IEEE Int. Conf. on Robotics and Automation
May 21-26, 2001. Seoul (KR)

Smooth Path Planning for Cars

Th. Fraichard and J.-M. Ahuactzin
Inria[a] Rhône-Alpes & Gravir[b]
655 av. de l'Europe, Montbonnot, 38334 St Ismier Cedex, France
http://www.inrialpes.fr/sharp

thierry.fraichard@inria.fr,
juan-manuel.ahuactzin@inria.fr

February 20, 2001

Abstract — *This paper addresses the problem of continuous-curvature path planning for car-like vehicles. Previously, the authors had presented a steering method,* i.e. *an algorithm computing paths in the absence of obstacles, derived from Reeds and Shepp's work on shortest paths for car-like vehicles. This paper presents two improvements of the initial steering method. To begin with, the improved steering method computes shorter paths. The second improvement concerns the completeness issue: the improved steering method verifies a given topological property ensuring that when it is used inside a global planning scheme (so as to take obstacles into account), it yields a complete path planner. Finally, the paper details the embedding of the steering method inside such a global path planning scheme, namely the Ariadne's Clew Algorithm, and presents experimental results.*

Keywords — vehicle, nonholonomic system, path planning, Ariadne's Clew algorithm.

Acknowledgements — this work was partially supported by the French programme "La Route Automatisée": http://www.lara.prd.fr.

[a]Institut National de Recherche en Informatique et en Automatique.
[b]Lab. Graphisme, Vision et Robotique.

Smooth Path Planning for Cars

Th. Fraichard and J.-M. Ahuactzin
Inria Rhône-Alpes & Gravir
http://www.inrialpes.fr/sharp

Abstract — *This paper addresses the problem of continuous-curvature path planning for car-like vehicles. Previously, the authors had presented a steering method,* i.e. *an algorithm computing paths in the absence of obstacles, derived from Reeds and Shepp's work on shortest paths for car-like vehicles. This paper presents two improvements of the initial steering method. To begin with, the improved steering method computes shorter paths. The second improvement concerns the completeness issue: the improved steering method verifies a given topological property ensuring that when it is used inside a global planning scheme (so as to take obstacles into account), it yields a complete path planner. Finally, the paper details the embedding of the steering method inside such a global path planning scheme, namely the Ariadne's Clew Algorithm, and presents experimental results.*

1 Introduction

As part of its effort to design an autonomous car-like vehicle, the SHARP research group at INRIA Rhône-Alpes has developed new motion planning techniques that take into account the various constraints that restricts the motion capabilities of such a vehicle (kinematics, dynamics, uncertainty). From a kinematic standpoint, cars are subject to nonholonomic kinematic constraints that restrict their admissible directions of motion (they can only move forward or backward in a direction perpendicular to the orientation of their rear wheels axle; besides their turning radius is lower bounded because of the mechanical limits on the steering angle). Nonholonomy makes motion planning more difficult since the motions planned must take into account both the constraints imposed by the obstacles and the nonholonomic constraints.

Nonholonomy appeared in path planning in the mid-eighties and a lot of results have been obtained since (the reader is referred to [12] for a recent review on this topic). For cars, most of the existing works compute paths made up of line segments connected with tangential circular arcs of minimum radius (one reason for this choice is that the shortest path between two configurations for a car is such a path. This result was established first by Dubins [6] for the car moving forward only and later by Reeds and Shepp [16] for the car moving both forward and backward). However the curvature of this type of path is discontinuous. Since curvature is directly related to the orientation of the front wheels, if a car were to track precisely such a type of path, it would have to stop at each curvature discontinuity so as to reorient its front wheels. Curvature continuity is therefore a desirable property[1] and, in the past few years, SHARP has focused on the problem of planning continuous-curvature paths for car-like vehicles.

SHARP's approach to the problem is to start from the classical 'line segments-circular arcs' paths and to remove the curvature discontinuities by introducing clothoid transitions[2] between the lines and the arcs.

To begin with, we considered the case of the car moving forward only and presented the first continuous-curvature path planner for such a vehicle [18]. It introduced a steering method[3] (called *CC-Steer*$_{\text{smooth}}$) able to compute a continuous-curvature path between any two configurations. This steering method was embedded into a global planning scheme in order to take into account the obstacles of the environment. Then, we moved to the case where the car can move both forward and backward and proposed a steering method (called *CC-Steer*) for this new system [7].

Both steering methods computes paths made up of line segments, circular arcs and clothoid arcs. They select the shortest path from families of paths that are the continuous-curvature counterparts of the paths used in Dubins and Reeds and Shepp's works. The purpose of this paper is threefold:

1. First, we give a result establishing that the families of paths used by Dubins and Reeds and Shepp do not suffice in the continuous-curvature case. Using this result, we show how both *CC-Steer* and *CC-Steer*$_{\text{smooth}}$ can be improved in order to compute shorter paths.
2. Then, we show how to extend *CC-Steer* so that it satisfies a given topological property ensuring that, when it is used inside a global planning scheme such as the Probabilistic Path Planner [21], the Ariadne's Clew Algorithm [14] or the Holonomic Path Approximation Algorithm [12], it yields a complete path planner.
3. Finally, in order to solve the full path planning

[1] As a matter of fact, it is emphasized in [3] that feedback controllers for car-like vehicles require this property in order to guarantee the exact reproducibility of a path.

[2] A clothoid is a curve whose curvature varies linearly with its arc length.

[3] 'Steering method' is a term borrowed from [12], it is an algorithm that computes a path without taking into account the obstacles of the environment.

problem, we embed *CC-Steer* into a global planning scheme, the Ariadne's Clew Algorithm, and present the experimental results obtained.

