Planning Continuous-Curvature Paths for Car-Like Robots

CONFERENCE PAPER · DECEMBER 1996

DOI: 10.1109/IROS.1996.568985 · Source: IEEE Xplore

IEEE/RSJ Int. Conf. on Intelligent Robots and Systems
November 4-8, 1996. Osaka, Japan. Volume 3, pp.1304–1311

Planning Continuous-Curvature Paths for Car-Like Robots

A. Scheuer and Th. Fraichard
INRIA[a] Rhône-Alpes — GRAVIR[b]
ZIRST. 655 avenue de l'Europe. 38330 Montbonnot Saint Martin. France
[Alexis.Scheuer, Thierry.Fraichard]@inria.fr

November 18, 1996

Abstract

This paper presents a Continuous-Curvature Path Planner (CCPP) for a car-like robot. Previous collision-free path planners for car-like robots compute paths made up of straight segments connected with tangential circular arcs. The curvature of this type of path is discontinuous so much so that if a car-like robot were to actually follow such a path, it would have to stop at each curvature discontinuity so as to reorient its front wheels. CCPP is one of the first planner to compute collision-free paths with continuous curvature profiles. These paths are made up of clothoïd arcs, i.e. curves whose curvature is a linear function of their arc length. CCPP uses a general planning technique called the Ariadne's Clew algorithm [17]. It is based upon two complementary functions: Search *and* Explore. Explore *builds an approximation of the region of the configuration space reachable from a start configuration by incrementally placing a set of reachable landmarks in the configuration space.* Search *checks the existence of a solution path between a landmark newly placed and the goal configuration.*

Keywords — mobile-robot, path-planning, non-holonomic-system.

Acknowledgements — this work was supported by the INRIA-INRETS[c] Praxitèle programme on urban public transports.

[a]Inst. Nat. de Recherche en Informatique et en Automatique.
[b]Lab. d'informatique GRAphique, VIsion et Robotique de Grenoble.
[c]Inst. Nat. de Recherche sur les Transports et leur Sécurité.

Planning Continuous-Curvature Paths for Car-Like Robots

A. Scheuer and Th. Fraichard

INRIA* Rhône-Alpes — GRAVIR†

ZIRST. 655 avenue de l'Europe. 38330 Montbonnot Saint Martin. France

[Alexis.Scheuer, Thierry.Fraichard]@inria.fr

Abstract

This paper presents a Continuous-Curvature Path Planner (CCPP) for a car-like robot. Previous collision-free path planners for car-like robots compute paths made up of straight segments connected with tangential circular arcs. The curvature of this type of path is discontinuous so much so that if a car-like robot were to actually follow such a path, it would have to stop at each curvature discontinuity so as to reorient its front wheels. CCPP is one of the first planner to compute collision-free paths with continuous curvature profiles. These paths are made up of clothoïd arcs, i.e. curves whose curvature is a linear function of their arc length. CCPP uses a general planning technique called the Ariadne's Clew algorithm [17]. It is based upon two complementary functions: Search *and* Explore. Explore *builds an approximation of the region of the configuration space reachable from a start configuration by incrementally placing a set of reachable landmarks in the configuration space.* Search *checks the existence of a solution path between a landmark newly placed and the goal configuration.*

1 Introduction

Ever since Laumond's pioneering paper in 1986 [14], a lot of research works have addressed path planning[1] for car-like robots[2] (the reader is referred to [13] for a review on this topic). The main difference between a car-like robot and, say, a manipulator arm, is that a car-like robot is subject to non-holonomic constraints[3], that restricts the set of its admissible directions of motion. Thus a car-like robot can only move forward or backward in a direction perpendicular to the orientation of the rear wheels axle; besides its turning radius is lower bounded.

Focusing on the research works that actually plan collision-free paths for car-like robots, i.e. paths that avoid collision with the obstacles present in the environment of the robot, it appears that all these works compute paths made up of straight segments connected with tangential circular arcs of minimum radius, e.g. [2, 9, 6, 16]. The primary reason for this is that it has been shown that the shortest path for a car-like robot between two configurations is such a path [4, 19]. The secondary reason is that these paths are easy to deal with from a computational point of view.

However it is important to note that the curvature of this type of path is discontinuous; the discontinuities occurring at the transitions between segments and arcs. Accordingly, when it is time for the robot to actually follow such a path, it will have to stop at each transition so as to reorient its front wheels. This is not really a problem when the path contains both forward and backward motions, but when it comes to smooth paths, i.e. paths without manœuvres, it does not make much sense to stop every now and then so as to reorient the wheels. Planning smooth paths being our main concern, we aim at designing a path planner for car-like robots that determines *smooth*, *collision-free* and *continuous-curvature* paths. This continuous-curvature path planner (CCPP) is described in this paper and, to the best of our knowledge, it is the first of its kind.

