# Path planning for a Parking Assistance System: Implementation and Experimentation

**C. Pradalier, S. Vaussier and P. Corke**
CSIRO ICT Centre
Autonomous Systems Laboratory
PO Box 883, Kenmore, Qld 4069, AUSTRALIA

## Abstract

Manoeuvre assistance is currently receiving increasing attention from the car industry. In this article we focus on the implementation of a reverse parking assistance and more precisely, a reverse parking manoeuvre planner. This paper is based on a manoeuvre planning technique presented in previous work and specialised in planning reverse parking manoeuvre. Since a key part of the previous method was not explicited, our goal in this paper is to present a practical and reproducible way to implement a reverse parking manoeuvre planner. Our implementation uses a database engine to search for the elementary movements that will make the complete parking manoeuvre. Our results have been successfully tested on a real platform: the CSIRO Autonomous Tractor.

## 1 Introduction

Manoeuvre assistance is currently receiving increasing attention from the car industry. For example such companies as Daimler-Chrysler[Franke, 2005] or Seeing Machines[SM, 2005] are developing technologies to assist the drivers in their driving tasks by increasing their perceptions and the speed of their actions. Full autonomy of a standard car is not something planned for the near future: the general trend is rather to keep the drivers in control, but to assist them. In this respect, power steering and ABS were some first steps in the direction of driving assistance.

On the other hand, path planning for car-like vehicles has been thoroughly studied in the robotics community. The main aim of this previous research has been vehicle autonomy, but parking assistance is also a possible application.

This paper is based on a path planning technique presented in [Paromtchik, 2004] and specialised in planning a reverse parking manoeuvre. Since a key part of the method in [Paromtchik, 2004] is not explicite, our goal in this paper is to present a practical and reproducible way to implement a reverse parking manoeuvre planner. Note that the techniques we present can be easily adapted to plan parallel parking manoeuvres.

Our paper is organised as follows: section 2 is a general introduction on path planning principles and related works. Section 3 presents the general principle of the reverse parking manoeuvre planner, and section 4 describes the specificities of our implementation. Finally the graphical user interface and our experiments are presented in 5 and 6.

## 2 Related works

### 2.1 General notions of path & trajectory planning

Let's start this section by some general notion of path planning. The general topic of path planning has been widely studied in the robotic community for several decades now. It is now a well established science with very nice results, especially in a static environment, however complex these may be. A nice starting point for reading about path planning is the classical [Latombe, 1992], which includes all the notion we will need for this article.

First let's define the context of path planning: a robot $R$, represented by a geometric shape is to move is an environment $W$ for which a geometric model is available.

**Configuration** $R$ has some degrees of freedom (for instance its position and orientation), and the state of these degrees of freedom will be assimilated to the state of $R$. Such a state is called a *configuration*. The space of all the possible states of $R$ is consequently called the *configuration space* $C$.

**Environment** It is assumed that the world $W$ can be partitioned into two sets of places: the one where an obstacle is present and the places free of obstacles, an obstacle being defined as a point of $W$ where no point

of $R$ can be. The part of $W$ without obstacles is called the *free space* $F$. The movement of $R$ in $W$ must be in $F$. Knowing $W$, a given configuration can represent a robot in the free space: it is then called a free – or safe – configuration.

**Goal** According to the task, a point in $W$ or $C$ will be specified as the goal or target configuration, that is the point that has to be reached by $R$ at the end of its movement in $W$.

**Path and trajectory** A movement of $R$ corresponds to a sequence of configurations. If this sequence is indexed by time, then we will call it a trajectory. Without timing, we just call it a path. A path or trajectory for which all the configurations are safe is called a safe path or a safe trajectory.

**Holonomy** We will not define this notion in detail here but rather try to give an intuitive feeling. At a given configuration $c$, a *holonomic* vehicle, can reach any configuration in a neighbourhood of $c$ with an elementary movement, whereas a non-holonomic vehicle cannot. A standard car is a non-holonomic vehicle since it cannot move sideways. A human, can be said holonomic when considering its movement on a floor-plan.

**Feasibility** The non-holonomy is a set of constraints on the dynamic of the vehicle. These constraints lead to the notion of *feasible path*, that is a path that can be effectively performed by a given vehicle with a given dynamic. For instance a path with a straight angle will not be feasible by a car, whatever the speed of the car. A trajectory with discontinuous speed, or a too strong acceleration, is generally not feasible.

**Path Planning** Given all these definitions, the goal of path planning is to find a feasible and safe trajectory going from the current configuration to a goal configuration. [Latombe, 1992; La Valle, 2006] give plenty of algorithms to solve the path planning problem.

### 2.2 Geometric-to-feasible path planning

A first class of approaches to path planning is usually a two step process: first a geometric path is computed, that links the current position to the goal, then the geometric path is approximated by a trajectory accounting for the dynamic constraints of the robot. Several methods are available to solve the first step: graph based such as the Voronoi graph[Hermosillo, 2003; Sekhavat *et al.*, 2001; Pradalier *et al.*, 2005], road maps[Latombe, 1992] or potential fields[Khatib, 1986] are the most often used. The main advantage of these methods are their ability to deal with very general movement, in any environment for any robot. The drawback, is that, even if the resulting movement is efficient and feasible it often looks unnatural or unpredictable. This is a problem in the context of Parking Assistance since the manoeuvre will have to be realised or supervised by a human operator, and in the context of human-machine interaction, a manoeuvre must be predictable for the human using the robot to feel safe.

