x, y, \theta, 0)$	(x, y, θ, κ)		yes	yes
Clotho-CLCA	C$_0$ L ^{0}C A	(x, y, θ, κ)	(x, y, θ, κ)	ℓ of 1st C	yes	yes
Clotho-ACLCA	A C$_0$ L ^{0}C A	(x, y, θ, κ)	(x, y, θ, κ)		yes	yes

- (x, y, θ, κ): maneuvers that may continue any trajectory that ends with an arbitrary non-zero curvature.

For target parameters we distinguish (x, y), (x, y, θ), $(x, y, 0)$, $(x, y, \theta, 0)$, and (x, y, θ, κ). All are suitable for *inner* maneuvers (i.e., accept for the last), even for continuous-curvature planning. For the last maneuver it depends, whether the application requires that the robot holds a certain orientation and/or curvature at the target position. E.g., there may be a charging station that only can be used with a certain robot orientation. It is not likely for an application to request a target curvature. However, it could, e.g., be required that the steering angles of the wheels must be zero when entering the charging station.

We have manifold free parameters. Most are arc radii and clothoid lengths. Some free parameters are marked with (*): these are a result of transforming non-clothoid maneuvers as described in section 4.2.

The penultimate column of Table 2 indicates whether the connected primitive trajectories have a continuous-curvature. These are single trajectory maneuvers (e.g., only an A) or each pair of subsequent trajectories are of AC, AC$_0$, L^{0}C, C$_0$L, C$_{-0}$L, CC$_{-}$, ^{0}CC$_{-0}$, or C$_0$^{0}C. For continuous-curvature planning (last column) we further require both to have a continuous-curvature maneuver and start parameters that cover the curvature (0 or κ).

3.4. Finding Optimal Maneuver Sequences

Our planning approach both covers non-continuous and continuous-curvature trajectory planning. Here, we describe the latter case. We start from a configuration $(x_s, y_s, \theta_s, \kappa_s)$ and plan to a target $(x_t, y_t, \theta_t, \kappa_t)$, whereas θ_t and/or κ_t may also be * for '*any target orientation or curvature (resp.) may be suitable*'.

Let Π denote the set of maneuvers that are suitable for continuous-curvature planning as described in Table 2. Let $P(M)$ for an $M \in \Pi$ be the start, target and free parameters. $M(p)$ for $p \in P(M)$ denotes a concrete maneuver, i.e., a specific sequence of trajectories between given configurations. Consider $M(p)$ to be a maneuver instance according to a maneuver class M.

Let $(x_s, y_s) = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) = (x_t, y_t)$ denote a collision-free route found by the Navigation component (Fig. 2). Our problem is to find a *valid* sequence $(M_i(p_i))$ of maneuvers. Valid means:

- $M_1(p_1)$ starts with the configuration $(x_s, y_s, \theta_s, \kappa_s)$ and $M_n(p_n)$ terminates with $(x_s, y_s, \theta_s, \kappa_s)$;
- $M_i(p_i)$ connect the positions (x_i, y_i), (x_{i+1}, y_{i+1});
- $M_i(p_i)$ and $M_{i+1}(p_{i+1})$ match at the connection point (x_{i+1}, y_{i+1}), i.e., they have the same orientation θ_{i+1} and curvature κ_{i+1}.

From all valid sequences, we look for an optimal sequence, whereas optimality was assessed by the Evaluator component (Fig. 2) that provides an evaluation function $costs(M_i(p_i))$. The problem is a typical optimization problem: we look for

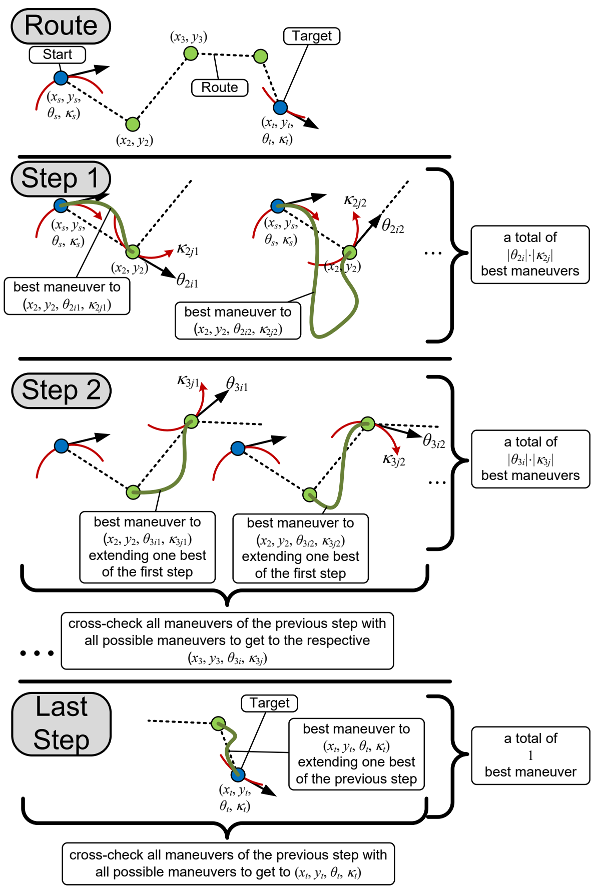
$$M_i, p_i = \underset{\substack{(M_j) \in \Pi^{n-1}, p_j \in P(M_j), \\ (M_i(p_j)) \text{ is valid}}}{\arg\min} costs(M_j(p_j)) \qquad (1)$$

The set Π is finite, moreover small, thus we could consider to iterate over all $(M_j) \in \Pi^{n-1}$. However, even for small n, the total number of possible combinations is getting very large. E.g., for $|\Pi| = 22$ (Table 2) and $n = 10$ route points we get more than a billion possible sequences of (M_j). Maneuver parameters (e.g., orientation and curvature of inner routing points) are even more crucial, as $P(M_j)$ are continuous values with a large number of dimensions. Moreover, there is no close relation of nearby values of $P(M_j)$ and the resulting *costs* value, i.e., even slightest modifications of parameters may significantly change the costs. Thus, optimization approaches as hill climbing or simulated annealing will fail.

Our approach [19] is inspired by the Viterbi algorithm [23]. The basic ideas:

- Of the infinite number of maneuver parameters we define a finite set of promising candidates. This obviously leads to sub-optimal results, however, opens the possibility to use a discrete optimization approach.
- We do not try to optimize all maneuvers at once, what would lead to a huge number of variations. Instead, we locally optimize *two* maneuvers that are connected by a route point.