The paths computed by CCPP are made up of *elementary paths*, where an elementary path is the concatenation of two symmetric clothoïd arcs (a clothoïd is a curve whose curvature is a linear function of its arc length). The elementary paths of a planned path are connected so as to form a smooth and continuous-curvature curve; besides they are collision-free with respect to the obstacles present in the environment of the robot.

In order to compute the sequence of elementary paths connecting a start and a goal configuration of the robot, CCPP uses a general planning technique called the Ariadne's Clew algorithm [17]. It is based upon two complementary functions: Search and Explore. Explore builds an approximation of the region of the configuration space reachable from the start configuration by incremen-

*Inst. Nat. de Recherche en Informatique et en Automatique.

†Lab. d'informatique GRAphique, VIsion et Robotique de Grenoble.

[1] I.e. computing a geometric path between a start and a goal configuration of a robot.

[2] I.e. a mobile robot with two directional front wheels and two rear wheels.

[3] A kinematic constraint is a relation involving the configuration parametres of the robot and their derivatives. When the derivative terms cannot be integrated, the constraint is non-holonomic (see [13, chapter 9] for more details).

tally placing a set of reachable landmarks in the configuration space; it does so by solving an optimization problem over the set of elementary paths starting from one of the existing landmarks. The purpose of SEARCH is to check the existence of a simple solution path between a landmark newly placed and the goal configuration; it ensures the termination of the algorithm.

Outline of the paper. After a short review of the works related to the problem at hand (§2), the problem to be solved is formally stated (§3). Then the solution algorithm we have developed is presented (§4) along with experimental results (§5).

2 Related Works

As mentioned earlier, all the research works on collision-free path planning for car-like robots compute paths made up of straight segments connected with tangential circular arcs of minimum radius, e.g. [2, 9, 6, 16]; this type of path does not meet the continuous-curvature property. Ref. [5] does address continuous-curvature path planning in the presence of obstacles but it does so for a robot slightly different from a car-like robot since its turning radius is not lower bounded: in this case, the authors show that the transition between two segments can be done with clothoïd arcs whose distance to the reference segments is never greater than a given tolerance ϵ. This property permits then to design an algorithm that takes as input a collision-free polygonal line and transforms it into a collision-free continuous-curvature path. Unfortunately this property is no longer true for a robot whose turning radius is lower bounded.

On the other hand, several works compute continuous-curvature paths but they do so without taking into account the obstacles present in the environment. These works do not address path planning in the classical sense of the word; they are more concerned with issues related to the command of mobile robots. It is nonetheless interesting to review them so as to gain an insight into the type of curve that could be used. A few authors use curves as different as B-splines [12], quintic polynomials [22] or polar splines [18]. However it appears that the most popular curves are by far curves whose curvature is a polynomial function of their arc length such as clothoïds [8, 11, 21], cubic spirals [10], or, more generally, intrinsic splines [3]. Although there are no closed-form expressions for these curves, they are interesting because they have a simple curvature profile that makes them easy to track. It is this property that led us to select the clothoïds, i.e. the simplest among this type of curve, for our path planning purposes.

3 Statement of the Problem

In this section, we start by presenting the model of the robot, its non-holonomic kinematic constraints and its workspace. Then we define what a feasible path for a car-like robot is. Finally we state the path planning problem to be solved formally.

3.1 The Robot and Its Workspace

Let $\mathcal{A}$ be a car-like robot. It moves on a planar workspace $\mathcal{W} \equiv \mathbb{R}^2$ that is cluttered up with a set of stationary obstacles $\mathcal{B}_i, i \in \{1, \ldots, b\}$, modelled as polygonal regions of $\mathcal{W}$. $\mathcal{A}$ is modelled as a rigid body moving on the plane. It is supported by four wheels making point contacts with the ground. $\mathcal{A}$ has two rear wheels and two directional front wheels. A configuration of $\mathcal{A}$ is defined by a triple $(x, y, \theta) \in \mathbb{R}^2 \times [0, 2\pi[$ where (x, y) are the coordinates of the rear axle midpoint R and θ the orientation of $\mathcal{A}$, i.e. the angle between the x axis and the main axis of $\mathcal{A}$ (Fig. 1).

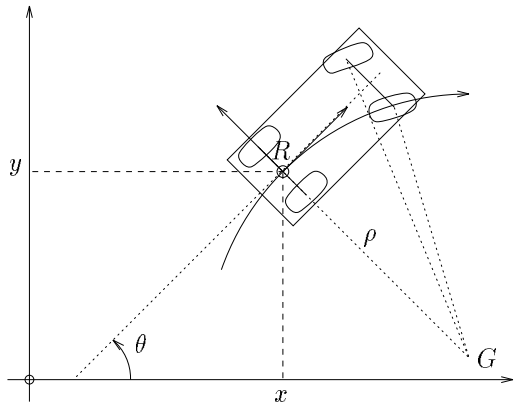

Figure 1: a car-like robot.