### 2.3 Command-based path planning

Another class of approach to path planning relies on the definition of elementary movement or piece of trajectory. Such an elementary movement can be the application of a constant speed and rotation speed for a given time. This approach is used in [Paromtchik, 2004], [Pivtoraiko and Kelly, 2005]. Approaches based on clothoids such as [Fraichard and Scheuer, 2004] are also in this class since they build elementary movement with a continuous and bounded curvature.

The advantage of this class of path planning is that resulting trajectory are naturally feasible and in agreement with the dynamic of the vehicle. Nevertheless, the search for a global path in $W$ becomes more complex since the elementary paths have no direct representation in $W$. Plenty of methods have been proposed to overcome these difficulties. Among the most popular are the one based on Probabilistic Path Planning [Ahuactzin *et al.*, 1991; Overmars, 1992; Sekhavat *et al.*, 1996] or on Rapidly Expanding Trees [La Valle and Kuffner, 1999; La Valle, 2006].

The approach presented in the remainder of this paper is based on [Paromtchik, 2004], the main difference being the way the search for a feasible trajectory is performed. In the next section we will present the details of this method.

## 3 Parking Assistance System: task-oriented path planning

### 3.1 Specificity of the PAS

The path planning techniques presented in the previous section are general methods that can find motions for any kind of manoeuvres. Yet, in the case of a parking assistance system, a general techniques might not be optimal. Whether we are dealing with reverse parking or parallel parking, we have additional knowledge on the kind of environment we are navigating in, and the kind of manoeuvre we are looking for.

For instance, we know that a parking manoeuvre will be required, when the vehicle is on a road, and a free parking lot is close. So we can expect an environment similar to the one illustrated on fig. 2. Furthermore, a natural and predictable reverse parking manoeuvre has

three sections: an approach, a first forward motion to rotate the car to an oblique configuration, and a final reverse motion to enter the lot (see fig. 5).

In [Paromtchik, 2004], a parking assistance system was presented, making the best of the specific structure of the problem. The core of this PAS is an elementary movement description we will present now.

### 3.2 The vehicle model

The application we are considering in this work is designed for a car-like vehicle. This means we consider a vehicle with the following properties:

- It moves on a 2D surface.
- Its configuration is represented by a position $(x, y)$ and an orientation $\theta$. This position corresponds to the middle of the rear axle.
- Its controls are its speed $S$ and its steering angle of the front axle $\Phi$.
- Its dynamic is described by the following equations:

$$\dot{x} = S\cos(\theta) \quad (1)$$
$$\dot{y} = S\sin(\theta) \quad (2)$$
$$\dot{\theta} = S\frac{\tan(\Phi)}{L} \quad (3)$$

where $L$ is the distance between the front and rear axles. Figure 1 summarises these variables and their relations.

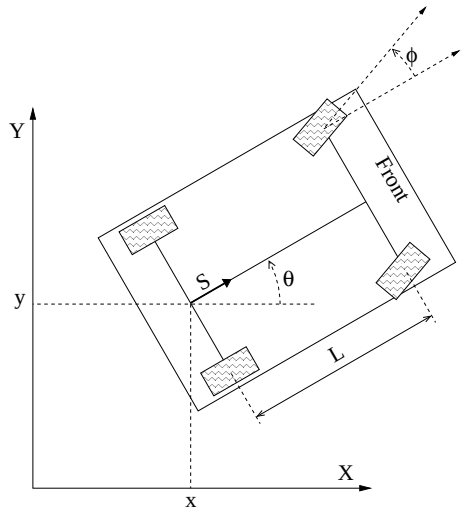


Figure 1: Model of a car-like vehicle

### 3.3 The environment model

The environment model we will present here is simplistic but can be fitted to any reasonable parking situation. There is two basic ideas behind the model:

- The parking lot is a rectangular area of free space adjacent to the road.

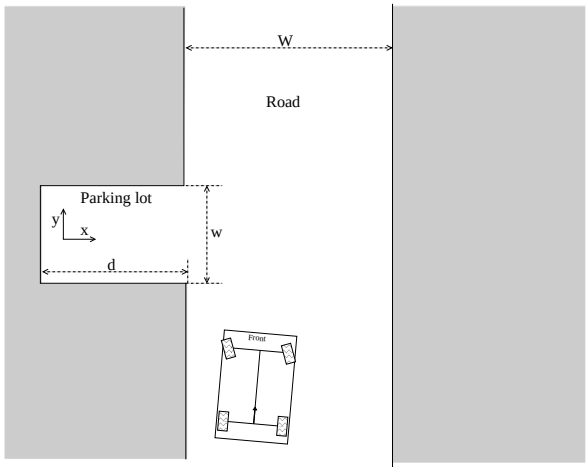


Figure 2: Environment model for a Parking Assistance System

- The road is a free space area delimited by two parallel lines.