This approach is suitable, because optimal paths have a characteristic: the interference between two trajectories in that path depends on their distance.



Route
Start
$(x_s, y_s, \theta_s, \kappa_s)$
(x_2, y_2)
(x_3, y_3)
Route
Target
$(x_t, y_t, \theta_t, \kappa_t)$
Step 1
κ_{2j1}
θ_{2i1}
best maneuver to $(x_2, y_2, \theta_{2i1}, \kappa_{2j1})$
κ_{2j2}
θ_{2i2}
best maneuver to $(x_2, y_2, \theta_{2i2}, \kappa_{2j2})$
a total of $|\theta_{2i}|\cdot|\kappa_{2j}|$ best maneuvers
Step 2
κ_{3j1}
θ_{3i1}
θ_{3i2}
κ_{3j2}
best maneuver to $(x_2, y_2, \theta_{3i1}, \kappa_{3j1})$ extending one best of the first step
best maneuver to $(x_2, y_2, \theta_{3i2}, \kappa_{3j2})$ extending one best of the first step
a total of $|\theta_{3i}|\cdot|\kappa_{3j}|$ best maneuvers
cross-check all maneuvers of the previous step with all possible maneuvers to get to the respective $(x_3, y_3, \theta_{3i}, \kappa_{3j})$
Last Step
Target
best maneuver to $(x_t, y_t, \theta_t, \kappa_t)$ extending one best of the previous step
a total of 1 best maneuver
cross-check all maneuvers of the previous step with all possible maneuvers to get to $(x_t, y_t, \theta_t, \kappa_t)$

Fig. 4. Idea of Viterbi optimization of trajectories

If they are close, a change of one usually also causes a change of the other. If they are far, we may change one trajectory of the sequence, without affecting the other. Viterbi reflects this characteristic, as it checks all combinations of neighbouring (i.e., close) maneuvers to get the optimum.

Fig. 4 illustrates the approach. We begin with the route points discovered by Navigation component. Step 1 varies orientation θ_2 and curvature κ_2 of route point (x_2, y_2). For each variation, we look for the best maneuver to get there – for this we vary the maneuvers and its parameters. As a result we get $|\theta_{2i}|\cdot|\kappa_{2j}|$ best maneuvers.

Step 2 is the pattern for all further steps. Now, all best maneuvers that are collected so far are combined with all matching maneuvers to get to (x_3, y_3), meanwhile varying θ_3 and κ_3. For short a time the algorithm deals with $|\theta_{2i}|\cdot|\kappa_{2j}|\cdot|\theta_{3i}|\cdot|\kappa_{3j}|$ permutations. However of these, we only keep the best one for each θ_3, κ_3. As a result, step 2 terminates with $|\theta_{3i}|\cdot|\kappa_{3j}|$ best maneuver sequences.

We iterate through all inner route points, meanwhile always keeping a set of current best maneuvers. If the target configuration contains orientation and curvature, the last step is different, as we do not have to vary them. Thus, only a single best maneuver remains. If one of orientation and curvature are *, then the respective candidate set as selected for inner route points is checked.

We finally have to clarify the candidate sets for maneuver parameters:

- For target orientations we consider variations of angles from the previous and to the next route point. This is suitable, as optimal trajectories mainly follow the basic route directions.
- For arc radii we consider some multipliers of the minimum curve radius r_{min}, e.g., $\{r_{min}, 3\cdot r_{min}, 5\cdot r_{min}\}$.
- For target curvatures we consider zero and reciprocals of candidates for arc radii.
- Candidates for clothoid lengths are computed from the vehicle's maximum curvature change and candidates for curvatures.

Note that configuration parameters that depend on other maneuver parameters implicitly define candidates for a certain step. E.g., Clotho-0C does not accept target orientation and curvature. This means, these values depend on other parameters such as start and target position. In particular, these parameters are *not* free. If in one step, such maneuvers are selected as best maneuver, the respective parameters are implicit candidates for the next step, even though not mentioned in the list above.

4. Clothoid Computations

Continuous-curvature paths require a) primitive trajectories with a continuous curvature and b) no discontinuity between two subsequent primitive trajectories. The curvature κ is defined as change of heading angle φ over running length s, i.e.,

$$\kappa = \frac{d\varphi}{ds} \tag{2}$$

This means, for linear trajectories we have $\kappa = 0$ and for arcs $\kappa = 1/r$. To connect these trajectories with constant curvature we further need a type with changing curvature. Of the infinite possible trajectories with changing curvature, *clothoids* play an important role: they have a constant change of curvature over running length, i.e.,

$$\kappa(s) = \kappa_0 + \frac{s}{\ell}\Delta\kappa \tag{3}$$

for a trajectory length of ℓ, curvature κ_0 for $s = 0$ and curvature $(\kappa_0 + \Delta\kappa)$ for $s = \ell$. Many formulas for clothoids request $\kappa = 0$ for $s = 0$, i.e.,

$$\kappa(s') = \frac{s'}{\ell}\Delta\kappa \tag{4}$$

To apply (3) instead of (2) we have to use $s' = s + s_0$ with $s_0 = \ell\kappa_0/\Delta\kappa$. As a result, we can use the simple formula (3) and just have to increment the running length s by s_0.

4.1. Direct Clothoid Functions

Direct clothoid functions map a running length s to position (x, y), curvature κ and heading angle ϕ. Positions are based on so-called the *Fresnel integrals*

$$C(s) = \int_0^s \cos(t^2)dt,\ S(s) = \int_0^s \sin(t^2)dt \tag{5}$$

For the *normalized Euler spiral* we set $x(s) = C(s)$ and $y(s) = S(s)$. To produce a clothoid with a certain curvature $\Delta\kappa$ at running length ℓ, we need a factor a

$$a = \sqrt{\frac{\Delta\kappa}{2\ell}} \tag{6}$$

Then

$$\begin{aligned} x(s) &= \frac{C(a\cdot s)}{a},\ y(s) = \frac{S(a\cdot s)}{a} \\ \kappa(s) &= 2a^2\cdot s,\quad \varphi(s) = a^2\cdot s^2 \end{aligned} \tag{7}$$

Fig. 5 shows a clothoid and its values. At a certain point we have a heading angle and a curvature. We may consider the trajectory at this point as an infinitely small arc with radius $1/\kappa$.