A body moving on the plane has only one centre of rotation. Under perfect rolling assumption, a wheel must move in a direction which is normal to its axle. Therefore, when $\mathcal{A}$ is moving, the axles of its wheels intersect at G, the centre of rotation of $\mathcal{A}$. The orientation of the rear wheels being fixed, G must be located on the rear wheels axle (possibly at an infinite distance) and R must move in a direction which is normal to this axle. In other words, the following constraint holds:

$$-\dot{x} \sin\theta + \dot{y} \cos\theta = 0 \tag{1}$$

Besides, owing to the fact that the front wheels orientation is mechanically limited, the distance ρ between R and G, i.e. the curvature radius at point R, is lower bounded by a certain value ρ_{min} and the following constraint holds:

$$\dot{x}^2 + \dot{y}^2 - {\rho_{min}}^2 \dot{\theta}^2 \geq 0 \tag{2}$$

Ref. [2] shows that (1) and (2) are non-holonomic; the derivative terms cannot be eliminated. Accordingly they restrict the shape of the feasible paths for $\mathcal{A}$ by reducing its set of admissible velocity vectors $(\dot{x}, \dot{y}, \dot{\theta})$ (see [13, chapter 9] for more details).

3.2 Feasible Paths

Let $\mathcal{C}$ be the configuration space of $\mathcal{A}$, i.e. the $xy\theta$-space. A path for $\mathcal{A}$ is a continuous sequence of configurations, i.e. a curve in $\mathcal{C}$. However it stems from (1) that θ is always tangent to the xy-curve traced by R on $\mathcal{W} \equiv \mathbb{R}^2$.

Therefore there is a one-to-one relationship between a path in $\mathcal{C}$ and its projection on $\mathcal{W}$ [15]. It is therefore possible to represent a path for $\mathcal{A}$ by a continuous xy-curve. Let Π be such a xy-path. It is *feasible*, i.e. it respects (1) and (2), if and only if a) its tangent direction is piecewise continuous with opposite semi-tangents at the cusp points[4], and b) the curvature at each point of Π is less than $1/\rho_{min}$, where ρ_{min} is the minimum turning radius of $\mathcal{A}$. In addition Π is *smooth*, i.e. without backing-up manœuvres, if and only if its tangent direction is continuous [15].

Thus paths made up of straight segments connected with tangential circular arcs of minimum radius are feasible. They can also be smooth if their tangent direction is continuous. As mentioned earlier however, the curvature of this type of path is discontinuous; the discontinuities occurring at the transitions between segments and arcs. Accordingly, when $\mathcal{A}$ follows such a path, it has to stop at each curvature discontinuity in order to reorient its front wheels (an instantaneous reorientation of the wheels is physically impossible). The continuity of the curvature is therefore a desirable property for paths. This leads us to define a *continuous-curvature* path as a path whose curvature profile is continuous[5].

Let us define the representation of a finite-length path Π for $\mathcal{A}$. It is a pair (l, κ) where l is a positive arc length and κ: $[0, l] \longrightarrow \mathbb{R}$ a curvature profile. The tangent direction along the path is given by:

$$\theta(s) = \int_0^s \kappa(t)dt$$

and a point on Π is given by:

$$x(s) = \int_0^s \cos\theta(t)dt \quad \text{and} \quad y(s) = \int_0^s \sin\theta(t)dt$$

with $s \in [0, l]$. Accordingly, a configuration of $\mathcal{A}$ along Π is given by $q(s) = (x(s), y(s), \theta(s))$.

3.3 The Path Planning Problem

In summary, the problem to be solved can be stated formally as follows: let (x_s, y_s, θ_s) be $\mathcal{A}$'s start configuration and (x_g, y_g, θ_g) be its goal. A path $\Pi = (l, \kappa)$ is a solution to the problem at hand if and only if:

- End conditions:

$$\begin{cases} q(0) = (x_s, y_s, \theta_s) \\ \kappa(0) = 0 \end{cases} \quad \text{and} \quad \begin{cases} q(l) = (x_g, y_g, \theta_g) \\ \kappa(l) = 0 \end{cases}$$

  Note that we impose the curvature to be null at the ends of the path.

- Π is continuous-curvature, smooth and feasible. In other words, κ is continuous and such that:

$$\forall s \in [0, l], |\kappa(s)| \leq \frac{1}{\rho_{min}}$$

  where ρ_{min} is the minimum turning radius of $\mathcal{A}$.

