Kinodynamic Motion Planning: Connecting Exploration Trees Using Trajectory Optimization Methods

Florent Lamiraux
LAAS-CNRS
7 avenue du Colonel Roche
31077 Toulouse cedex 4
France
Email: florent@laas.fr

Etienne Ferré and Erwan Vallée
Kineo CAM
Prologue - La Pyrénéenne
31312 Labege cedex
France
Email: eferre@kineocam.com

***Abstract*— Motion planning for complex dynamic systems as well as kinodynamic motion planning are still problems difficult to solve in their generic formulation. For systems for which no steering method is known, the only existing algorithms consist in building an exploration tree in the configuration space by exploring the input space of the system. The main drawback of this type of methods is that they never reach exactly the goal but stops the search when a small neighborhood of the goal has been reached. If the neighborhood is small, the exploration method needs to produce a lot of nodes. In this paper, we propose a solution to cope with this complexity issue. We run a tree exploration method with a big neighborhood and then we locally modify the trajectory in order to make it reach exactly the goal. Our solution is based on a trajectory optimization method we have developed earlier in a mobile robot context and that we have adapted for the problem raised here. The method is generic and can be applied to any dynamic system. A few experimental results are given at the end of the paper.**

I. Introduction

Planning collision-free trajectories for systems subject to kinematic or dynamic constraints – that is any real system – is today still an open problem. Solutions based on local methods have been proposed for the nonholonomic systems for which a local method could be designed [6], [9], [10]. Most of these methods produce a path and the additional kinematic constraints imposed by the maximal velocities and accelerations of the systems are considered in a second step. These methods are usually efficient, however, the current state of the art provides local methods for only very few systems, more or less systems belonging to the class of differentially flat systems [9]. Other work globally addresses kinematic constraints and obstacle avoidance, but only for specific systems like car-like vehicles [8]. From a control theory point of view, a system subject to kinematic constraints is ususally considered as a system with less input than configuration variables. Including velocities in the state of the system enables us to treat bounds on the velocities as collisions. In this case, the problem is called "kinodynamic motion planning" [3]. For these systems, exploring the input space has the double advantage that it is less time-consuming and that it automatically produces trajectories satisfying the constraints. This idea was first developed in [1] and then extended to randomized techniques [7]. In both papers, an exploration tree is developed in the configuration space, either randomly or deterministically. Nodes are configurations and edges are feasible trajectories obtained by applying different control vectors for short periods of time. Two trees can also be developed: one from the initial configuration and one from the goal configuration. Exploration ends when two nodes of different trees are closer to each other than a tolerance distance. The main advantage of this approach is that it can be applied to any dynamic system as soon as the relation between the input vector and the state vector of the system is known. It also has an important drawback: the algorithm needs to explore most of the free configuration space before reaching the goal or connecting the trees. Indeed the lack of relevant distance function in the configuration space makes the use of heuristics to guide the exploration very unefficient. Moreover, if the tolerance distance is small, the exploration step will create a huge tree, even for small environments. If the tolerance distance is big, either the trajectory ends far from the goal or there is an important discontinuity in the solution trajectory. [2] propose a solution to this issue. In this paper, we propose another method to improve the performance of exploration tree methods applied to systems subject to kinematic constraints or to the kinodynamic problem. This method is based on earlier work on trajectory deformation for nonholonomic systems we did in the past years [4] and consists in two steps. In the first step, we develop a random tree from the initial and goal configurations, with a big tolerance distance. We thus get two feasible trajectories: one starting from the initial configuration and the other one ending at the goal configuration. The distance between the end of the first trajectory and the beginning of the second one is smaller than the tolerance distance. In the second step, we iteratively deform both trajectories in such a way that both free ends get closer to each other and that the trajectory keeps collision-free and remains feasible. To perform this second step, we use the

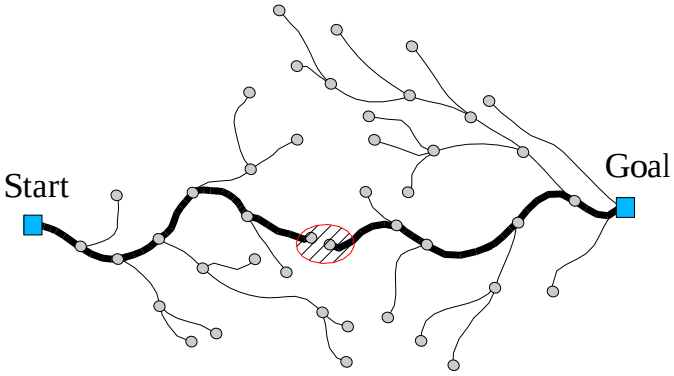


Fig. 1. Bi-RRT Algorithm: one tree is developed from the initial configuration and one from the final configuration. The expansion ends whenever two configurations of both trees are closer to each other than a tolerance distance d_{tol}.