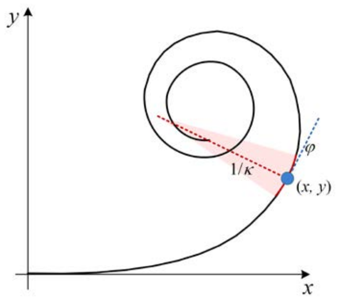


Fig. 5. A clothoid

Only the formulas for κ and φ can easily be inverted. Moreover, for $C(s)$ and $S(s)$ we only know approximations

$$\begin{aligned} C(s) &= s - \frac{s^5}{5\cdot 2!} + \frac{s^9}{9\cdot 4!} - \frac{s^{13}}{13\cdot 6!} + \ldots \\ S(s) &= \frac{s^3}{3\cdot 1!} - \frac{s^7}{7\cdot 3!} + \frac{s^{11}}{11\cdot 5!} - \ldots \end{aligned} \tag{8}$$

hat do not easily support inverse questions. E.g., for given x, y, it is not obvious to get s, a of a clothoid to get there. The following sections show how to construct maneuvers with clothoids and how to compute clothoid parameters.

4.2. Replacing Arcs by Symmetric Clothoids

A first idea to construct continuous-curvature maneuvers: we take a non-continuous maneuver and replace each arc by two symmetric clothoids ($^0C\ C_{-0}$). The replacement starts and terminates with zero curvature, whereas both clothoids are connected by the same non-zero curvature. As an example, we can replace the J-Bow (L A) by J-Clothoid-Bow (L $^0C\ C_{-0}$). If the original maneuver had an arc radius as free parameter, we consider this radius also as free parameter for the new maneuver, even though no arc appeared in the maneuver. This was indicated by (*) in Table 2.

We assume an arc that starts in x-direction, i.e., the arc centre resides on the y-axis. For the general case we have to roto-translate our approach to any location/orientation. We want to replace the arc with angle α by two symmetric clothoids. This means, each clothoid spans an angle of $\alpha/2$ viewed from the arc centre. This also must be the heading angle at the end of the first clothoid (Fig. 6).

Of the clothoid we do not know the respective a. However, we can benefit from a property of clothoids: they only differ in their scaling, but have, apart from this, identical shapes. Our approach to compute a is as follows:

- We first construct a clothoid with $a' = 1/\sqrt{2}$.
- We compute s' to get to $\varphi(s') = \alpha/2$.
- We compute (x', y') for s'.
- We add a symmetric clothoid and get to (t'_x, t'_y).
- We compute an f with $(t_x, t_y)=f(t'_x, t'_y)$ for the former arc termination (t_x, t_y).
- We finally get $a=1/\sqrt{2f}$.

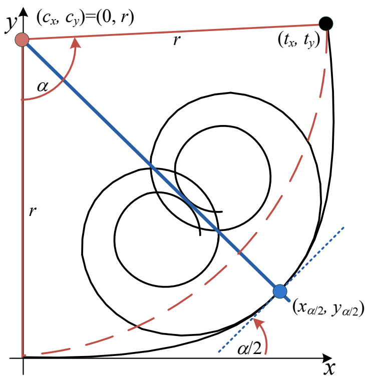


Fig. 6. Replacement of arc by two clothoids

The idea behind this: we first create a 'test' clothoid and append the symmetric clothoid. Typically, we will not get to the desired end point. But from the nature of all clothoids, the position is only wrong by a scaling factor. When we applied the scaling f, we finally get the respective a.

4.3. Maneuvers with Leading Clothoid

A second approach to create continuous-curvature maneuvers is to add a leading clothoid of type C_0. An existing continuous-curvature maneuver that starts with zero curvature (i.e., L or ^0C) can be transformed to a maneuver that starts with any non-zero curvature. The example is ʃ-Clothoid-Arcs (^0C C_0 L ^0C C_{-0}) that is extended to C-ʃ-Clothoid-Arcs (C_0 ^0C C_0 L ^0C C_0).

The construction is as follows: we construct a clothoid that starts with the start curvature and ends with zero curvature in any point. As the original maneuver already was able to connect any start point with any target point, it also is possible to choose the first clothoid's end point as alternative start. However, we get an additional free parameter ℓ – the first curvature's length.

With the methods in section 4.2 and this section we made use of direct clothoid computations as shown in formula (7) and only modified or extended existing maneuvers. However, we also wish to directly create continuous-curvature maneuvers. For this, we need to invert clothoid functions. The following sections deal with this problem. For most questions related to inverse clothoid functions, we do not know closed formulas. We thus developed numerous approximations to quickly compute such functions in the context of trajectory planning.

4.4. Running Length and Distance to Given Point

This next question is not actually related to maneuver construction, but it is heavily used during plan execution: given a point (p_x, p_y) and a clothoid defined by a. What is the running length s to get to (p_x, p_y)? As a clothoid may not exactly go through (p_x, p_y), we extend the question: what is the running length s of the nearest clothoid point? We need this function in the context of motion regulation [20] when we want to check if the robot still drives on the planned route and if not, what is the current deviation.

Our approach to compute this function is as follows:

- We first only consider the standard clothoid $a = 1$. A clothoid is infinitely twisted, but we stop after a reasonably running length in the context of trajectory planning, e.g., $s = \sqrt{(2\pi)}$.
- We pre-compute a grid (Fig. 7 left): for each grid centre we store the running length s of the nearest clothoid point. A grid of 20×20 is sufficient.
- For a position (p_x, p_y) we look for the cell with the nearest cell centre to (p_x, p_y). If the position was outside the grid, we use a projection into the grid.
- The running length s retrieved from the grid is only a rough approximation. We thus need a correction step.

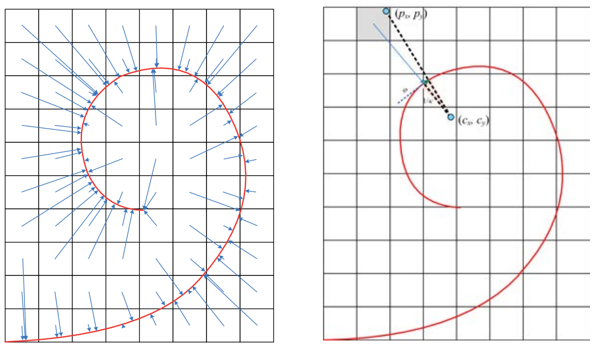


Fig. 7. Getting the nearest clothoid point

To pre-compute the grid, we iterate through s in a fine-grained manner, apply formula (7) and memorize in the grid cells, whether a new clothoid point is closer than formerly computed clothoid points. This pre-computation usually costs only some seconds.