The paper is organized as follows: §2 describes the model of the car-like vehicle considered and its main properties (controllability, optimal paths, etc.). §3 briefly recalls the principle of *CC-Steer*. The improvements concerning both *CC-Steer* and *CC-Steer*$_{\text{smooth}}$ are presented in §4 while the extension of *CC-Steer* with respect to the completeness issue is presented in §5. §6 details the embedding of *CC-Steer* inside the Ariadne's Clew Algorithm and presents experimental results. Finally, a review of the works related to continuous-curvature path planning is carried out in §7 along with a discussion about the features of *CC-Steer*.

2 Continuous-Curvature Car

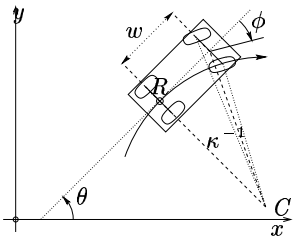

Fig. 1: model of the car-like vehicle.

Let $\mathcal{A}$ denote the system represented in Fig. 1. A configuration of $\mathcal{A}$ is defined as $q = (x, y, \theta, \kappa)$ where (x, y) are the coordinates of the rear axle midpoint R and θ the orientation of $\mathcal{A}$. κ is the curvature of the xy-curve traced by R, it is used to characterize the orientation of the front wheels of $\mathcal{A}$: $\kappa = w^{-1} \tan\phi$, where w is the wheelbase of $\mathcal{A}$ and ϕ its steering angle[4]. The steering angle is mechanically limited, $|\phi| \leq \phi_{\max}$, and a configuration is admissible if $|\kappa| \leq \kappa_{\max} = w^{-1} \tan\phi_{\max}$. The configuration space $\mathcal{C} \equiv \mathbb{R}^2 \times [0, 2\pi[\times [-\kappa_{\max}, \kappa_{\max}]$ of $\mathcal{A}$ is 4-dimensional.

The two controls of $\mathcal{A}$ are: (1) v, the driving velocity of the rear wheels, and (2) σ, the derivative of κ, which is directly related to the steering velocity of $\mathcal{A}$: $\sigma = \dot{\phi} / \cos^2\phi$. The model of $\mathcal{A}$ is described by the following differential system:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\kappa} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \kappa \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \sigma \qquad (1)$$

with $|v| \leq v_{\max}$ and $|\sigma| \leq \sigma_{\max}$. *CC car* (for Continuous-Curvature car) is used to denote a vehicle whose model is (1).

The CC car is a small-time controllable system[5] [19, Theorem 1]. In the absence of obstacles, if a path exists between two configurations then an optimal path exists [19, Theorem 2]. The nature of the optimal paths is more difficult to establish. However [19] demonstrates that, for the CC car moving forward only, the optimal paths are made up of: (a) line segments, (b) circular arcs of radius $\kappa_{\max}^{-1}$, and (c) clothoid arcs of sharpness $\pm\sigma_{\max}$. Unfortunately, whenever the shortest path includes a line segment, it is irregular and contains an infinite number of clothoid arcs that accumulate towards each endpoint of the segment [2]. Furthermore, when the distance between two configurations is large enough, the shortest path contains a line segment (hence an infinite number of clothoid arcs) [4]. In summary, although the exact nature of the optimal paths for the CC car has not been established yet, it seems reasonable to conjecture that they will (at least) be made up of line segments, circular arcs and clothoid arcs, and that they will be irregular in most cases.

[4] ϕ is the average orientation of the two front wheels of $\mathcal{A}$.

[5] The set of configurations reachable from q before a time t contains a neighbourhood of q for any t.

3 *CC-Steer*

This section recalls the principle of *CC-Steer*, a steering method for the CC car, that is otherwise presented in full details in [7].

The conjectured irregularity of the shortest paths for the CC car prevents their use (*cf.* §2). Instead, *CC-Steer* computes paths (denoted CC paths) derived from the Reeds and Shepp's paths (denoted RS paths) [16]. CC paths are similar to RS paths but, in order to ensure curvature continuity, the circular arcs are replaced by transitions called CC turns whose curvature varies continuously from 0 up and then down back to 0, and that are made up of circular arcs of radius $\kappa_{\max}^{-1}$ and clothoid arcs.

3.1 CC Turns

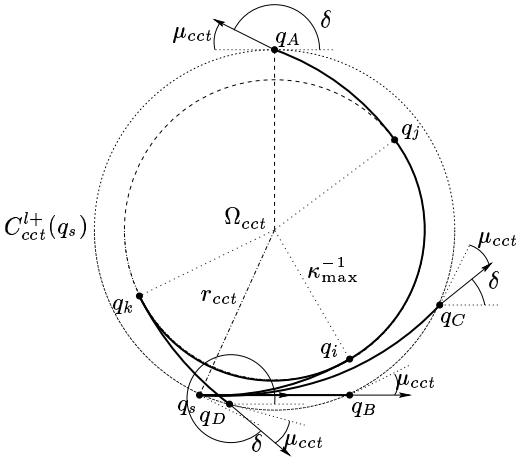

Fig. 2: a CC turn.

The definition of the CC turns stems from the following observation: a natural way to change the orientation of $\mathcal{A}$ is to: (a) move forward while turning the front wheels as fast as possible. (b) follow a circular arc once the maximum steering angle is reached, and (c) move forward while turning the front wheels back to a null steering angle.