Figure 2 illustrates this model.

Formally, the environment of a parking problem can be described by three parameters:

- the width of the lot $w$;
- the depth of the lot $d$;
- the width of the road $W$.

In order to simplify the representations, we will consider the origin of the world frame to be at the parked position, such that the final position of the car will always be $(0, 0)$ with a null orientation.

### 3.4 The elementary movement

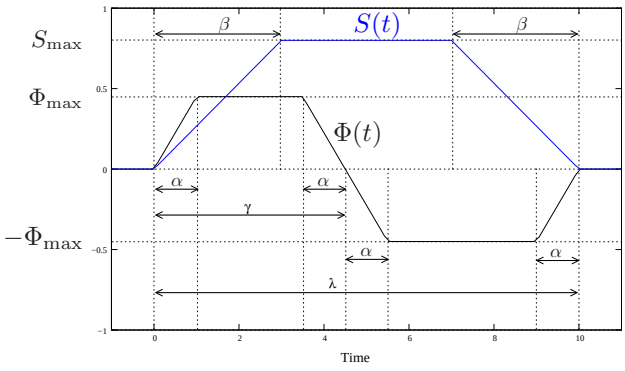


Figure 3: Elementary movement control profile

We define now the control profile depicted in figure 3. This profile represent the application of given control to the vehicle as a function of time and for a given duration. The model can be formalised using equations 4 and 5. It is parametrised by six parameters:

- $\lambda$: the duration of the movement;
- $\gamma$: the time when the steering angle changes sign;
- $S_{\max}$: the maximum speed on the profile;
- $\Phi_{\max}$: the maximum absolute steering angle on the profile;
- $\beta$: time to reach $S_{\max}$ from 0;
- $\alpha$: time to reach $\Phi_{\max}$ from 0.

The formal expression of our control profile is:

$$S(t) = S_{\max} \times \begin{cases} t/\beta & 0 \leq t < \beta \\ +1 & \beta \leq t < \lambda - \beta \\ (\lambda - t)/\beta & \lambda - \beta \leq t < \lambda \end{cases} \quad (4)$$

$$\Phi(t) = \Phi_{\max} \times \begin{cases} t/\alpha & 0 \leq t < \alpha \\ +1 & \alpha \leq t < \gamma - \alpha \\ (\gamma - t)/\alpha & \gamma - \alpha \leq t < \gamma + \alpha \\ -1 & \gamma + \alpha \leq t < \lambda - \alpha \\ (t - \lambda)/\alpha & \lambda - \alpha \leq t < \lambda \end{cases} \quad (5)$$

It has been proved in [Paromtchik and Laugier, 1996b; 1996a; 1998] that using $\gamma = \lambda/2$ generates a "parallel parking" movement, that is, orientation is the same at the beginning and the end of the motion. Using an arbitrary $\gamma$, [Paromtchik, 2003; 2004] demonstrated that it is possible to reach an arbitrary orientation. Figure 4 illustrates typical movement resulting of the afore mentioned control profile.

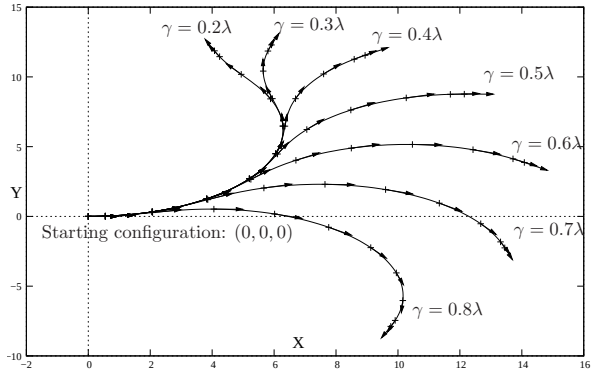


Figure 4: Resulting elementary movement: $\lambda = 10s$, $V_{\max} = 2m/s$, $\Phi_{\max} = 0.45rad$, varying $\gamma$.

Using the vehicle dynamic model, and the control profiles, we can associate to each set of 6 parameters $(\lambda, \gamma, S_{\max}, \Phi_{\max}, \alpha, \beta)$, a resulting displacement $(\Delta x, \Delta y, \Delta\theta)$. This displacement is the result of the application of the control profile for the duration $\lambda$. Let's call $\mathcal{M}$ the motion model resulting from this injective mapping between parameters and displacement:

$$(\Delta x, \Delta y, \Delta\theta) = \mathcal{M}(\lambda, \gamma, S_{\max}, \Phi_{\max}, \alpha, \beta) \quad (6)$$

This motion model can also be expressed using the dynamic model:

$$(\Delta x, \Delta y, \Delta\theta)^T = \int_{t=0}^{\lambda} (\dot{x}, \dot{y}, \dot{\theta})^T dt \quad (7)$$

Practically, the motion model is computed by integrating the dynamic model, or, in other words, by applying the control profile on a simulated car.