- Π is collision-free:

$$\forall i \in \{1, \ldots, b\}, \forall s \in [0, l], \mathcal{A}(q(s)) \cap \mathcal{B}_i = \emptyset$$

  where $\mathcal{A}(q(s))$ denotes the region of $\mathcal{W}$ occupied by $\mathcal{A}$ when in the configuration $q(s)$.

[4] The passage from forward to backward motion occurs at these cusp points.

[5] Note that when the path contains cusp points, i.e. passage from forward to backward motion, it would be possible to reorient the front wheels at these cusp points where the speed becomes null. Thus a continuous-curvature path could alternatively be defined as a path whose curvature profile is piecewise continuous; the discontinuities occurring at the points where the tangent direction changes its sign.

4 The Path Planning Algorithm

In this section, we start by giving an overview of the continuous-curvature path planner (CCPP) we have developed in order to solve the problem at hand. Then we detail three key points of CCPP, namely the so-called elementary paths that are used to build the solution path and the two complementary functions, Search and Explore, that lies at the heart of the algorithm.

4.1 Outline of the Algorithm

We have decided to adapt an existing planning algorithm in order to solve the problem at hand: the so-called Ariadne's Clew algorithm [17]. To begin with, let us briefly recall the main features of this algorithm. The Ariadne's Clew algorithm is based upon two complementary functions: Search and Explore. Search is a local path planner; it looks for a solution path between two given configurations. In order to be efficient, Search narrows its search to a particular subset of the whole set of solution paths. This efficiency is thus obtained at the expense of completeness. Therefore, in order to have a complete planner (a resolution-complete planner actually), Search is completed by Explore whose purpose it to build an approximation of the region of the configuration space reachable from the start configuration q_s. Explore builds an approximation of the region of the configuration space reachable from q_s by incrementally placing a set of landmarks in $\mathcal{C}$ in such a way that a solution path from q_s to any landmark is known. Explore places the landmarks as far as possible from one another. In other words, when Explore looks for a new landmark, it aims at maximizing the minimum distance between the new landmark and the existing ones. This property guarantees the resolution-completeness of the algorithm since, in a finite number of iterations, Explore eventually places a landmark at a distance less than ϵ from any configuration of the region of $\mathcal{C}$ reachable from q_s. In summary, Explore solves an optimization problem over the set of solution paths starting from one of the existing landmarks. Once again, for the sake of efficiency, Explore narrows its optimization to a particular subset of the whole set of solution paths. The reader is referred to [1] for a complete presentation of the Ariadne's Clew algorithm and its properties. Two important features of this algorithm are worth recalling however:

- The configuration space $\mathcal{C}$ and its obstacles are never explicitly computed. This accounts for the efficiency of the algorithm.
- Because of the optimization performed in EXPLORE, the algorithm automatically adapts itself to the difficulty of a given planning problem.

Given SEARCH and EXPLORE, the Ariadne's Clew algorithm works in the following way: it incrementally builds a tree rooted at the start configuration q_s. A node of this tree, i.e. a landmark, is a collision-free configuration while an arc represents a collision-free feasible paths between two landmarks. At each iteration, EXPLORE computes a new landmark. SEARCH is used afterwards to check whether it is possible to reach the goal configuration q_g from this new landmark. More precisely, let Λ be the tree of landmarks and ϵ be the desired planning resolution. Let λ_n denote the latest landmark computed by EXPLORE, the algorithm can then be sketched as follows:

The Ariadne's Clew algorithm

1. **Init**: $\Lambda = \{q_s\}, \lambda_n = q_s$.
2. **If** SEARCH finds a solution path between λ_n and q_g **then** goto 7.
3. $\lambda_n =$ EXPLORE.
4. **If** the distance from λ_n to the other landmarks is less than ϵ **then** exit, there is no solution.
5. $\Lambda = \Lambda \cup \{\lambda_n\}$.
6. Goto 2.
7. **End**: return the path from q_s to q_g using Λ's tree structure and the result of SEARCH.

SEARCH and EXPLORE functions both operate on a subset of the whole set of solution paths. EXPLORE solves its optimization problem over the set of so-called elementary paths: an elementary path is the concatenation of two symmetric clothoïd arcs, i.e. curves whose curvature is a linear function of their arc length. As for SEARCH, it looks for a solution path made up of two elementary paths. The elementary paths are presented in the next section. Then EXPLORE and SEARCH are detailed.