What happens then is illustrated in Fig. 2. Let q_s be the start configuration. $\mathcal{A}$ can either move forward or backward and turn either left or right. Let us assume that $\mathcal{A}$ moves forward and turns left. First, it follows a clothoid arc of sharpness $\sigma_{\max}$ until it reaches q_i. Then it follows a circular arc of radius $\kappa_{\max}^{-1}$ until it reaches q_j. Finally it follows a clothoid arc of sharpness $-\sigma_{\max}$ until it reaches a goal configuration q_A. The path $q_s q_i q_j q_A$ constitutes a CC turn. It is characterized by its deflection, *i.e.* the change of orientation δ between q_s and q_A. When δ varies from 0 to 2π, the main result established in [18] is that the locus of the configurations q_A is a circle $C_{cct}^{l+}(q_s)$ ($l+$ stands for forward-left) whose centre Ω_{cct} and radius r_{cct} can be computed easily. In addition, the angle μ_{cct} between the orientation of q_A and the tangent to $C_{cct}^{l+}(q_s)$ at q_A is constant.

This general definition of the CC turns has further been refined in order to generate shorter paths (*cf.* [7]): when $\delta = 0$, the CC turn reduces a line segment (case q_B in Fig. 2). When $0 < \delta < \delta_{\min}$ where $\delta_{\min}$ is the deflection of the CC turn whose circular arc has zero length, the CC turn reduces to a pair of symmetric clothoid arcs of sharpness $\sigma \leq \sigma_{\max}$ (case q_C). When $\delta_{\min} \leq \delta < \delta_{\min} + \pi$, the circular arc from q_i to q_k is traveled backward (case q_D).

The above analysis can be repeated for the cases where $\mathcal{A}$ moves backward or turns right. It yields three similar circles $C_{cct}^{r+}(q_s)$, $C_{cct}^{l-}(q_s)$ and $C_{cct}^{r-}(q_s)$.

3.2 CC Paths

Like RS paths, CC paths are determined through the analysis of the tangency relationships that exists between the circles C_{cct}^{l+}, C_{cct}^{r+}, C_{cct}^{l-} and C_{cct}^{r-} defined earlier and attached to the start and goal configurations. *CC-Steer* computes the shortest among a set of paths that belong to one of the nine following families [20]:

$$\begin{array}{ll} (i)(ii)(iii) & T|T|T \text{ or } T|TT \text{ or } TT|T \\ (iv) & TT|TT \\ (v) & T|TT|T \\ (vi) & T|TST|T \\ (vii)(viii) & T|TST \text{ or } TST|T \\ (ix) & TST \end{array} \quad (2)$$

where T (resp. S) denotes a CC turn (resp. line segment). $|$ denotes a change of direction of motion. T may be replaced by $l+$, $l-$, $r+$ or $r-$ to specify a left/right turn and a forward/backward motion. The conditions of existence of the paths of the different families guarantee that at least one such path exists. Accordingly *CC-Steer* is complete, *i.e.* it always returns a path.

4 Improving *CC-Steer*$_{\text{smooth}}$/*CC-Steer*

In their initial design, both *CC-Steer*$_{\text{smooth}}$ and *CC-Steer* selected their shortest path from families of paths strictly corresponding to the families defined by Dubins[6] and Reeds and Shepp [16]. The motivation behind this choice was that it has been proved that the shortest Dubins and Reeds and Shepp's paths could not belong to another family. However, it turns out that this property is no longer satisfied for the type of paths searched by *CC-Steer*$_{\text{smooth}}$ and *CC-Steer*.

To establish this, we produce a counter-example showing that a result established by Dubins on the shape of the shortest paths [6, Sublemma, page 513] does not hold anymore when transferred to the continuous-curvature case. Based on this result, we propose an improvement of *CC-Steer*$_{\text{smooth}}$ and *CC-Steer* so that they compute shorter paths.

4.1 Counter-Example

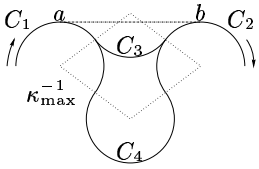

Fig. 3: Dubins' sublemma [6, page 513] : $C_1C_3C_2$ is never a shortest path.

Dubins' sublemma [6, page 513] is illustrated in Fig. 3. It states that a path made up of three circular arcs where the middle arc is of length less than $\pi\kappa_{\max}^{-1}$ (such as $C_1C_3C_2$) cannot be a shortest path (on the other hand, the path $C_1C_4C_2$ can be a shortest path). The intuition behind the proof is that a path of the type $C_1\overline{ab}C_2$ would be shorter.

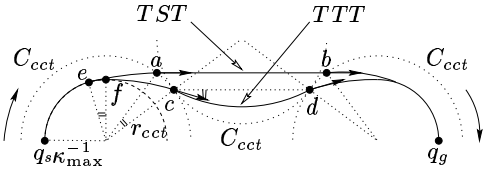

Fig. 4: the situation of Dubins' sublemma for CC paths.

Fig. 4 depicts the same situation for CC paths. Let us consider two CC paths from q_s to q_g (both with $\kappa_{\max}$ curvature), the first one of type TST and the second one of type TTT. In Dubins' case, TTT would always be longer than TST. It is not the case for CC paths.

Proof. Let $l(.,.)$ denote the length of the CC path connecting two configurations. The length of TST is $l(q_s, a) + l(a, b) + l(b, q_g)$ whereas the length of TTT is $l(q_s, c) + l(c, d) + l(d, q_g)$.

Let us focus on the first CC turn of each path, *i.e.* the CC turns from q_s to a and c respectively. The difference between them is that, in TST case, $\mathcal{A}$ starts to reorient its wheels when it reaches e whereas, in TTT case, it starts later when it reaches f. Accordingly, the CC turn from q_s to c is longer by $\delta\kappa_{\max}^{-1}$ where δ is the angular displacement from e to f. In other words, $l(q_s, c) = l(q_s, a) + \delta\kappa_{\max}^{-1}$. Symmetrically, it can be shown that $l(d, q_g) = l(b, q_g) + \delta\kappa_{\max}^{-1}$. Straightforwardly, TTT is shorter than TST if $l(a, b) > l(c, d) + 2\delta\kappa_{\max}^{-1}$.