### 3.5 Planning a Reverse Parking manoeuvre

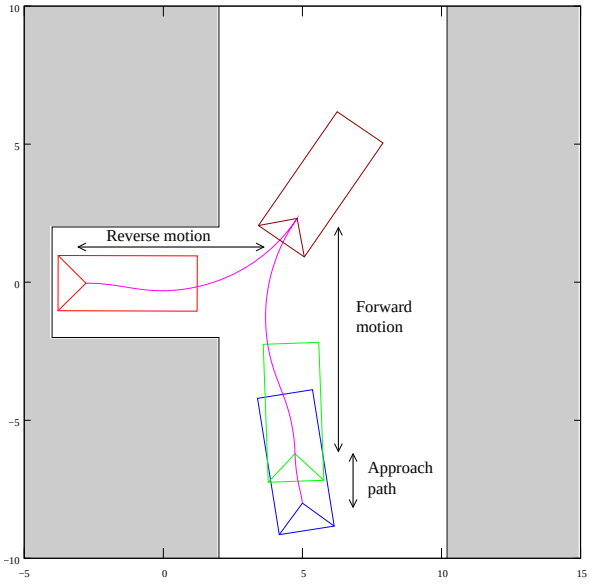


Figure 5: Basic reverse parking manoeuvre steps

Using the notion of elementary movement, it becomes quite easy to define a reverse parking manoeuvre: it is the connection of 3 elementary movements (see fig. 5):

- The approach: starts from the initial configuration and brings the front of the car at the beginning of the lot, with a good orientation.
- The forward part brings the back of the car in a good position for entering the lot.
- The reverse part enters the lot and finishes parking the car.

So, the planning of the parking manoeuvre seems to be an easy procedure: just find the three sets of parameters that generates three good elementary movements. The difficulty arises from the fact that this basically means inverting the model $\mathcal{M}$, and this is where the articles [Paromtchik, 2003; 2004] stop.

Consequently we had to develop our own, practical and reliable method to find the elementary movements. This is what we will present in the next sections.

## 4 Path Planning with SQL

### 4.1 Using a database

Let's consider the problem we are facing when trying to plan paths from elementary movement. If we want to plan a path to a goal configuration, then this means we know $(\Delta x, \Delta y, \Delta\theta)$, and we want to find a set of parameters $(\lambda, \gamma, S_{\max}, \Phi_{\max}, \alpha, \beta)$, such that:

$$\mathcal{M}(\lambda, \gamma, S_{\max}, \Phi_{\max}, \alpha, \beta) = (\Delta x, \Delta y, \Delta\theta)$$

Unfortunately, for a car-like vehicle, $\mathcal{M}$ is not easily invertible, and the computation of a specific value of $\mathcal{M}$ is quite expensive since we have to integrate numerically the dynamic model over $[0, \lambda]$.

A usual way to solve this complex problem on a computer is to precompute and store in memory a discretised version of the motion model, and then to use efficient search techniques to look for a given value of $(\Delta x, \Delta y, \Delta\theta)$. We believe the good answer to this problem is to use a database Engine. Database engines are designed to store and retrieve efficiently information, so reimplementing our own mechanism would be a waste of effort. Furthermore, we will see that using the database engine, it is easy and efficient to find answers to requests that are more complex than just how to move to a goal configuration.

### 4.2 Schema of an elementary movement DB

The information we want to store in the database is the motion model, that is the relation between control profile parameters $(\lambda, \gamma, S_{\max}, \Phi_{\max}, \alpha, \beta)$, and resulting movement $(\Delta x, \Delta y, \Delta\theta)$. Consequently, the resulting database schema is a single table in which each tuple is the concatenation of the parameters and the resulting movement, stored as floating point numbers.

In order to reduce the size of the database, we will consider $\alpha$ and $\beta$ fixed for a given vehicle. So, one entry of the database will be a 7-tuple: $(\lambda, \gamma, S_{\max}, \Phi_{\max}, \Delta x, \Delta y, \Delta\theta)$. Also, to increase the performance of the elementary movement search, we create an index on $(\Delta x, \Delta y, \Delta\theta)$.

Now we just have to fill the database: we sample the parameter space $(\lambda, \gamma, S_{\max}, \Phi_{\max})$, for each value of the parameters we compute the resulting movement, and store the resulting 7-tuple in the database. The resolution of the sampling will determine the size of the database. Then this size will influence first the speed of the path planning algorithm, and second the complexity of the environments in which planning will be successful. For instance, on a very wide road, with a very wide parking lot, any sensible manoeuvre will easily enter the lot. On the contrary, only a few very well chosen control profiles will allow to enter a narrow lot on a narrow road.

### 4.3 Moving to a spot

Let's assume the vehicle initial configuration is $C_i = (x_i, y_i, \theta_i)$ and that the goal to reach is $C_g = (x_g, y_g, \theta_g)$. The displacement $\delta C$ between $C_i$ and $C_g$ is $(\delta x, \delta y, \delta\theta)$ where

$$\delta\theta = \theta_g - \theta_i$$

and

$$\begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \cos\delta\theta & \sin\delta\theta \\ -\sin\delta\theta & \cos\delta\theta \end{pmatrix} \begin{pmatrix} x_g - x_i \\ y_g - y_i \end{pmatrix}$$

Theoretically, we could request if there is a parameter tuple that results in the displacement $\delta C$. Nevertheless, due to the discretised nature of the database, we have to allow for some approximation and to look for parameters such that resulting movement $(\Delta x, \Delta y, \Delta\theta)$ verifies:

$$|\Delta x - \delta x| \leq \xi_x,\ |\Delta y - \delta y| \leq \xi_y,\ |\Delta\theta - \delta\theta| \leq \xi_\theta \quad (8)$$

where $(\xi_x, \xi_y, \xi_\theta)$ is the tolerance vector.