4.2 Elementary Paths

A clothoïd is a curve whose curvature profile κ is a linear function of its arc length s, i.e. $\kappa(s) = \sigma s$, where σ is a real constant called the *sharpness* of the clothoïd. Now let us consider the path $\Pi = (l, \kappa)$ such that:

$$\begin{cases} \kappa(s) = \sigma s, & \forall s \in [0, l/2[\\ \kappa(s) = \sigma(l - s), & \forall s \in [l/2, l] \end{cases}$$

Π is an *elementary path*. When σ is strictly positive, its curvature increases linearly from zero to its maximum $\kappa(l/2) = \sigma l/2$, then it decreases linearly back to zero. Note that Π's curvature is symmetric. Accordingly the resulting path is symmetric and [10] shows that $\theta(0)$ and $\theta(l)$, i.e. the orientations of $\mathcal{A}$ at the start and end of Π, are symmetric with respect to the line that joins $q(0)$ and $q(l)$. This leads us to define the notion of symmetric configurations. Two configurations $q_a = (x_a, y_a, \theta_a)$ and $q_b = (x_b, y_b, \theta_b)$ are *symmetric* if and only if:

$$(x_b - x_a)\sin\left(\frac{\theta_b + \theta_a}{2}\right) = (y_b - y_a)\cos\left(\frac{\theta_b + \theta_a}{2}\right) \quad (3)$$

An elementary path is depicted in Fig. 2. $q(0)$ and $q(l)$ are symmetric. When an elementary path respects the minimum turning radius constraint (2), i.e. is such that $|\kappa(l/2)| \leq 1/\rho_{min}$, then it is obviously smooth, continuous-curvature and feasible. If an elementary path starting at a given configuration $q = q(0)$ is furthermore collision-free then it is an *elementary solution path* between $q(0)$ and $q(l)$. Henceforth an elementary path is denoted by Ξ and defined by the triple (q, l, σ). Collision checking for an elementary path Ξ is done by computing the volume swept by $\mathcal{A}$ when it moves along Ξ and testing whether it intersects one of the obstacle $\mathcal{B}_i, i \in \{1, \ldots, b\}$.

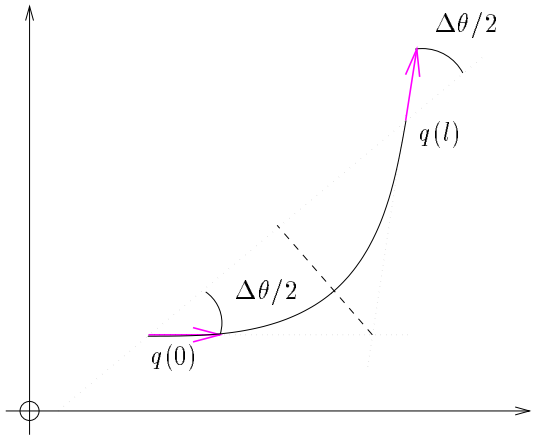

Figure 2: an elementary path, $\Delta\theta = |\theta(l) - \theta(0)|$.

The following theorem establishes the conditions of existence of an elementary path between two symmetric configurations. Due to lack of space, the proof is omitted here and the reader is referred to [20] for more details.

Theorem 1 *Let $q_a = (x_a, y_a, \theta_a)$ and $q_b = (x_b, y_b, \theta_b)$ be two symmetric configurations. Let us rewrite the vector $(x_b - x_a, y_b - y_a)$ in polar coordinates as $(r\cos(\theta_a + \alpha), r\sin(\theta_a + \alpha))$, with $r \in [0, +\infty[$ and $\alpha \in]-\pi, \pi]$. An elementary path Ξ that respects the minimum turning radius constraint (2) exists between q_a and q_b if and only if r and α respects the following constraints:*

$$or \begin{cases} |\alpha| \in [0, \theta_{root}[& and \quad r \geq 4\rho_{min}\sqrt{|\alpha|}D_1(|\alpha|) \\ |\alpha| = \theta_{root} & and \quad r = 0 \end{cases}$$

where D_1 is the function defined over $[0, \pi]$ as:

$$D_1(\beta) = \cos\beta \int_0^{\sqrt{\beta}} \cos u^2 du + \sin\beta \int_0^{\sqrt{\beta}} \sin u^2 du$$

and where θ_{root} is the unique root of D_1 over $]0,\pi]$ (cf. Fig. 3).

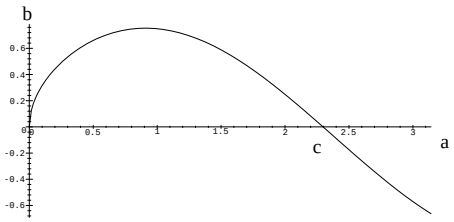

Figure 3: the diagram of the function D_1 over $[0,\pi]$.

Theorem 1 shows that the existence of an elementary path that respects the minimum turning radius constraint (2) between two symmetric configurations depends on r and α, i.e. their respective position. This is illustrated in Fig. 4 that depicts the set of symmetric configurations that are reacheable from the null configuration $(0,0,0)$. The unreachable symmetric configurations lies in the grey areas.