Geometric analysis permits to express $l(a, b)$ as a function of δ, r_{cct} and μ_{cct}:

$$l(a,b) \quad = \quad 4r_{cct}sin(\delta/2)cos(\mu_{cct} + \delta/2) \qquad (3)$$

$$+2r_{cct}sin(\mu_{cct}+\delta)$$

The CC path from c to d is a CC turn of deflection 2δ. For small values of δ, it will be made up of two symmetric clothoid arcs whose sharpness $\sigma_{2\delta}$ can be computed uniquely [18]. It has also been shown in [18] that:

$$l(c,d) = 2\sqrt{2\delta/\sigma_{2\delta}} \tag{4}$$

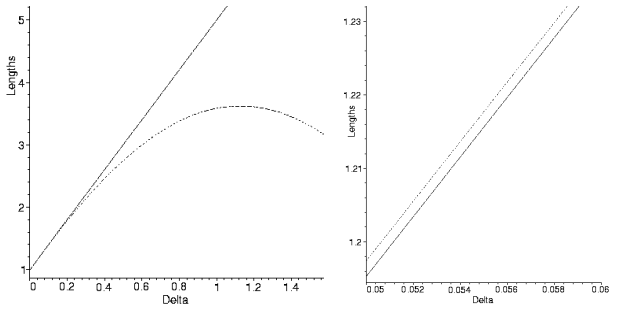

Fig. 5: $l(a,b)$ (solid line) and $l(c,d)+2\delta\kappa_{\max}^{-1}$ (dashed line) as functions of $\delta \in [0,\pi/2]$ (left), and a close-up for values of δ between 0.05 and 0.06 (right).

Setting the values of $\kappa_{\max}$ and $\sigma_{\max}$ determines the values of r_{cct} and μ_{cct}. Then, using (3) and (4), it is possible to plot both $l(a,b)$ and $l(c,d)+2\delta\kappa_{\max}^{-1}$ as functions of δ in order to compare them.

When $\kappa_{\max}$ and $\sigma_{\max}$ are set to 1, we obtain the plot depicted in Fig. 5-left. $l(a,b)$ (the solid line) appears to always be greater than $l(c,d)+2\delta\kappa_{\max}^{-1}$ (the dashed line). However, upon closer inspection, it appears that it is not the case for small values of δ (Fig. 5-right). In other words, there are situations where TTT can be shorter than TST.

4.2 Improvement

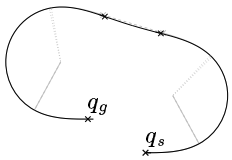

Fig. 6: the CC paths computed by *CC-Steer*$_{\text{smooth}}$ (dashed line) and the improved *CC-Steer*$_{\text{smooth}}$ (solid line): $\kappa_{\max} = \sigma_{\max} = 1$, length difference = 0.00221.

The counter-example above shows that, when *CC-Steer*$_{\text{smooth}}$ (resp. *CC-Steer*) restrict its search for a CC path to the families of paths defined by Dubins (resp. Reeds and Shepp), it may miss a CC path that would be shorter. A straightforward way to improve both *CC-Steer*$_{\text{smooth}}$ and *CC-Steer* is to extend the set of families to search.

Improving *CC-Steer*$_{\text{smooth}}$. The initial set of families included TST and TTT paths with the restriction on TTT paths imposed by Dubins' sublemma. Lifting this restriction leads *CC-Steer*$_{\text{smooth}}$ to consider the TTT paths for which the arc length of the middle arc is less than $\pi\kappa_{\max}^{-1}$. Fig. 6 depicts a situation where the improved *CC-Steer*$_{\text{smooth}}$ finds a shorter path.

Improving *CC-Steer*. The improved *CC-Steer* also considers an extension of the set of families defined in (2). The nature of the CC turns makes the characterization of the minimum set of families that would include the shortest CC path an intricate task. We therefore decided, arbitrarily, to consider the following 40 families that include the initial 9 families plus families of paths made up of at most 5 parts:

$$\begin{array}{ll} (i)-(iv) & T[|]S[|]T \\ (v)-(viii) & T[|]T[|]T \\ (ix)-(xvi) & T[|]T[|]S[|]T \\ (xvii)-(xxiv) & T[|]S[|]T[|]T \\ (xxv)-(xl) & T[|]T[|]S[|]T[|]T \end{array} \tag{5}$$

where [|] denotes an optional change of direction.

Note. In *CC-Steer*$_{\text{smooth}}$'s case, the improvement is marginal since the gain in length is small (of the order of δ) and seldom happens. In *CC-Steer*'s case on the other hand, the gain can be quite important. Simply by considering TTT paths, we have encountered situations where *CC-Steer* would find CC paths 20% shorter.

5 Completing *CC-Steer*

A steering method such as *CC-Steer* computes path in the absence of obstacles. To take them into account so as to compute collision-free paths, steering methods can be used inside global path planning schemes such as the Probabilistic Path Planner [21], the Ariadne's Clew Algorithm [14] or the Holonomic Path Approximation Algorithm [12]. These schemes use the steering method to connect selected configurations. In order to ensure that the coupling between the planning scheme and the steering method yield a complete planner, the steering method is required to verify the following topological property:

$$\begin{array}{l} \forall\varepsilon>0, \exists\eta>0, \forall(q_1,q_2)\in\mathcal{C}^2, \\ q_2\in\mathcal{B}(q_1,\eta) \Longrightarrow \mathit{Steer}\ (q_1,q_2)\subset\mathcal{B}(q_1,\varepsilon) \end{array} \tag{6}$$

Where $\mathcal{B}(q,\varepsilon)$ denotes the configuration space ball of size ε centered around q, and *Steer* (q_1,q_2) denotes the path from q_1 to q_2 computed by the steering method. In other words, the steering method must be able to connect two η-neighbour configurations by a path that remains in an ε-neighbourhood.