SQL is a language to express request to database engines. The SQL request selecting the tuples that satisfies the constraint above is:

**select** $\star$ **from** `ElementaryMovement`
**where**
$abs(\Delta x - \delta x) \leq \xi_x$ `and`
$abs(\Delta y - \delta y) \leq \xi_y$ `and`
$abs(\Delta\theta - \delta\theta) \leq \xi_\theta$

### 4.4 Moving to a line

In some situations, we are not really interested in moving to a specific configuration, but rather to move to a specific line. For instance, for the reverse part of the parking manoeuvre, finding a elementary movement that reaches exactly the parking position can be tricky. On the other hand, once the vehicle is on a line passing by the parking position, with a correct orientation, the end of the manoeuvre is an easy straight line motion. So from the parking algorithm point of view, it is sufficient to find a elementary movement that reaches this line.

Formally now, let's assume we start from a configuration $C_i = (x_i, y_i, \theta_i)$, and we want to reach orientation $\theta_g$ on a line defined by a point $A = (x_a, y_a)$ and a unit vector $\vec{u} = (x_u, y_u)$. The final configuration of the vehicle is $C_g = (x_g, y_g, \theta_g)$ and it satisfies:

$$\begin{pmatrix} x_g - x_a \\ y_g - ya \end{pmatrix} \times \vec{u} = 0$$

Practically, we also have to introduce some tolerance with respect to the line. We define $\xi_l$ as the maximum allowed distance to the line. $C_g$ must satisfy:

$$|(x_g - x_a)y_u - (y_g - y_a)x_u| \leq \xi_l \quad (9)$$

We can now express the resulting constraint for the elementary movement. We use the definition of $\delta C = (\delta_x, \delta y, \delta\theta)$ introduced previously. We have one constraint for $\delta\theta$:

$$|\Delta\theta - \delta\theta| \leq \xi_\theta \quad (10)$$

and equation 9 can be rewritten as:

$$|(x_i - x_a + \Delta x \cos\delta\theta - \Delta y \sin\delta\theta)y_u - (y_i - y_a + \Delta x \sin\delta\theta + \Delta y \cos\delta\theta)x_u| \leq \xi_l \quad (11)$$

Let's define

$$a = y_u \cos\delta\theta - x_u \sin\delta\theta \quad (12)$$
$$b = -x_u \cos\delta\theta - y_u \sin\delta\theta \quad (13)$$
$$c = (x_i - x_a)y_u - (y_i - y_a)x_u \quad (14)$$

The SQL request selecting the tuples that results in the required movement is:

**select** $\star$ **from** `ElementaryMovement`
**where**
$abs(\Delta\theta - \delta\theta) \leq \xi_\theta$ `and`
$abs(a\Delta x + b\Delta y + c) \leq \xi_l$

In general, searching the parameter space efficiently for tuples satisfying the above constraints would be a hard problem, especially when there is more than 50 millions records in the database. A good database engine uses several tricks to solve this problem: first it uses its index on $(\Delta x, \Delta y, \Delta\theta)$ to easily access a specific tuple. This index is usually made of some kind of hash table or research tree such that access to a specific element can be expected to be at most logarithmic. Second, the engine parses the numeric constraints so as to make sub-request in the database and only look at the potentially useful part of the data-set. Implementing all these optimisations on a home-made data management would require a tremendous amount of development, so using a database engine is really a time-saving choice for this problem.

As an illustration, a request as the one above will take some tenth of a second on a powerful database server.

### 4.5 Planning a complete manoeuvre

From now on, we will consider that the vehicle initiated the parking manoeuvre at a configuration $(x_0, y_0, \theta_0)$, and that the parking lot is on the left side of the road. The procedure we will present now is an heuristic to decompose the complete parking manoeuvre into elementary movements. The procedure for a lot on the right side can easily be deduced from this one.

#### Approach

The first part of the manoeuvre will be to plan an approach path. The goal is to align the vehicle with the road – with the two lines that delimit the road –, to bring its front close to the beginning of the lot, and to bring it at a reasonable distance from the border of the road. Figure 6 illustrates this objective.

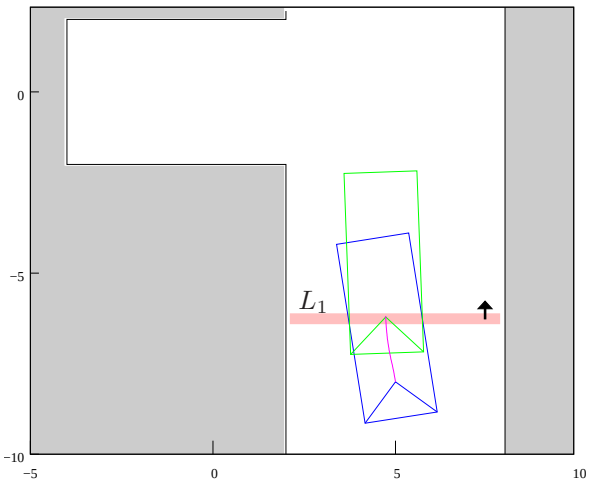


Figure 6: Approach manoeuvre

In the case of our experimental vehicles, we chose to bring the front of the vehicle at $0.5m$ from the lot, at least $1m$ from the border of the road.