When using the grid at runtime, we have to face the case when a given position is outside. For each of the major direction (e.g., x too large) we assume a virtual clothoid centre and linearly project (p_x, p_y) to border grid cell.

For a certain cell, we ask for the respective running length s' of nearest clothoid point. To get a more precise s we proceed as follows (Fig. 7 right):

- From s' we compute $\varphi(s')$ and $\kappa(s')$ and construct an arc with centre (c_x, c_y) and radius $1/\kappa(s')$. This arc approximates the clothoid in the area of s'.
- We cut the arc with the line $(c_x, c_y) - (p_x, p_y)$. From this we can compute a more precise s.
- We can repeat this to improve the precision, but a single iteration usually is sufficient.

Until now, we assume a=1. For any other value, we perform the steps above for $(a{\cdot}p_x, a{\cdot}p_y)$ to get the respective running length s/a.

4.5. Clothoid Through a Given Point with Zero Start Curvature

The next question is: given a (p_x, p_y) what is the a of a clothoid and running length s to go to this point? We need the answer to construct the Clotho-0C maneuver (^{0}C). Even though we have two equations and two variables, the respective equation system contains the functions C and S that prohibit common techniques to solve equations. We thus reduce the equations with a single function Q that we numerically invert. We define

$$Q(x)=\frac{S(x)}{C(x)} \tag{9}$$

We then are able to express the ratio of p_y and p_x as

$$\frac{p_y}{p_x}=\frac{S(a\cdot s)}{C(a\cdot s)}=Q(a\cdot s) \tag{10}$$

From

$$a\cdot s=Q^{-1}\frac{p_y}{p_x},\quad p_x=\frac{C(a\cdot s)}{a} \tag{11}$$

we get

$$a=\frac{C(a\cdot s)}{p_x}=\frac{C\left(Q^{-1}\frac{p_y}{p_x}\right)}{p_x} \tag{12}$$

and finally

$$s=\frac{Q^{-1}\frac{p_y}{p_x}}{a} \tag{13}$$

As a result, we now 'only' have to invert Q. A first problem: Q is periodic, thus we can only invert from 0 to the first maximum (Fig. 8). For a standard-clothoid (a = 1) we only convert up to a running length from 0 to approx. 2.04 (represents 75% of the spiral turn). This is sufficient for trajectory planning requests.

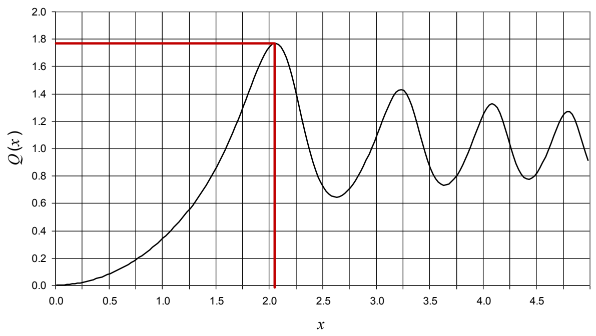


Fig. 8. Q function

Second, we have to numerically invert Q. As Q has a simple shape, we can use simple techniques. E.g., we may sample some hundreds values of Q as pre-computed data points and interpolate to compute any value of Q^{-1}.

4.6. Clothoid Through a Given Point with Non-Zero Start Curvature

The approach in the section 4.5 is only applicable for start curvature zero. In this section we assume a non-zero start curvature. We need this approach to construct the Clotho-C maneuver (C). We assume a clothoid as presented in formula (3) with start curvature κ_0. We have two cases (Fig. 9).

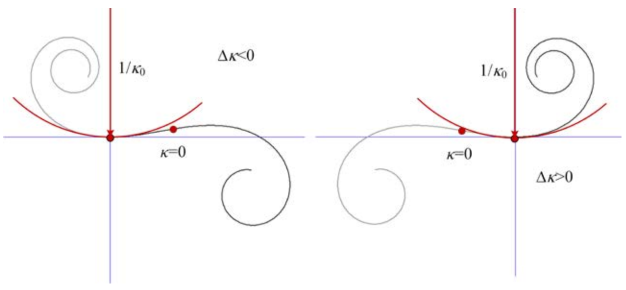


Fig. 9. Two cases of clothoids with start curvature

For a certain κ_0 and value a we have two major ways to create a clothoid, dependent on the sign of $\Delta\kappa$. It distinguishes, whether the turning point κ=0 appears for increasing or decreasing running length s. For a given (p_x, p_y) we are thus looking for value a, s and the sign of $\Delta\kappa$ of a clothoid that goes through the point.

Our approach to compute these values is as follows:

- We only consider the case κ_0=1 and pre-compute a grid.
- In the pre-computation phase we construct sample clothoids for different values of a and the two cases for the signs of $\Delta\kappa$. For each clothoid we iterate through s in a fine-granular manner.
- Each grid element holds the parameters of the clothoid that was the nearest passing the grid centre (Fig. 10).

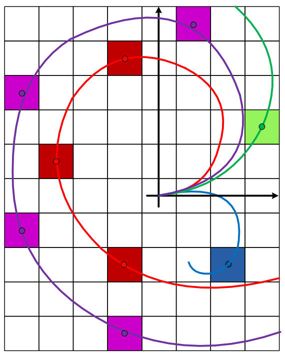


Fig. 10. Look-up array

For trajectory planning, the values of a from the interval [0.05; 5.0] are reasonable. This is because smaller values create very large and larger values

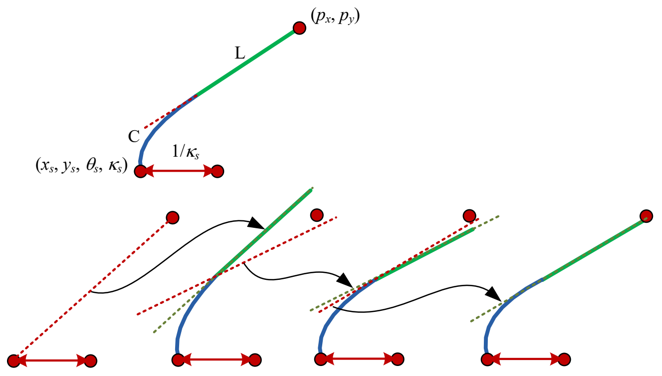


Fig. 11. Construction of C_0L maneuvers

very tight spirals. Both are not suitable in the context of trajectory planning. Values from s up to 25 are sufficient as larger values create more than one spiral 'round'. For our experiments we created a grid with cell size of 0.2 with 100×100 cells. We created approx. 1.7 millions sample clothoids to fill the grid.