Unfortunately, because of the nature of the CC paths that it computes, *CC-Steer* does not verify property (6). It can be seen in Fig. 7: the different CC turns composing the CC paths are supported by the circles C_{cct} whose radius r_{cct} is a function of $\kappa_{\max}$ and $\sigma_{\max}$. No matter how close the start and goal configurations are (except if they are aligned), the CC path between them includes at least one CC turn and since the minimum length of a CC turn is $2r_{cct}\mu_{cct}$ [7], (6) will be violated.

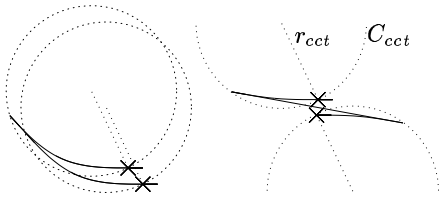

Fig. 7: two examples of CC paths computed by *CC-Steer* for neighbour configurations.

The approach we have chosen to make *CC-Steer* verify (6) is to further extend the set of families defined in (5). A new family of CC path is introduced, it will be considered when the start and goal configurations are close (for instance when their distance is less than $2r_{cct}\mu_{cct}$), and it will ensure that *CC-Steer* verify (6).

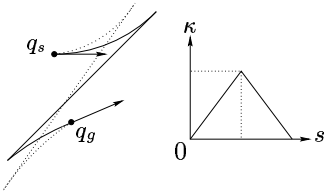

Fig. 8: the new family of CC paths for close configurations (left), and the curvature profile of an elementary path with respect to its arc length s (right).

The new CC paths are called εCC paths, they are made up of a pair of symmetric clothoid arcs, a line segment and a final pair of symmetric clothoid arcs (Fig. 8-left). This type of path can connect any two configurations. The two pairs of clothoid arcs are elementary paths as defined in [17] (Fig. 8-right).

The rest of the section outlines why a steering method that computes εCC paths verifies (6). Depending on the respective sharpness of their elementary paths, there is an infinity of εCC paths between two given configurations (Fig. 8-left). However, it will be seen how to select such a path. Due to lack of space, some of the details are omitted. The reader is referred to the forthcoming research report for more details.

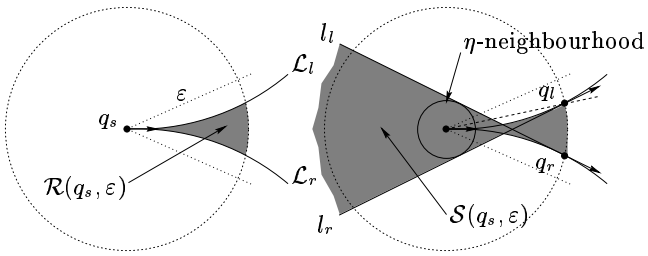

Fig. 9: the set of symmetric configurations reachable by an elementary path (left), and the region through which the line segment of the εCC path must pass (right).

Let us consider an εCC path from q_s to q_g. q_i denote the configuration reached at the end of the first elementary path. q_s and q_i are symmetric, *i.e.* their orientations are symmetric with respect to the line connecting their respective positions [17].

An elementary path reaches its maximum curvature κ at its midpoint. The set of symmetric configurations that can be reached by an elementary path without violating the maximum curvature constraint $\kappa \leq \kappa_{\max}$ has been formally characterized in [17]. Within an ε-neighbourhood of q_s, it is bounded by two semi-circular curve $\mathcal{L}_l$ and $\mathcal{L}_r$ (*cf.* [17] for more details). The grey area denoted $\mathcal{R}(q_s,\varepsilon)$ of Fig. 9-left depicts this region.

Given a point (x_i, y_i) of $\mathcal{R}(q_s,\varepsilon)$, there is a unique orientation θ_i such that $q_i = (x_i, y_i, \theta_i)$ is symmetric with q_s. θ_i is also the orientation of the line segment of the εCC path. Let us study the values that θ_i can take. Let q_l (resp. q_r) denote the configuration symmetric to q_s whose position is the intersection point between $\mathcal{L}_l$ (resp. $\mathcal{L}_r$) and the boundary of the ε-neighbourhood of q_s. By construction, q_l and q_r are the configurations (symmetric with q_s) for which the change of orientation is maximum. Accordingly the maximum range of orientation of the line segment of the εCC path is $]\theta_r, \theta_l[$ (θ_l and θ_r respectively denote the orientation of q_l and q_r). Note that, since $\mathcal{L}_l$ and $\mathcal{L}_r$ are symmetric with respect to the x axis, $\theta_l = -\theta_r$. It can also be shown that $\theta_l < \varepsilon$.

Now, the set of line segments determined by every possible symmetric configurations located in $\mathcal{R}(q_s,\varepsilon)$ defines an open cone-like region $\mathcal{S}(q_s,\varepsilon)$ such as the one depicted in Fig. 9-right. $\mathcal{S}(q_s,\varepsilon)$ is 'bounded' by the two line segments l_l and l_r of respective orientation θ_l and θ_r. $\mathcal{S}(q_s,\varepsilon)$ is important since it defines the region within which q_j, the end point of the line segment of the εCC path must be located (with respect to q_s). q_j is also the starting point of the second elementary path of the εCC path, *i.e.* the elementary path that connects q_j to q_g.

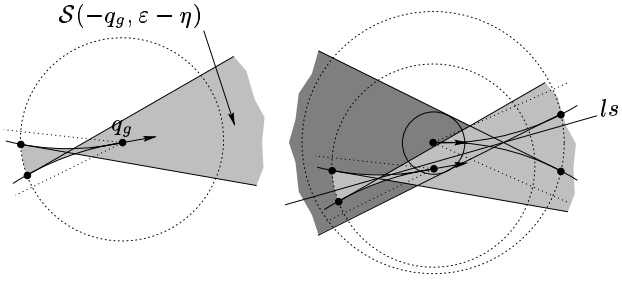

Fig. 10: the region $\mathcal{S}(-q_g, \varepsilon-\eta)$ (left), and an example of a line segment ls (right).