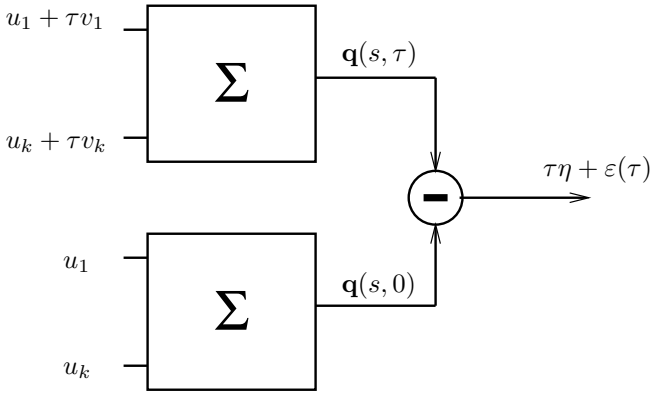


Fig. 2. Perturbing the input functions of a dynamic system results in a deformation of the initial trajectory.

trajectory deformation method introduced in [4], extended to systems with drift, and described in Section III. A simpler version of this method has already been used to plan paths of oversized convoys through villages [5]. In Section II, we quickly describe the first step of our algorithm consisting in a classical bi-RRT method. In Section IV, we present an example of our method applied to the parallel-parking problem for a car controlled by acceleration.

II. Exploring Trees for Kinodynamic Motion Planning

The first step of our motion planning algorithm consists in exploring the configuration space (state space for kinodynamic motion planning) of the system using a classical bi-RRT procedure as described in [7]. The idea consists in expanding two trees: one from the initial configuration and one from the final configuration. Algorithm 1 explains how each tree is expanded. At each iteration, a random configuration is chosen. From the closest node to this configuration in the tree, N_{rand} paths are produced by applying random control functions to the system. For each path, the number of nodes of the tree at a distance smaller than a threshold $d_{neighbor}$ to the end of the path is counted. The path ending at the configuration with the least neighbors is added to the tree. Thus, the tree tends to expand in regions with few nodes. This iteration is applied for both trees (initial and goal) and stops as soon as one configuration of the initial tree is closer than a tolerance distance d_{tol} to one node of the goal tree. Figure 1 illustrates the data-structure built by the bi-RRT algorithm.

The value of d_{tol} has an important influence over the final trees. The bigger d_{tol} is, the denser the trees are in the free configuration space at the end of the expansion step and the longer the tree expansion lasts.

Algorithm 1 Tree expansion

```
Tree T     //tree to be developed
while NOT ADDED do
  q ← shootRandomState
  N_near ← getNearestNode(q)
  for i = 1 to N_rand do
    u_i ← shootRandomCommand
    path_i ← computePath(f, N_near,u_i)
    if path_i VALID then
      n_nodes ←countNodesInNeighborhood(path_i)
      if n_nodes ≤ bestN_nodes then
        bestPath ← path_i
        bestN_nodes ← n_nodes
      end if
    end if
  end for
  add final configuration of bestPath to T.
  ADDED = TRUE
end while
```

III. Trajectory Deformation Method

After the first step of our method, described in the earlier section, we get two trajectories in the configuration space of the system. The first trajectory starts from the initial configuration $\mathbf{q}_{init}$ and ends at a configuration that we denote by $\mathbf{q}_{1\ end}$. The second trajectory starts from a configuration that we denote $\mathbf{q}_{2\ start}$ and ends at the final configuration $\mathbf{q}_{goal}$. The distance between $\mathbf{q}_{1\ end}$ and $\mathbf{q}_{2\ start}$ is smaller than the tolerance distance d_{tol} (Figure 1).