For a position (p_x, p_y) we first look up clothoid parameters a', s' asking the cell of $(p_x \cdot \kappa_s, p_y \cdot \kappa_s)$. As this is only an approximation, we improve the precision with an optimization approach (e.g., downhill simplex). We finally get $a = a' \cdot \kappa_s$, $s = s' \cdot \kappa_s$.

4.7. Clothoid Extended by a Straight Line Through a Given Point

Also in continuous-curvature planning we have to create maneuvers that span large distances; usually we have to integrate an L trajectory for this. A maneuver with single L would require that the start orientation exactly points to the target position. In practice, this never occurs. The most simple continuous-curvature maneuver that uses an L trajectory and starts from an arbitrary pose is of type C_0L (Clotho-CL). Note that this pattern does not support to specify a target orientation.

The actual problem is to construct a clothoid that points to the target position at $\kappa = 0$ (Fig. 11). Our approach to construct such a clothoid is as follow: we first measure the angle of start point to target point. We then construct a clothoid with the required start curvature that has the respective orientation at $\kappa = 0$. This can easily be computed using the formulas for κ and φ (7).

The required clothoid now has the respective angle to create an L trajectory, but the position of $\kappa = 0$ is not the starting point, thus we miss the target. We use the point to again measure the angle to the target, which starts from a more suitable position. We again use the measured angle to create a more precise clothoid. The angle and thus the clothoid position at $\kappa = 0$ quickly converges, if the target is not too close the clothoid turning point. Thus, this approach is only suitable, if targets exceed a certain distance. This however is reasonable, as this type of maneuver explicitly should spawn a larger distance.

4.8. Clothoid Extended by a Straight Line With Given Distance

The next question: how to create a maneuver that starts from a non-zero curvature and ends with zero curvature, when the target orientation is given?
We assume the structure pattern AC_0L (maneuver Clotho-ACL or the reverse Clotho-LCA). Fig. 12 illustrates the problem.

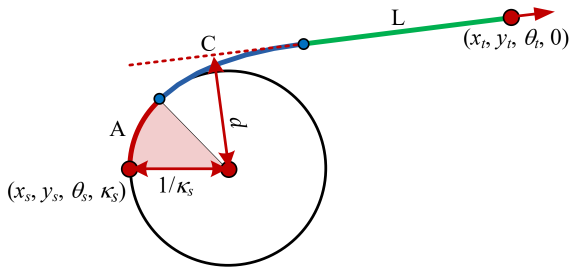


Fig. 12. Construction of AC_0L maneuvers

The start configuration defines a circle, on which the clothoid must start with curvature κ_s. The target configuration defines a line, on which the clothoid must terminate with curvature zero. The respective clothoid's start and target points are not given. However, we know the distance d that is the distance of leading arc centre to terminating line. Let

$$d = \delta_{\kappa s}(a) \tag{14}$$

the function that computes the respective distance for given start curvature and value a. We are able to compute δ using formulas (7), but these formulas cannot easily be inverted. What we actually need is

$$a = \delta_{\kappa s}^{-1}(d) \tag{15}$$

A first approach already applied in former sections: as all clothoids are similar apart from the scaling, we only invert δ for a single start curvature $\kappa_s = 1$ and rescale the results later. Moreover, we do not have to consider all d for $\kappa_s = 1$: it should not be too close to 1.0 as this would mean to drive a long distance until we change to L. It should also not be too large as this would require a quick change of curvature. For our experiments we select $d \in [1.1, 5.0]$, what results in $a \in [0.1254, 0.565]$. As we effectively are able to compute δ_1, we sample some hundreds values of δ_1 as pre-computed data points and interpolate to compute any value of δ_1^{-1}. We again improve the precision with an optimization approach (e.g., downhill simplex). We finally get an approximation for all start curvatures:

$$\delta_{\kappa s}^{-1}(d) \approx \kappa_s \delta_1^{-1}(d) \tag{16}$$

If the target angle θ_t is not given, we can construct the maneuver that drives to (x_t, y_t) with any orientation. This can be useful for inner route points, when we only want to visit the route point whereas the actual orientation is not of interest (Clotho-ACL-nodir). In this case, we do not have to construct a certain d, but are interested, how fast the clothoid transforms the start curvature to zero. This is expressed by the factor $d \cdot \kappa_s$. Table 3 shows some values.

Table 3. Value a for different distance factors

$d{\cdot}\kappa_s$	a/κ_s
1.5	0.3687568411482851
2.0	0.2983815391022415
2.5	0.2550963611370387
3.0	0.21431547506198895
4.0	0.1575822277624827
5.0	0.12540637523181616

Here $d{\cdot}\kappa_s$ expresses the speed of curvature change and can be adapted to the robot's properties.

4.9. Maneuvers Between Non-Zero Curvatures

The most complex continuous-curvature maneuver that is not a result of transforming or extending existing maneuvers has the structure pattern AC_0L^0CA. It connects two configurations with non-zero curvatures.

The construction is similar to the AC_0L maneuver of section 4.8. In contrast to AC_0L *with* given target orientation, the orientation of the L trajectory in the AC_0L^0CA is not given. We can consider the orientation as free parameter. However, it is more suitable to express the degree of freedom with the approach of fixed values of $d{\cdot}\kappa_s$ (Table 3).