Let us assume that q_g is located within an η-neighbourhood of q_s, $\eta < \varepsilon$. It is possible, through a similar reasoning, to build first $\mathcal{R}(-q_g, \varepsilon-\eta)$ and then $\mathcal{S}(-q_g, \varepsilon-\eta)$ where $-q_g$ is the configuration whose orientation is $-\theta_g$. In this case, $\mathcal{S}(-q_g, \varepsilon-\eta)$ defines the region within which q_i must be located (with respect to q_g) (Fig. 10-left).

Now to build the εCC path from q_s to q_g, it suffices to find a line segment ls passing through both $\mathcal{S}(q_s,\varepsilon)$

and $\mathcal{S}(-q_g, \varepsilon - \eta)$ (Fig. 10-right). Since ls passes through $\mathcal{R}(q_s, \varepsilon)$ (resp. $\mathcal{R}(-q_g, \varepsilon - \eta)$), it defines a unique q_i (resp. q_j) symmetric with q_s (resp. q_g).

As shown earlier, The elementary path from q_s to q_i is included inside the η-neighbourhood of q_s. As for the elementary path from q_j to q_g, the fact that the $(\varepsilon - \eta)$-neighbourhood of q_g was used, it is guaranteed that $\mathcal{R}(-q_g, \varepsilon - \eta)$ is also entirely included inside the ε-neighbourhood of q_s. The resulting εCC path is therefore entirely included inside the η-neighbourhood of q_s.

Finally, the last problem to solve is the problem of the existence of the line segment ls. η must be selected so as to ensure that at least one such line segment will exist. It can be shown that such a line segment always exists if η is equal to the size of the neighbourhood centered at q_s and entirely included in $\mathcal{S}(q_s, \varepsilon)$ (Fig. 9-right). A good candidate for ls is the line segment that passes through the two intersections of l_l and l_r for $\mathcal{S}(q_s, \varepsilon)$ and $\mathcal{S}(-q_g, \varepsilon - \eta)$.

In summary, for a given ε, it is possible to find η such that, if q_g is located inside the η-neighbourhood of q_s, the εCC path from q_s to q_g is entirely included inside the ε-neighbourhood of q_s. This is the topological property (6) and if *CC-Steer* is modified so as to compute εCC path when the start and goal configurations are close, it will yield a complete path planner when used within a global path planning scheme.

6 *CC-Steer* and ACA

This section shows how to use *CC-Steer* in a global planning scheme, namely the Ariadne's Clew Algorithm [14].

6.1 General presentation of ACA

ACA is composed of two sub-algorithms: SEARCH and EXPLORE. SEARCH is a predicate that uses a local planner *local_planner* to answer the query "Can we reach the goal region $X_\bullet$ from q_i using the local planner?".

In order to build a complete planner, the second algorithm, called EXPLORE, is used. While the purpose of SEARCH was to directly look for a path from q_i to $X_\bullet$, the purpose of EXPLORE is to compute a representation of the accessible region from a given initial configuration $q_\circ$. This representation is made by placing landmarks, *i.e.* configurations, all over the connected component of the search space containing $q_\circ$. EXPLORE generates new landmarks starting with $q_\circ$ as the first landmark. At step i, the new landmark q_i is generated as far as possible from the previously placed landmarks $\mathcal{L}_m = \{q_0, q_1, .., q_{i-1}\}$ and such that a path from one landmark in $\mathcal{L}_m$ to q_i is known. Once the SEARCH algorithm responds positively to the query, a path is constructed by using the information (the path) provided by the local planner (going from q_i to the goal region) and the information provided by EXPLORE (the path from the initial configuration to q_i).

In the following, we detail EXPLORE. Let $\Omega(X, q)$ denote the set of free paths in X beginning at $q \in X$. We denote by $\mathcal{L}_m$ the set of existing landmarks at step m, and by $\mathcal{PL}_m$ the set of paths which start from a configuration corresponding to one of the landmarks q_i, *i.e.* $\mathcal{PL}_m = \bigcup_{j=1}^{m} \Omega(X, q_j)$. EXPLORE is posed as an optimization problem over a suitable space of paths. We initialize the set of landmarks to $\mathcal{L}_1 = \{q_1\}$ where $q_1 = q_\circ$. Let ρ_1 be the path such that its endpoint $\rho_1(1)$ is the farthest from $\mathcal{L}_1$: $\rho_1 : \max_{\rho \in \mathcal{PL}_1} dist(\rho(1), q_1)$.

The endpoint of ρ_1 becomes the second landmark, *i.e.* $q_2 = \rho_1(1)$ and $\mathcal{L}_2 = \mathcal{L}_1 \cup \{q_2\}$. For the next iteration, the third landmark is placed by maximizing the distance from $\mathcal{L}_2$ over the space of paths starting from either landmarks in the set

$$\rho_2 : \max_{\rho \in \mathcal{PL}_2} dist(\rho(1), \mathcal{L}_2) = \max_{\rho \in \mathcal{PL}_2} \min_{q_j \in \mathcal{L}_2} dist(\rho(1), q_j)$$

where $dist(x, Y) = \min_{y \in Y} dist(x, y)$. More generally, at each iteration m, the following maximization is performed:

$$\begin{cases} \text{Maximize } dist(\rho(1), \mathcal{L}_m) \\ \rho \in \mathcal{PL}_m \end{cases} \tag{7}$$