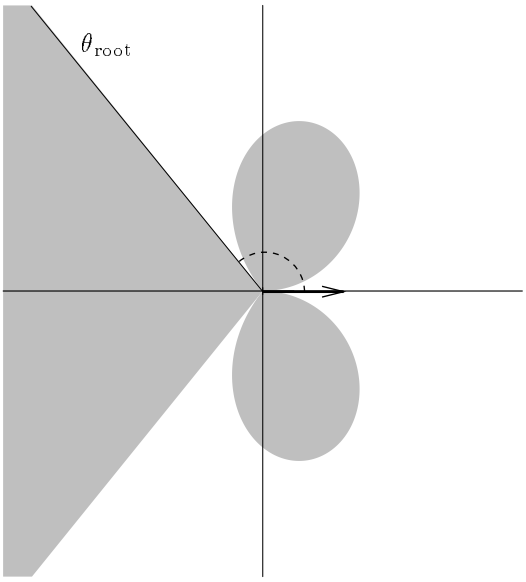

Figure 4: symmetric configurations unreachable from the null configuration $(0,0,0)$.

4.3 Explore

The purpose of Explore is to build an approximation of the region of the configuration space $\mathcal{C}$ reachable from the start configuration q_s. It does so by incrementally placing a set of landmarks in $\mathcal{C}$ in such a way that a solution path from q_s to any landmark is known (the landmarks are organized as a tree rooted at q_s). Explore places the landmarks as far as possible from one another; it maximize the minimum distance between the new landmark and the existing ones. A new landmark is the end configuration of a solution path starting from one of the existing landmarks. In other words, Explore has to solve an optimization problem over the set of solution paths starting from one of the existing landmarks. For the sake of efficiency, Explore narrows its optimization to a particular subset of the whole set of solution paths; namely the set of elementary solution paths presented earlier.

Formally Explore must solve the following optimization problem: let $\Lambda = \{\lambda_1 \dots \lambda_n\}$ be the current set of landmarks placed in $\mathcal{C}$. Let $ESP(\lambda)$ be the set of elementary solution paths starting from λ, and let $ESP^{\cup}(\Lambda)$ be defined as:

$$ESP^{\cup}(\Lambda) = \bigcup_{\lambda \in \Lambda} ESP(\lambda)$$

It is the set of paths considered by Explore. Let $e(\Xi)$ denote the end configuration of Ξ then Explore must determine the path $\Xi_* \in ESP^{\cup}(\Lambda)$ such that:

$$\min_{\lambda \in \Lambda} d(\lambda_*, \lambda) = \max_{\Xi \in ESP^{\cup}(\Lambda)} \left(\min_{\lambda \in \Lambda} d(e(\Xi), \lambda) \right) \qquad (4)$$

where $\lambda_* = e(\Xi_*)$.

In order to solve the optimization problem (4), a randomized optimization method, namely a genetic algorithm, is used. The reader is referred to [7] for a detailed presentation of genetic algorithms and their use as optimization tools.

4.4 Search

The purpose of Search is to find a solution path between two given configurations. In the framework of the Ariadne's Clew algorithm, Search does not have to be complete; a local path planner suffices even if it fails to find a solution in some cases. Actually efficiency is a more desirable property [1]. In a first phase and with this idea in mind, we have designed a very simple Search function: it checks for the validity of only one path between the two configurations. This particular path is called a *bi-elementary* path; it is the concatenation of two elementary paths. As shown earlier, elementary paths exist between symmetric configurations only. By combining two of these elementary paths, it becomes possible to link any pair of configurations. Given two configurations q_a and q_b, [10] shows that there is at least one intermediate configuration q_i such that q_i is symmetric with both q_a and q_b. Once q_i is known, it is easy to check for the existence of an elementary solution path from q_a to q_i and from q_i to q_b. If both paths exists then Search has succeeded otherwise it has failed. As simple as it may seem, Search combined with Explore gives nonetheless good results (cf. §5).

Fig. 5 depicts a bi-elementary path between two configurations q_a and q_b that are in general disposition, i.e. they are not symmetric and their orientations are not parallel. In this case, there is an infinite number of candidate intermediate configurations, they are located on a circle passing through q_a and q_b [10]. By using an iterative Newton

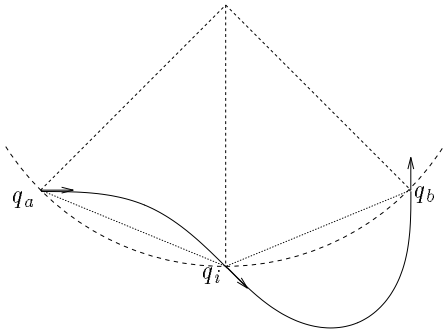

Figure 5: a bi-elementary path between q_a and q_b.

search algorithm, it is possible to compute the intermediate configurations q_i that minimizes the total length of the resulting bi-elementary path.