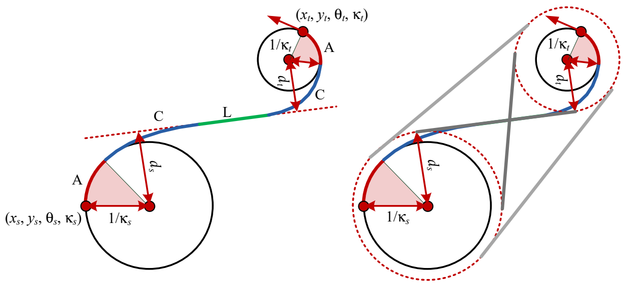


Fig. 13. Construction of AC_0L^0CA maneuvers

Fig. 13 shows the idea. We have two free parameters $d_s{\cdot}\kappa_s$ and $d_t{\cdot}\kappa_t$. For a certain selection we get two circles and four tangents (Fig. 13 right) that touch these circles. From the four tangents two would change to target orientation by 180° – the choice for the remaining tangents thus is an additional free (binary) parameter.

4.10. Summary of Approaches

In addition to the well-known clothoid computations in section 4.1, we introduced functions that are related to continuous-curvature trajectories.

Table 4 gives an overview of the last sections. Some of them reverse clothoid functions with the help of pre-computations and subsequent optimizations. The applied techniques cover a wide range of methods as there is no such best approach for all required clothoid functions.

Table 4. Clothoid functions

Function	Approach	Section	Example Maneuver
Cont-curve maneuver creation	replace A by $^0CC_{\text{-}0}$	4.2	J-Clothoid-Bow
Cont-curve maneuver creation	append trailing C_0	4.3	C-∫-Clothoid-Arcs
Nearest clothoid point	pre-computed grid, correction with arc approximation	4.4	all
^0C to given point	inversion of Q	4.5	Clotho-0C
C to given point	pre-computed grid, subsequent optimization	4.6	Clotho-C
C_0L to given point	converging L angle construction	4.7	Clotho-CL
AC_0L to given pose	closed formula for approximation of *a*, subsequent optimization	4.8	Clotho-ACL
AC_0L^0CA to given pose	tangents of two circles, *a* values from table	4.9	Clotho-ACLCA

5. Experiments

We implemented the approach on our Carbot robot (Fig. 14). It has a size of 35 cm x 40 cm x 27 cm and a weight of 4.9 kg. It is able to run with a speed of 31 cm/s. The wheel configuration allows to independently steer two wheels.

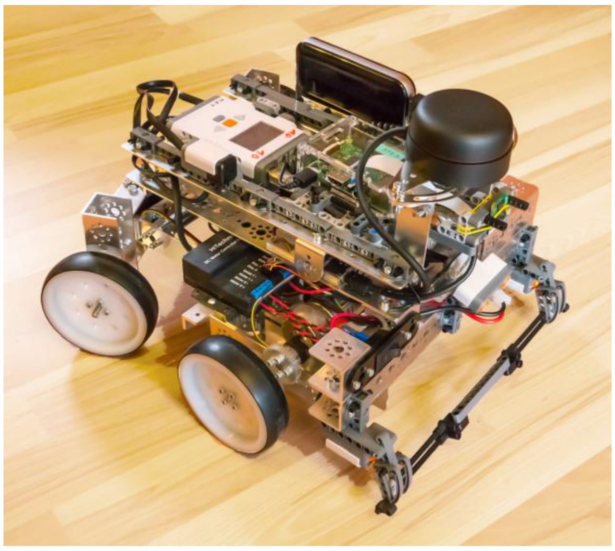


Fig. 14. The Carbot

For arcs, the different numbers of revolutions of the powered front wheels as well as the steering angles of the steered rear wheels are adapted to follow the respective curve geometry. The motion system is also able to drive clothoids with a single motion command – here the servos and revolutions are smoothly changed according to the clothoid's geometry.

We run the experiments in our simulation environment that simulates the robot on hardware-level [21]. It is very close to the real robot. E.g., the same binary code runs on the real robot and simulator. Motors and sensors are simulated very low-level. E.g., the simulated and real motors have the same I^2C command interface. Physical effects such as slippage and sensor errors are applied. The benefit of the simulation tool: we can easily construct very large environments. E.g., we constructed a virtual maze with size of 50 m × 50 m which the robot has to discover in order to find a way out.

In our experiments, we first want to know the performance of our approximation functions (Table 5). We measured the time on a i7-4790 CPU, 3.6 GHz. Most of the approximations are uncritical concerning execution time. The exception is the approach described in section 4.6. We investigate that nearly all of the time to compute the C trajectory is used by the downhill simplex approach to improve the precision of a. Fortunately, the approximation is only used to construct a single maneuver, the Clotho-C.

Tab. 5. Clothoid function computation times

Function	Section	Execution time
Nearest clothoid point	4.4	2.78 µs
${}^0C$ to given point	4.5	2.21 µs
C to given point	4.6	196.1 µs
C_0L to given point	4.7	11.5 µs
AC_0L to given pose	4.8	17.7 µs

We further tested the continuous-curvature planning in several scenarios. We constructed different environments. A maze generator is able to construct mazes with arbitrary size. We also created mixed environments with different shaped obstacles. In each environment the robot is able to perform two tasks:

- *Exploration and navigation*: Here, the environment is unknown at startup and the robot should drive to a certain position. For this, the robot plans movement on an incomplete map. During movement, the robot explores formerly unknown obstacles, thus has to reschedule the movement. Driving and rescheduling are alternated, until the target is reached.
- *Only Navigation*: here the environment is known, either from former exploration or because the ground map is provided by the caller. In this case, the robot only has to compute a single path and drive it.

Fig. 15. shows screenshots of two environments (top: a large maze, bottom: a mixed environment). For each, the left images show the single plan for a known map (*only navigation*). The right images show some steps when the robot tried to move to the target in unknown environments (*exploration and navigation*).

Paths for known environments show very fine-granular, smooth and rational trajectories. For the case of unknown maps, the robot may unwittingly drive to dead-ends. In this case, the robot either may drive a U-turn or a turning maneuver. We can see both cases. Note that planning turning maneuvers usually is not trivial, in particular if we want to have continuous curvatures. The robot has to take into account the costs for backdriving compared to the costs for a U-turn. As our approach incorporates arbitrary costs functions, we are able to penalize trajectories that go backwards, as in this case the robot has to drive slower. Note that the Viterbi approach is able to deal with such situations, even though only pairs of connected trajectories are locally optimized.

We finally evaluated the occurrence of our continuous-curvature maneuvers in paths. We measured:

- The percentage of a specific maneuver that is selected as best to connect two subsequent (intermediate) configurations (Fig. 16 left). This was indicated as 'best maneuver' for each step in Fig. 4.
- The percentage of a specific maneuver to appear in the finally planned trajectory sequence to a target (Fig. 16 right).