and we obtain the $(m+1)^{th}$ landmark by setting $q_{m+1} = \rho_m(1)$. EXPLORE can then be written as follows:

```
EXPLORE(m)
begin
    define f(ρ) = dist(ρ(1), L_m)
    ρ_m = GETMAX(PL_m, f)
    q_{m+1} = ρ_m(1)
    return(f(ρ_m))
end
```

where GETMAX(D, f) is an algorithm that returns an element of the domain D such that it maximizes the function $f : D \to \mathbb{R}$. At a given iteration, the minimum distance between any two landmarks is the resolution. It has been shown in [1] that, if there exists a $q \in X_\bullet$ such that q is reachable from $q_\circ$ within a resolution ε, there exists a finite n such that EXPLORE will place a landmark within a distance ε from $X_\bullet$ at the n^{th} iteration. EXPLORE is thus resolution complete. In the case where the local planner is guaranteed to find a solution within a distance ε, it is clear that by combining SEARCH and EXPLORE, we obtain a complete algorithm. Now, using SEARCH and EXPLORE, we define ACA.

```
ACA(X, q_o, X_•, ε)
 q_1 = q_o; L = {q_1}; m = 1;
 ε_m = ∞; path_found = false;
 while( ¬path_found and ε_m < ε );
      path_found = SEARCH(q_m, X_•);
      if (¬path_found)
        ε_{m+1} = EXPLORE(m);
        m = m + 1;
        L_m = L_{m-1} ∪ {q_m};
      endif
 endwhile
 if (path_found) return(history_of(q_m));
 else return(Null)
end
```

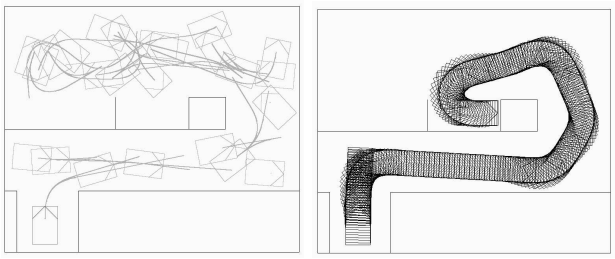

Fig. 11: An example of a path planned by ACA: the landmarks (left) and the computed path (right).

The "*history_of*" function is simply the concatenation of the paths between the ancestors of q_m and the path found by SEARCH.

6.2 Implementing ACA with *CC-Steer*

As it must be clear now, an implementation of ACA essentially requires three components: (i) the local planner *local_planner*,(ii) the set of paths $\Omega(X, q)$ and (iii) the optimization technique GETMAX(D, f). In our current implementation, the local planner is *CC-Steer* completed with a collision checker. The local planner uses *CC-Steer* to compute a path between q_i and the goal configuration. If this path is collision-free, the local planner returns the goal configuration, otherwise it returns the last free configuration along the path. Similarly, the set $\Omega(\mathcal{C}, q)$ can be represented by using $\mathcal{C}^0 = I\!R^2 \times [0, 2\pi[$. The basic idea is very simple. Let $Steer(q, q_f)$ be the path from q to q_f computed by *CC-Steer*, then it is possible to define $\Omega(\mathcal{C}, q) = \{Steer(q, q_f) \in \mathcal{C}_{free} | q_f \in \mathcal{C}^0\}$. The great advantage of this is that a *CC-Steer* path, starting from a configuration q, can be represented by a vector $q_f \in \mathcal{C}^0$ this is important for the implementation of GETMAX(D, f). Note that the ideal version, of EXPLORE requires a global optimization on GETMAX(D, f), a complex problem in itself. Rather than a using an ideal version it is possible to estimate the true maximum by using optimization technique such as Genetic Algorithms [14] or quasi-random methods for global optimization [1]. In our implementation, we use the second technique. It has been shown in [1] that, by using these kind of optimization, the implementation of the Ariadne's Clew Algorithm is probabilistically complete.

Figure 11 shows an example of a path obtained by our planner. The final path is a simplification of the path returned by ACA. In short, this path is obtained by eliminating redundant landmarks and then by eliminating useless configurations between two CC paths. Figure 12 shows a second example in the parking lot at INRIA Rhône-Alpes. The set of landmarks are shown in grey. Note that *CC-Steer* permits to place landmarks very far from each other, few landmarks suffice to cover the whole space and the resulting path looks quite natural.

7 Related Work and Discussion

As mentioned in §1, most of the research works that plan collision-free paths for car-like vehicles compute paths

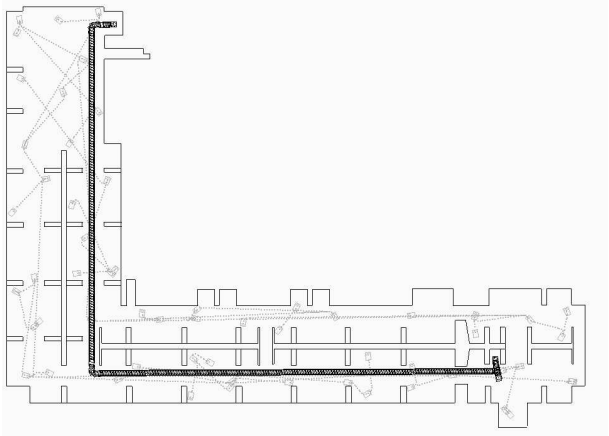

Fig. 12: Planning a path in parking lot environment

made up of line segments connected with tangential circular arcs of minimum radius, *e.g.* [8, 21]. Such paths are locally optimal in length [6, 16], but their curvature is discontinuous.

A first class of works dealing with continuous-curvature paths have studied the nature of optimal paths (*cf.* §2).