In the second step of the method, we deform these trajectories until the distance between $\mathbf{q}_{1\ end}$ and $\mathbf{q}_{2\ start}$ decreases to 0. To perform this step, we use the method presented in [4] and adapted to the problem of motion planning. We briefly explain the method now.

A. Trajectory of a dynamic system

A trajectory $\mathbf{q}(s)$ for a dynamic system is uniquely defined by an initial configuration $\mathbf{q}_0$ and input functions defined over an interval of time $[0, T]$:

$$\forall s \in [0, S]\ \mathbf{q}'(s) = X_0(\mathbf{q}(s)) + \sum_{i=1}^{k} u_i(s) X_i(\mathbf{q}(s))$$

where X_0 is the drift vector field and the other X_i's are the control vector fields of the system.

B. Input perturbation and trajectory deformation

To deform the initial trajectory of the system, we thus need to pertub the input functions of the trajectory. In this aim, let us define k *input perturbation* functions $v_1, ..., v_k$ over the same interval $[0, S]$ and let us define a set of trajectories $\mathbf{q}(s, \tau)$ parameterized by a real number τ by perturbing the initial trajectory, adding the input perturbations scaled by real number τ to the initial input functions. For each τ, trajectory $\mathbf{q}(s, \tau)$ is defined by:

$$\begin{aligned} \mathbf{q}(0, \tau) &= \mathbf{q}_0 \\ \frac{\partial \mathbf{q}}{\partial s}(s, \tau) &= X_0(\mathbf{q}(s, \tau)) + \sum_{i=1}^{k} (u_i(s) + \tau\ v_i(s)) X_i(\mathbf{q}(s, \tau)) \end{aligned}$$

Each value of τ thus defines a feasible trajectory starting at $\mathbf{q}_0$.

On the other side, for each value of s, $\tau \to \mathbf{q}(s, \tau)$ draws a curve in the configuration space of the system. We denote by $\eta(s, \tau)$ the derivative of this curve for $\tau = 0$ and we call this derivative the *direction of deformation* induced by input perturbation $\mathbf{v}(s) = (v_1(s), \cdots, v_k(s))$:

$$\eta(s) = \frac{\partial \mathbf{q}}{\partial \tau}(s, 0)$$

For each value of s, vector $\eta(s)$ gives the direction in which configuration $\mathbf{q}(s)$ of the initial trajectory moves when input perturbation $\tau \mathbf{v}$ is applied for infinitesimal values of τ. Figure 2 illustrates the notation introduced in this section.

The relation between η and τ is given by the linearized system of the dynamic system about the initial trajectory:

$$\eta'(s) = A(s)\eta(s) + B(s)\mathbf{v}(s) \tag{1}$$

where

$$A(s) = \frac{\partial X_0}{\partial \mathbf{q}}(\mathbf{q}(s, 0)) + \sum_{i=1}^{k} u_i(s) \frac{\partial X_i}{\partial \mathbf{q}}(\mathbf{q}(s, 0))$$

and

$$B(s) = \begin{pmatrix} X_1(\mathbf{q}(s, 0)) & \cdots & X_k(\mathbf{q}(s, 0) \end{pmatrix}$$

is the $n \times k$-matrix the columns of which are the control vector fields evaluated along the trajectory.

C. Potential field

The trajectory deformation algorithm is applied successively to each of both trajectories computed in Section II. We explain here an iteration of the algorithm, applied to the first part of the trajectory going from $\mathbf{q}_{init}$ to $\mathbf{q}_{1\ end}$. The goal here is to make $\mathbf{q}_{1\ end}$ move toward $\mathbf{q}_{2\ start}$. To reach this goal we define a potential field over the set of feasible trajectories defined by the distance of the end configuration to $\mathbf{q}_{2\ start}$ and by the distance of the trajectory to obstacles:

$$V(\tau) = \frac{1}{2} \|\mathbf{q}(S) - \mathbf{q}_{2\ start}\|^2 + \int_0^S u(\mathbf{q}(s)) ds \tag{2}$$

where u is a potential field defined over the configuration space of the system. u increases close to obstacles and decreases when the distance to obstacles increases. For instance, u can be defined by:

$$\begin{aligned} u: \ \mathcal{C} &\to \mathbf{R} \\ \mathbf{q} &\to u(\mathbf{q}) = \tfrac{1}{d_{obst}(\mathbf{q})} \end{aligned}$$

where $d_{obst}(\mathbf{q})$ is the distance between the robot in configuration $\mathbf{q}$ and the obstacles. The variation of the path potential along a given direction of deformation is given by the following expression:

$$\begin{aligned} \frac{dV}{d\tau}(\tau) &= (\mathbf{q}(S) - \mathbf{q}_{2\ start} | \eta(S, \tau)) \\ &\quad + \int_0^S \frac{\partial u}{\partial \mathbf{q}}(\mathbf{q}(s, \tau)) \eta(s, \tau) ds \end{aligned}$$