5 Experimental Results

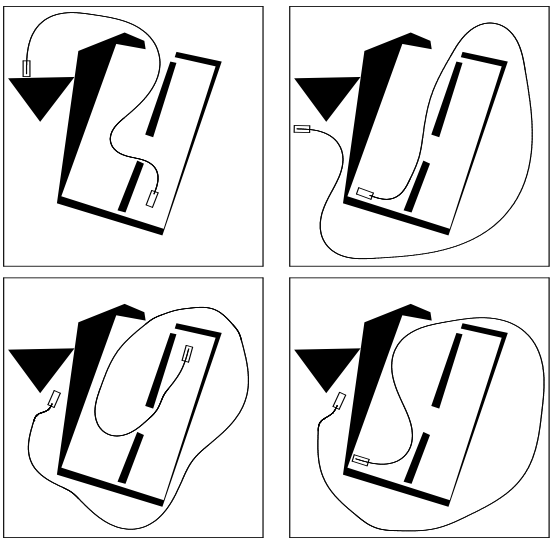

Figure 6: experiments with 4 obstacles.

A prototype of CCPP has been implemented in C and tested on a SUN Sparc 10 workstation. It has been tested on different workspaces generated manually or randomly. A few examples are depicted in Figs. 6, 7 and 8. Because of the stochastic nature of the optimization technique used by EXPLORE, namely a genetic algorithm, the results obtained are hard to evaluate. The running times and the number of landmarks generated are very different from one experiment to the next. Even when CCPP is run twice on the same experiment, i.e. same workspace and same start and goal configurations, the results obtained can differ significantly: see for instance the top part of Fig. 7. It took 250s. and 52 landmarks to determine the top left path and only 166s. and 29 landmarks for the top right example.

A summary of our experiments is shown in Tab. 1. For

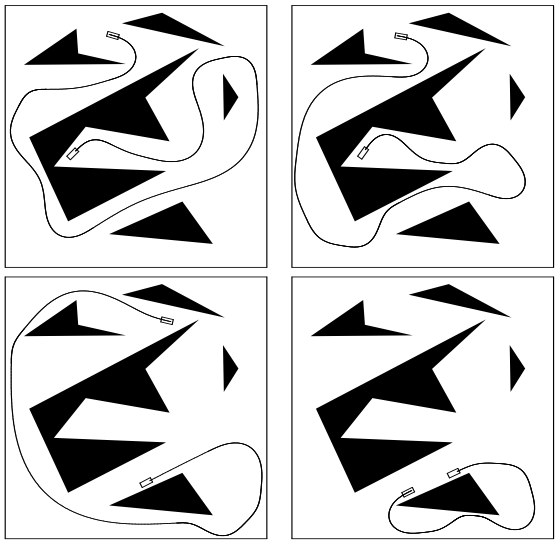

Figure 7: experiments with 5 obstacles.

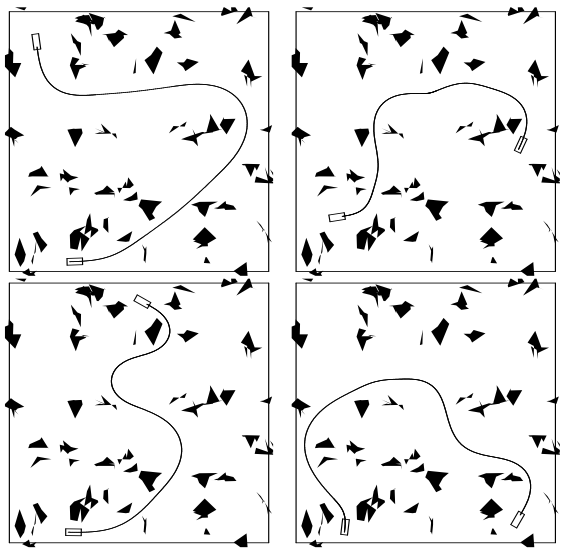

Figure 8: experiments with 50 obstacles.

each workspace (4, 5 and 50 obstacles), 15 experiments were made. The running time of the algorithm can be very high. Most of the time is spent in the optimization and the collision checking performed by EXPLORE. Accordingly the running time of the algorithm is directly related to the number of landmarks that EXPLORE has to determine which, in turn, is related to the 'complexity' of the workspace. Optimizing the current implementation of EXPLORE would definitely increase the overall efficiency of CCPP. However a more promising way to improve CCPP would be in increasing SEARCH's capabilities to find a path between two configurations (at present, SEARCH checks for the validity of only one path.). With a better SEARCH function, the number of necessary iterations could be reduced

	Running time (in s.)			Number of landmarks		
Workspace	min.	max.	average	min.	max.	average
4 obstacles	15	400	125	5	85	25
5 obstacles	15	300	150	5	60	33
50 obstacles	40	500	150	7	52	17
Over all tests	15	500	140	5	85	25

Table 1: summary of the experimental results.

dramatically. It is nonetheless interesting to notice that, as simple as SEARCH may be, CCPP is able to find a solution when it exists.