A second class of works dealing with continuous-curvature paths falls into the 'path generation' class: collision avoidance is not addressed. They usually make simplifying assumptions (on the respective positions of the configurations that are to be connected, on the type of path generated, etc.). It is nonetheless interesting to review these works so as to gain insight on the type of curves that could be used for a car-like vehicle. The curves used fall into two categories: (1) curves whose coordinates have a closed-form expression such as B-splines [10], quintic polynomials [22] or polar splines [15], and (2) parametric curves whose curvature is a function of their arc length such as clothoids [13], cubic spirals [9] or intrinsic splines [5]. The main advantage of the closed-form curves is that their coordinates can be obtained directly whereas they are derived by integration for the parametric curves. On the other hand, with parametric curves, it is much easier to take into account the upper-bounded curvature constraint. Finally, from a control standpoint, the curvature profile of the parametric curves make them easy to track (the variation of the curvature corresponds to the variation of the front wheels orientation).

A third class of works dealing with continuous-curvature paths have appeared more recently, it stems from the introduction of control theory techniques to address nonholonomy. The issue here is to study the controllability of nonholonomic systems so as to derive steering methods to move such systems between two configurations. The results obtained are more general in scope and concern different classes of nonholonomic systems. Several interesting results have been obtained for vehicles without upper-bounded curvature pulling one or several trailers. Steering methods exploiting the properties of such a system

(chained form, flatness), and yielding continuous-curvature paths have been proposed (*cf.* the review made in [12]).

One of these techniques has recently been applied to the case of the car, *i.e.* with upper-bounded curvature [11]. This work is the closest to the one presented in this paper. Both rely upon a steering method yielding complete path planners. Should we compare them, we would advocate that *CC-Steer* computes paths that are easier to turn into a trajectory and thus easier to track. Indeed, each component of a CC path is simple and can be related to the driving and steering velocity of the car straightforwardly.

Acknowledgements. This work was partially supported by the French programme "La Route Automatisée": `<http://www.lara.prd.fr/>`.

References

[1] J. M. Ahuactzin and K. Gupta. The kinematic roadmap: A novel motion planning based approach for inverse kinematics of redundant manipulators. *IEEE Transactions on Robotics and Automation*, 15(5), 1999.

[2] J.-D. Boissonnat, A. Cérézo, and J. Leblond. A note on shortest paths in the plane subject to a constraint on the derivative of the curvature. Research Report 2160, Inst. Nat. de Recherche en Informatique et en Automatique, Rocquencourt (FR), January 1994.

[3] A. De Luca, G. Oriolo, and C. Samson. Feedback control of a nonholonomic car-like robot. In J.-P. Laumond, editor, *Robot motion planning and control*, volume 229 of *Lecture Notes in Control and Information Science*, pages 171–253. Springer, 1998.

[4] E. Degtiariova-Kostova and V. Kostov. Irregularity of optimal trajectories in a control problem for a car-like robot. Research Report 3411, Inst. Nat. de Recherche en Informatique et en Automatique, April 1998.

[5] H. Delingette, M. Hébert, and K. Ikeuchi. Trajectory generation with curvature constraint based on energy minimization. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 206–211, Osaka (JP), November 1991.

[6] 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*, 79:497–517, 1957.

[7] Th. Fraichard, A. Scheuer, and R. Desvigne. From Reeds and Shepp's to continuous-curvature paths. In *Proc. of the IEEE Int. Conf. on Advanced Robotics*, pages 585–590, Tokyo (JP), October 1999.

[8] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 2–7, Scottsdale, AZ (US), May 1989.

[9] Y. Kanayama and B. I. Hartman. Smooth local path planning for autonomous vehicles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 1265–1270, Scottsdale, AZ (US), May 1989.

[10] K. Komoriya and K. Tanie. Trajectory design and control of a wheel-type mobile robot using B-spline curve. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 398–405, Tsukuba (JP), September 1989.

[11] F. Lamiraux and J.-P. Laumond. Smooth motion planning for car-like vehicles. In *Proc. of the Int. Conf. on Intelligent Autonomous Systems*, pages 1005–1012, Venezia (IT), July 2000.

[12] J.-P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In J.-P. Laumond, editor, *Robot motion planning and control*, volume 229 of *Lecture Notes in Control and Information Science*, pages 1–53. Springer, 1998.

[13] R. Liscano and D. Green. Design and implementation of a trajectory generator for an indoor mobile robot. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 380–385, Tsukuba (JP), September 1989.

[14] E. Mazer, J.-M. Ahuactzin, and P. Bessière. The Ariadne's Clew Algorithm. *Journ. of Artificial Intelligence Research*, 9:295–316, July-December 1998.

[15] W. L. Nelson. Continuous curvature paths for autonomous vehicles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 1260–1264, Scottsdale, AZ (US), May 1989.

[16] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[17] A. Scheuer and Th. Fraichard. Collision-free and continuous-curvature path planning for car-like robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 867–873, Albuquerque, NM (US), April 1997.

[18] A. Scheuer and Th. Fraichard. Continuous-curvature path planning for car-like vehicles. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 997–1003, Grenoble (FR), September 1997.

[19] A. Scheuer and Ch. Laugier. Planning sub-optimal and continuous-curvature paths for car-like robots. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 25–31, Victoria, BC (CA), October 1998.

[20] P. Souères and J.D. Boissonnat. Optimal trajectories for nonholonomic mobile robots. In J.-P. Laumond, editor, *Robot motion planning and control*, volume 229 of *Lecture Notes in Control and Information Science*, pages 93–170. Springer, 1998.

[21] P. Svestka and M. H. Overmars. Probabilistic path planning. In J.-P. Laumond, editor, *Robot motion planning and control*, volume 229 of *Lecture Notes in Control and Information Science*, pages 255–304. Springer, 1998.

[22] A. Takahashi, T. Hongo, and Y. Ninomiya. Local path planning and control for AGV in positioning. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 392–397, Tsukuba (JP), September 1989.