D. Finite-dimensional subspace of input perturbations

To make the trajectory stay away from obstacles and the end of the trajectory move toward $\mathbf{q}_{2\ start}$, we need to find an input perturbation $\mathbf{v}$ that makes the above potential decrease. For that, we define p k-dimensional vector valued test-functions $\mathbf{e}_1, ..., \mathbf{e}_p$ defined over $[0, S]$, where p is a positive integer. For each of them, we compute the corresponding direction of deformation $\mathbf{E}_i$:

$$\mathbf{E}_i(0) = 0 \tag{3}$$

$$\mathbf{E}_i'(s) = A(s)\mathbf{E}_i(s) + B(s)\mathbf{e}_i(s) \tag{4}$$

We restrict input perturbation $\mathbf{v}$ into the linear subspace of functions spanned by the $\mathbf{e}_i$'s:

$$\mathbf{v}(s) = \sum_{i=1}^{p} \lambda_i \mathbf{e}_i(s)$$

where λ is a vector. From the linearity of System (1), the corresponding direction of deformation is:

$$\eta(s) = \sum_{i=1}^{p} \lambda_i \mathbf{E}_i(s) \tag{5}$$

and the variation of potential induced by this direction of deformation can be written as follows:

$$\frac{dV}{d\tau}(\tau) = \sum_{i=1}^{p} \mu_i \lambda_i$$

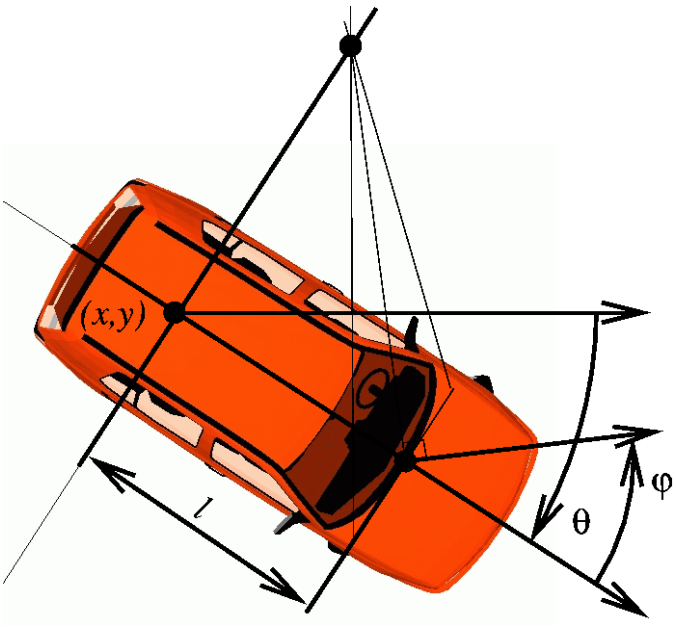


Fig. 3. Car-like mobile robot. (x, y) is the vector of coordinates of the center of the rear wheel axis, θ is the orientation of the car with respect to x-axis, and φ is the steering angle.

where μ_i is the variation of potential induced by $\mathbf{E}_i$:

$$\mu_i = (\mathbf{q}(S) - \mathbf{q}_{2\ start}|\mathbf{E}_i(S)) + \int_0^S \frac{\partial u}{\partial \mathbf{q}}(\mathbf{q}(s,\tau))\mathbf{E}_i(s)ds$$

Thus choosing $\lambda_i = -\mu_i$ in (5) yields a direction of deformation that makes decrase the potential. The trajectory is updated as follows:

$$\mathbf{q}(s) \leftarrow \mathbf{q}(s) + \tau\eta(s)$$

with $\tau > 0$. For τ small enough, the new trajectory has a smaller value of potential after deformation.