6 Conclusion and Discussion

In this paper, we have presented CCPP, one of the first path planner for car-like robots that computes collision-free paths with a continuous curvature profile. These paths are made up of *elementary paths*, where an elementary path is the concatenation of two symmetric clothoïd arcs (a clothoïd is a curve whose curvature is a linear function of its arc length). CCPP uses a general planning technique called the Ariadne's Clew algorithm [17]. It is based upon two complementary functions: SEARCH and EXPLORE. EXPLORE builds an approximation of the region of the configuration space reachable from a start configuration by incrementally placing a set of reachable landmarks in the configuration space. It does so by solving an optimization problem over the set of elementary paths starting from one of the existing landmarks. SEARCH checks the existence of a solution path between a landmark newly placed and the goal configuration; it ensures the termination of the algorithm.

CCPP has been implemented and tested successfully. The way it is implemented now, CCPP returns the first solution path found between two given configurations. In general, due to the nature of the Ariadne's Clew algorithm, this is not the shortest path. This point could be dealt with by a) turning the landmark tree structure into a graph tree structure and b) adding a post-processing step that would extract the shortest path from this graph. Furthermore the efficiency of CCPP could be improved by increasing SEARCH's capabilities to find a path between two configurations (at present, SEARCH checks for the validity of only one path.). These two points are currently under development.

Acknowledgements

This work was supported by the INRIA-INRETS[6] Praxitèle programme on urban public transports.

[6] Inst. Nat. de Recherche sur les Transports et leur Sécurité.

References

[1] J. M. Ahuactzin Larios. *Le Fil d'Ariane : Une Méthode de Planification Génerale. Application à la Planification Automatique de Trajectoires.* Thèse de doctorat, Inst. Nat. Polytechnique, Grenoble (F), September 1994.

[2] J. Barraquand and J.-C. Latombe. On non-holonomic mobile robots and optimal maneuvering. *Revue d'Intelligence Artificielle*, 3(2):77–103, 1989.

[3] 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.

[4] 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*, 79:497–516, 1957.

[5] S. Fleury, Ph. Souères, J.-P. Laumond, and R. Chatila. Primitives for smoothing paths of mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 832–839, Atlanta (GA), September 1993.

[6] Th. Fraichard. Smooth trajectory planning for a car in a structured world. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 318–323, Sacramento (CA), April 1991.

[7] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, 1989.

[8] J. Iijima, Y. Kanayama, and S. Yuta. A locomotion control system for mobile robots. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, Vancouver (CAN), 1981.

[9] P. E. 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), May 1989.

[10] 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), May 1989.

[11] Y. Kanayama and N. Miyake. Trajectory generation for mobile robots. In *Proc. of the Int. Symp. on Robotics Research*, pages 16–23, Gouvieux (FR), 1985.

[12] 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.

[13] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Press, 1990.

[14] J.-P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In *Proc. of the Int. Conf. on Intelligent Autonomous Systems*, pages 346–354, Amsterdam (NL), December 1986.

[15] J.-P. Laumond. Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, pages 1120–1123, Milan (I), August 1987.

[16] J.-P. Laumond, P. E. Jacobs, M. Taïx, and R. M. Murray. A motion planner for non-holonomic mobile robots. *IEEE Trans. Robotics and Automation*, 10(5):577–593, October 1994.

[17] E. Mazer, J.M. Ahuactzin, P. Bessière, and E.G. Talbi. Robot motion planning with the ariadne's clew algorithm. In *Proc. of the Int. Conf. on Intelligent Autonomous Systems*, Pittsburgh (PA), February 1993.

[18] 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), May 1989.

[19] 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.

[20] A. Scheuer and Th. Fraichard. Global continuous-curvature path planners for car-like robots. Research report, Inst. Nat. de Recherche en Informatique et en Automatique, Grenoble (F), 1996. To appear.

[21] D. Shin and S. Singh. Path generation for robot vehicles using composite clothoïd segments. Technical Report CMU-RI-TR-90-31, Carnegie Mellon Univ., Pittsburgh (PA), 1990.

[22] A. Takahashi, T. Hongo, and Y. Ninomiya. Local path planning and control for AGV in positionning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 392–395, Tsukuba (JP), September 1989.