The histograms are not identical, as the creation of the final trajectory is strongly influenced by the selection of near-optimal intermediate configurations. In other words: a certain maneuver may have a large portion to connect arbitrary configurations (that may turned out not to be near-optimal), but a lower portion to connect configurations that appear more often in optimal sequences.

An observation: maneuvers that make use of the arc replacement (Section 4.2) and leading clothoids (Section 4.2) often appear in planned sequences. On the other hand, maneuvers that connect non-zero curvatures (Section 4.9) only rarely appear. This however is a result of the fact that many of our maneuvers terminate with L, C_0 or C_{-0}, i.e., zero curvature.

A further observation: of the computations in Table 5, the ' C to given point' had worse performance. But the respective maneuver Clotho-C only rarely appeared both as best maneuver in a optimization step and in the planned path. We thus could remove this maneuver from the list without serious consequences.

6. Conclusion

Trajectory planning for mobile robots still is a non-trivial task. If we additionally request continuous-curvature planning that considers obstacles and arbitrary cost functions, we have to face numerous challenges, starting from the actual planning approach down to approximations that compute the respective trajectory parameters.

There is an infinite number of trajectories that connect two configurations and optimal continuous-curvature paths may have an infinite number of primitive trajectories. We thus need a simplification of the overall problem. We suggest modelling the final path by maneuvers, for which we know formulas to define the respective geometric parameters. We introduced a set of about twenty continuous-curvature maneuvers and provide basic approximations, when-