In terms of path planning query, this objective corresponds to a path to the line $L_1$ on figure 6, with a orientation of $\pi/2$. This line is perpendicular to the road, and located such that the front of the vehicle is at $0.5m$ from the lot.

The resulting configuration is named $C_1 = (x_1, y_1, \theta_1)$.

#### Forward manoeuvre

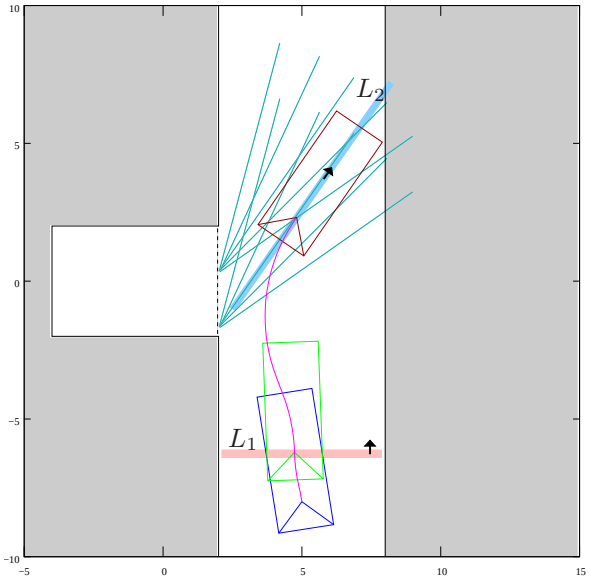


Figure 7: Forward manoeuvre. Lines in blue are example of destination lines for this part of the manoeuvre.

From configuration $C_1$, we chose to look for path to oblique lines since they seem to be the ones we try to reach when parking a car manually. Conversely to the approach path, there is not one preferred destination line, so we have to try to complete the parking manoeuvre for each tested line. Some of the tried destination lines are depicted on figure 7. We basically try lines passing through the entrance of the parking lot, with an orientation close to $\pi/4$.

**Reverse manoeuvre**

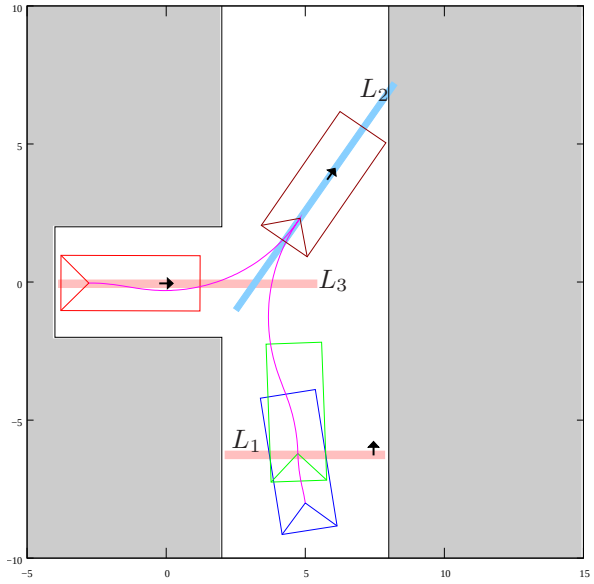


Figure 8: Complete parking manoeuvre, including the reversing part to line $L_3$.

The reverse manoeuvre is the final one of the parking manoeuvre. Its goal is to reach the line $L_3$ aligned with the lot, passing by the parked configuration. The goal orientation is the final parked orientation: $0\,rad$. For each configuration reachable with the forward manoeuvre we test if a reverse manoeuvre is possible. Such a final manoeuvre is shown on figure 8.

**Obstacles**

For all the elementary path forming the complete parking manoeuvre, if the database if dense enough, there is plenty of possible elementary movement. The additional constraint comes from the environment model: an elementary movement is only acceptable is it does not lead to any collision.

To check possible collisions, we use the vehicle model and the environment model: we simulate the application of the control profile to the vehicle model with small discrete time steps, and for each intermediary position, we check that the vehicle stays in the free space. Even with our simple geometry, this test can be quite expensive if applied on too many possible elementary movements.

In order to reduce its cost, several approaches are possible: decrease the density of the database, increase the size of the time steps or reduce the tolerance parameters. Since the trajectory resulting from our control profiles are quite smooth, increasing reasonably the size of the time steps is a safe and efficient option.

**Summary**

To summarise the complete parking manoeuvre, it is a three step approach where elementary movements to convenient lines are looked for in the database. Each of these movements is tested against collision with the environment. The resulting manoeuvre is dynamically feasible, collision free and predictable from an external observer.