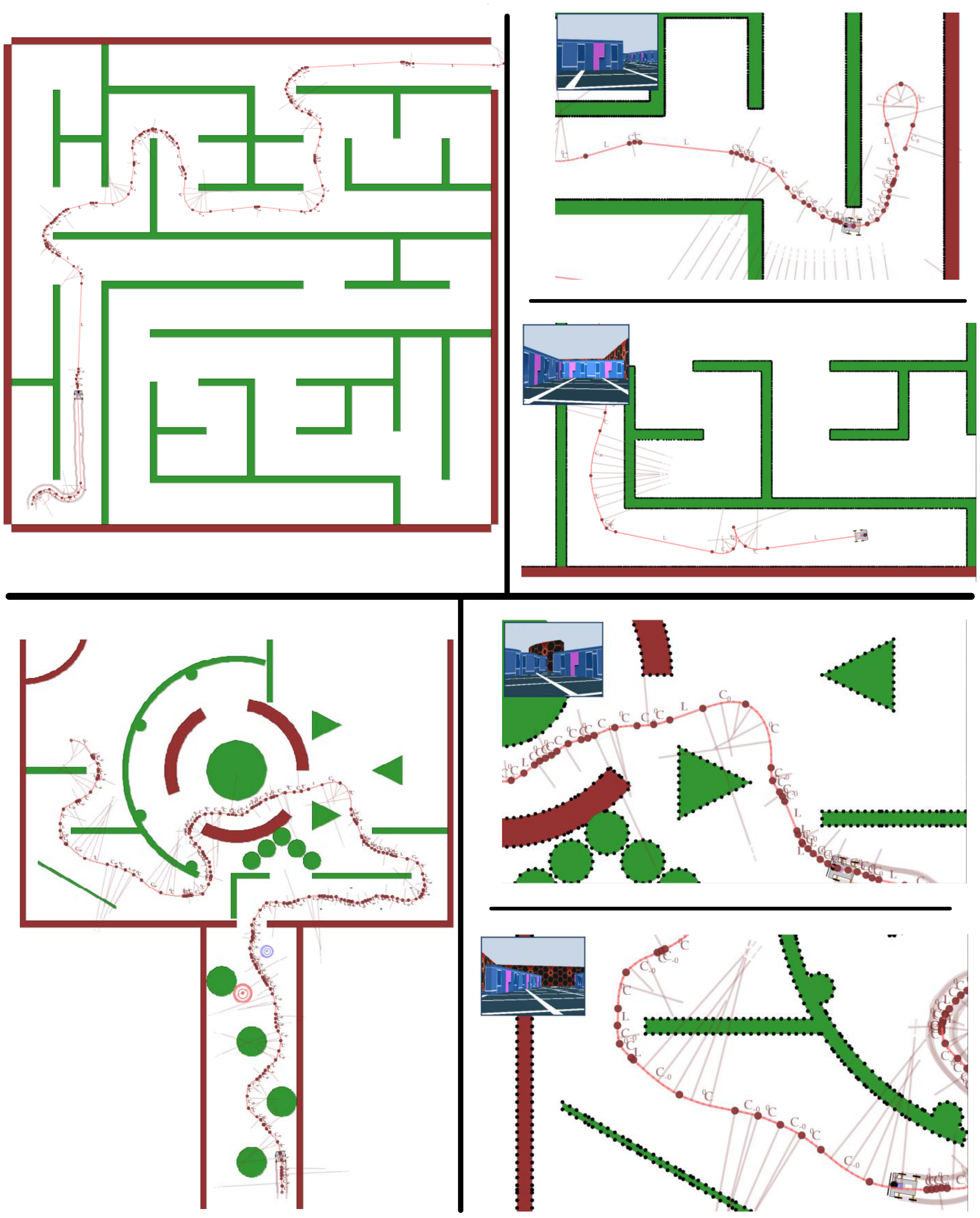


Fig. 15. Planned paths in different environments

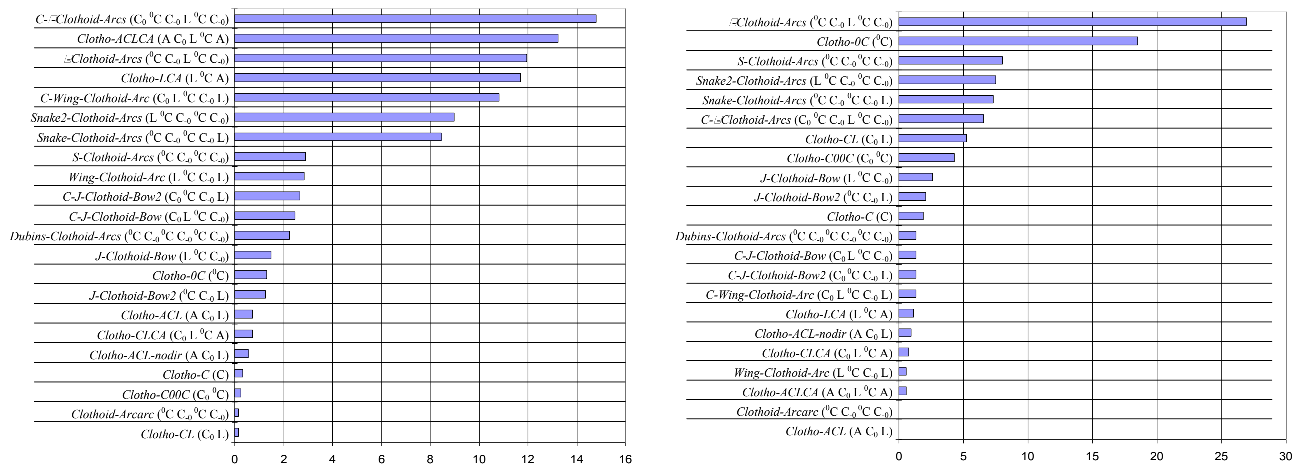


Fig. 16. Percentage of a maneuver in a Viterbi step (left), percentage of a maneuver in the final path (right)

ever the underlying clothoid function cannot easily be reversed. As an observation: the respective questions to find parameters are different between the maneuvers and lead to different approximation – we finally discovered five methods.

We successfully implemented the approach on our Carbot platform. Carbot is able to create continuous-curvature paths to, e.g., get through a large maze.

AUTHOR

Jörg Roth – Faculty of Computer Science, Nuremberg Institute of Technology, Nuremberg, Germany, e-mail: joerg.roth@th-nuernberg.de.

REFERENCES

[1] 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, 1992, 224–241, 10.1109/21.148426.

[2] J.-D. Boissonnat, A. Cerezo and J. Leblond, "A note on shortest paths in the plane subject to a constraint on the derivative of the curvature", *Research Report,* Institut National de Recherche en Informatique et an Automatique, 1994.

[3] X.-N. Bui, J.-D. Boissonnat, P. Soueres and J.-P. Laumond, "Shortest path synthesis for Dubins non-holonomic robot". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, 2–7, 10.1109/ROBOT.1994.351019.

[4] H. Delingette, M. Hebert and K. Ikeuchi, "Trajectory generation with curvature constraint based on energy minimization". In: *Proceedings IROS '91: IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, 1991, 206–211, 10.1109/IROS.1991.174451.

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

[6] 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, 1995, 441–448, 10.1109/70.388788.

[7] T. Gu and J. M. Dolan, "On-Road Motion Planning for Autonomous Vehicles". In: C.-Y. Su, S. Rakheja and H. Liu (eds.), *Intelligent Robotics and Applications*, 2012, 588–597, 10.1007/978-3-642-33503-7_57.

[8] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli and S. Teller, "Anytime Motion Planning using the RRT*". In: *2011 IEEE International Conference on Robotics and Automation*, 2011, 1478–1483, 10.1109/ICRA.2011.5980479.

[9] J.-P. Laumond, P. E. Jacobs, M. Taïx and R. M. Murray, "A motion planner for nonholonomic mobile robots", *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, 1994, 577–593, 10.1109/70.326564.

[10] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning" , *Research Report,* Computer Science Department, Iowa State University, 1998.

[11] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning", *The International Journal of Robotics Research*, vol. 20, no. 5, 2001, 378–400, 10.1177/02783640122067453.

[12] T.-C. Liang, J.-S. Liu, G.-T. Hung and Y.-Z. Chang, "Practical and flexible path planning for car-like mobile robot using maximal-curvature cubic spiral", *Robotics and Autonomous Systems*,

vol. 52, no. 4, 2005, 312–335, 10.1016/j.robot.2005.05.001.

[13] D. S. Meek and D. J. Walton, "A note on finding clothoids", *Journal of Computational and Applied Mathematics*, vol. 170, no. 2, 2004, 433–453, 10.1016/j.cam.2003.12.047.

[14] D. S. Meek and D. J. Walton, "An arc spline approximation to a clothoid", *Journal of Computational and Applied Mathematics*, vol. 170, no. 1, 2004, 59–77, 10.1016/j.cam.2003.12.038.

[15] M. Mukadam, X. Yan and B. Boots, "Gaussian Process Motion planning". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, 9–15, 10.1109/ICRA.2016.7487091.

[16] V. Muñoz and A. Ollero, "Smooth Trajectory Planning Method for Mobile Robots". In: *Proc. of the Congress on Comp. Engineering in System Applications*, Lille, France, 1995, 700–705.

[17] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices". In: *Proceedings of 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS '05)*, 2005.

[18] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards.", *Pacific Journal of Mathematics*, vol. 145, no. 2, 1990, 367–393.

[19] J. Roth, "A Viterbi-like Approach for Trajectory Planning with Different Maneuvers". In: M. Strand, R. Dillmann, E. Menegatti and S. Ghidoni (eds.), *Intelligent Autonomous Systems 15*, 2019, 3–14, 10.1007/978-3-030-01370-7_1.

[20] J. Roth, "Trajectory Regulation for Walking Multipod Robots", *International Journal on Advances in Systems and Measurements*, vol. 12, no. 3 & 4, 2019, 265–278.

[21] J. Roth, "Robots in the Classroom: Mobile Robot Projects in Academic Teaching". In: K.-H. Lüke, G. Eichler, C. Erfurth and G. Fahrnberger (eds.), *Innovations for Community Services*, 2019, 39–53, 10.1007/978-3-030-22482-0_4.

[22] A. Scheuer and T. Fraichard, "Continuous-curvature path planning for car-like vehicles". In: *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS '97)*, vol. 2, 1997, 997–1003, 10.1109/IROS.1997.655130.

[23] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Transactions on Information Theory*, vol. 13, no. 2, 1967, 260–269, 10.1109/TIT.1967.1054010.

[24] D. K. Wilde, "Computing clothoid segments for trajectory generation". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, 2440–2445, 10.1109/IROS.2009.5354700.

[25] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian Optimization for Motion Planning", *The International Journal of Robotics Research*, vol. 32, no. 9-10, 2013, 1164–1193, 10.1177/0278364913488805.