## 5 Graphical User Interface

In an ideal world, our system would perceive the environment, find itself a convenient parking lot, plan and realise its manoeuvre. Nevertheless, on our real system we chose to keep the user in the loop. This is makes it a parking *assistance* system.

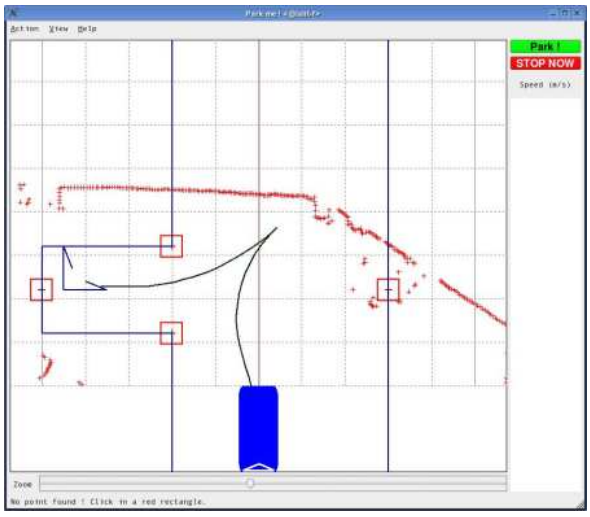


Figure 9: Graphical User Interface

Our next objective is that the system find itself a convenient parking lot, displays it to the user who can confirm it, reject it or modify it. In our current application, the user is shown the output of a sweeping laser range finder with respect to its vehicle, and uses this information to mark the limits of the road and of the parking lot. The resulting interface is shown in figure 9.

Once the environment has been defined, the user can start the planning process. The resulting path is then displayed and, upon user confirmation, executed. A

planning path computed by the system is visible on figure 9.

# 6 Experimentation

## 6.1 Driving assistance vs. full autonomy

Our experimentations where to be conducted on two experimental platforms (see fig. 11): a 4WD vehicle[1] and a small mowing tractor[2]. Both vehicles follow the dynamic of a car, so we can fortunately use the method presented in this paper. The tractor is fully autonomous: its speed and steering angle can be software controlled. Conversely, on the 4WD the steering angle is controllable, but the speed can only be controlled through the cruise control at speeds over $40km/h$, which is obviously excessive for a parking manoeuvre. Consequently, on the 4WD we will have the driver controlling the vehicle speed while the vehicle steers autonomously.

The 4WD situation has several practical advantages: first by keeping the driver deeply in the loop, its is really a driving assistance system and second the driver keeps the the liability in case of an incident. This is interesting since liability issues seem to be one of the major impediments for the introduction or robotics in the every day cars.

From a technical point of view now, we will have to modify our control profiles to account for the specificity of the 4WD. The required modification is straightforward: since we cannot control the speed, we will convert the steering profile into a function of the travelled distance, and ignore the speed profile.

## 6.2 Pushing the limits

In order to test the limit of the planning method, we tried it with different road sizes, and different parking lot sizes. We obtained the results shown in table 1. For each road width – in percent of the vehicle length –, we tried different lot sizes – in percent of the vehicle width – and we marked with an "X" the combinations where a parking manoeuvre was found. In each test, the vehicle started with a $\pi/2$ orientation, in the middle of the road, two vehicle lengths from the lot. Figure 10 shows two successful plan for a very narrow road and a very narrow lot.

## 6.3 On a real vehicle

As mentioned before, our system was designed to work both on the ANU 4WD vehicle and on the CSIRO Autonomous Tractor (AT) (see fig. 11). Unfortunately, due to some sensor failures and administrative problems we have not been able to perform experiment on the 4WD yet, but we expect some results for the final version of this paper. In the meantime, we performed autonomous parking manoeuvres on the AT.

[1]Owned by Australian National University

[2]Owned and retrofitted by the CSIRO ICT Centre [Usher *et al.*, 2003]

| | Lot width (%) | | | |
|---|---|---|---|---|
| Road width (%) | 150 | 200 | 300 | 400 |
| 80 | | | | |
| 90 | | | X(1) | X |
| 100 | | | X | X |
| 120 | | X | X | X |
| 150 | X(2) | X | X | X |

Table 1: Limits of the parking manoeuvre planning. Successful planning indicated with an "X". Road width in percent of the vehicle length, lot width in percent of the vehicle width.

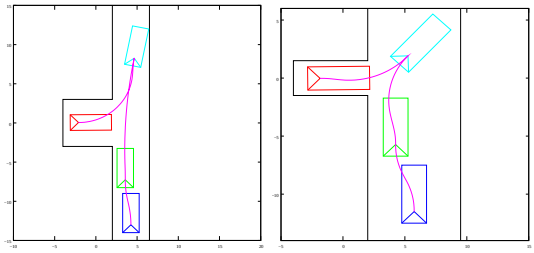

Figure 10: Successful planning. Left: cell marked (1) in table 1, Right: cell marked (2) in table 1

The experimental setting on the AT was as follows: the vehicle is equipped with a tilted sweeping laser range finder on the front. Data from this sensor are displayed in the graphical user interface on a laptop setup above the steering wheel. A virtual parking lot is delimited with traffic cones. These cone are clearly visible in the laser data, so the user can use them to mark the parking lot, and start the parking manoeuvre. Once a feasible, safe path has been found, the manoeuvre starts. Figure 12 shows some images from a parking manoeuvre.

### Trajectory tracking

In this application, trajectory following is performed as in [Paromtchik, 2004]. The idea is the following: if the low level control loop – the control of the vehicle speed and steering angle – are accurate enough, then the control profiles defining the trajectory can be applied di-

Figure 11: Experimental platforms

rectly, in open-loop. This is a satisfying solution as long as the considered trajectories are short but it requires a finely tuned low-level control.

In order to account for minor tracking errors on the AT, the vehicle displacement is tracked from odometry, and when the trajectory reaches its cusp point, a new reverse manoeuvre is computed.

Figure 12: Sequence from a real parking manoeuvre: in this case the user only stays on-board for safety since no obstacle avoidance is active.

## 7 Conclusion

In this paper we presented a practical technique to implement the reverse parking planner introduced in [Paromtchik, 2004]. Our implementation relies on a database engine to compute elementary motions to specific configuration or line of configurations. Using a database also introduces flexibility in terms of required resources: according to the available resources a smaller database may be generated, with a smooth degradation of the planning performances.

Our method has been successfully tested on an autonomous vehicle (the CSIRO Autonomous Tractor) and should be applied to the ANU 4WD in the near future.

## 8 Acknowledgement

We would like to express our gratitude to K. Usher for his assistance when experimenting on the CSIRO Autonomous Tractor, and to L. Fletcher (ANU) and L. Petersson(Nicta), for their help in setting up this algorithm on the ANU 4WD vehicle.

## References

[Ahuactzin *et al.*, 1991] Juan Manuel Ahuactzin, El-Ghazali Talbi, Pierre Bessiere, and Emmanuel Mazer. Using genetic algorithms for robot motion planning. In *Geometric Reasoning for Perception and Action*, pages 84–93, 1991.

[Fraichard and Scheuer, 2004] Th. Fraichard and A. Scheuer. From reeds and shepp's to continuous-curvature paths. *IEEE Trans. on Robotics*, 20(6):1025–1035, December 2004.

[Franke, 2005] Uwe. Daimler Chrysler Research Franke. Vision a key to accident free driving. In *The 5th International Conference on Field and Service Robotics*, 2005.

[Hermosillo, 2003] J. Hermosillo. *Planification et exécution de mouvements pour un robot bi-guidable: une approche basée sur la platitude différentielle.* Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), June 2003.

[Khatib, 1986] O. Khatib. Real-time obstacle avoidance for robot manipulator and mobile robots. *The International Journal of Robotics Research*, 1986.

[La Valle and Kuffner, 1999] Steven M. La Valle and James Kuffner. Randomized kinodynamic planning. In *Proc. 1999 IEEE Intl. Conf. on Robotics and Automation*, 1999.

[La Valle, 2006] Steven M. La Valle. *Planning Algorithms.* 2006. http://msl.cs.uiuc.edu/planning.

[Latombe, 1992] J.-C. Latombe. *Robot Motion Planning.* 1992.

[Overmars, 1992] M. Overmars. A random approach to motion planning, 1992.

[Paromtchik and Laugier, 1996a] Igor E. Paromtchik and C. Laugier. Autonomous parallel parking of a nonholonomic vehicle. In *Proc. of the IEEE Intelligent Vehicles Symp.*, 1996.

[Paromtchik and Laugier, 1996b] Igor E. Paromtchik and C. Laugier. Motion generation and control for parking an autonomous vehicle. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1996.

[Paromtchik and Laugier, 1998] Igor E. Paromtchik and C. Laugier. Automatic parallel parking and returning to traffic maneuvers. In *Video Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1998.

[Paromtchik, 2003] Igor E. Paromtchik. Planning control commands to assist in car maneuvers. In *Proc. of the 11th IEEE Int. Conf. on Advanced Robotics*, 2003.

[Paromtchik, 2004] Igor E. Paromtchik. 2004.

[Pivtoraiko and Kelly, 2005] M. Pivtoraiko and A. Kelly. Constrained motion planning in discrete state spaces. In *The 5th International Conference on Field and Service Robotics*, 2005.

[Pradalier *et al.*, 2005] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier. The cycab: a car-like robot navigating autonomously and safely among pedestrians. *Robotics and Autonomous Systems*, 50(1):51–68, 2005.

[Sekhavat *et al.*, 1996] S. Sekhavat, P. SVESTKA, J.P. LAUMOND, and M.H. OVERMARS. Probabilistic path planning for tractor trailer. Technical report, LAAS, Robotics and Artificial Intelligence, 1996.

[Sekhavat *et al.*, 2001] S. Sekhavat, J. Hermosillo, and P. Rouchon. Motion planning for a bi-steerable car. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2001.

[SM, 2005] 2005. http://www.seeingmachines.com.

[Usher *et al.*, 2003] K. Usher, M. Dunbabin, P. Corke, and P. Ridley. Sensing for visual homing. In *Australian Conference on Robotics and Automation*, 2003.