a) Deformation algorithm:: the iteration above is successively applied to the trajectory going from $\mathbf{q}_{init}$ to $\mathbf{q}_{1\ end}$ and to the trajectory going from $\mathbf{q}_{2\ start}$ to $\mathbf{q}_{goal}$. For this latter trajectory, the limit condition (3) is replaced by:

$$\mathbf{E}_i(S) = 0$$

in such a way that the end of the trajectory stays at $\mathbf{q}_{goal}$ and in Expression (2), $\mathbf{q}(S) - \mathbf{q}_{2\ start}$ is replaced by $\mathbf{q}(0) - \mathbf{q}_{1\ end}$.

In some cases, the trajectory optimization method can be trapped into a local minimum. For instance if an obstacle lies between $\mathbf{q}_{1\ end}$ and $\mathbf{q}_{2\ start}$. These cases are detected (the trajectory potential does not decrease anymore) and the trajectory optimization method returns failure. The trees are expanded further until another pair of nodes are closer than d_{tol} to each other. Let us notice that in practice, failure happens very scarcely if d_{tol} is not chosen too big.

IV. Experimental Results

In this section, we give a few experimental results that show the benefit of using the trajectory optimization method in addition to RRT exploration algorithm. We run the method on the car-like robot shown on Figure 3. The inputs of this robots are the linear acceleration of the center of the rear wheel axis and the time derivative of the steering angle φ. Thus the configuration space we need to explore is of dimension 5: $\mathbf{q} = (x, y, \theta, \varphi, v)$ and the equation of the dynamic system is:

$$\begin{aligned} \dot{x} &= v\ \cos\theta \\ \dot{y} &= v\ \sin\theta \\ \dot{\theta} &= v\ \frac{\tan\varphi}{l} \\ \dot{v} &= u_1 \\ \dot{\varphi} &= u_2 \end{aligned}$$

where l is the distance between the front and rear wheel axes.

A. Parallel parking

We have applied our method to the parallel parking problem (Figure 4). In a first try, we set d_{tol} to $l/80$. The distance used in the end condition of the RRT expansion is defined as follows:

$$dist(\mathbf{q}_1, \mathbf{q}_2) = \max(|x_2 - x_1|, |y_2 - y_1|, l|\theta_2 - \theta_1|, l|\varphi_2 - \varphi_1|, \frac{l}{v_{max}}|v_1 - v_2|)$$

where v_{max} is the maximal linear velocity of the car. With this tolerance distance, the RRT-expansion step needed to produce 3355 nodes which is a lot for this type of problem. The trajectory deformation step however was very fast and ended after a few iterations.

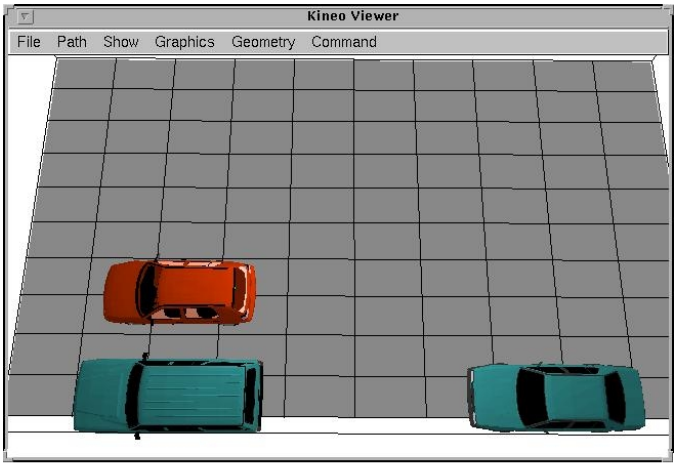
In a second test, we set d_{tol} to a much bigger value: $\frac{l}{8}$. The RRT-expansion step produced only 967 nodes displayed on Figure 5. Figure 6 shows both trajectories after this step. Figure 7 shows the trajectory after optimization. Let us notice that the gap between both trajectories is rather big. The presence of obstacles however did not make the trajectory optimization fail. Again the number of iterations of the trajectory deformation method was small compared to the number of nodes produced by RRT.

B. Exploring a maze

The second trajectory planning problem we have dealt with is a problem of exploration of a maze for the same car-like robot as in the first example. Again, we have run the RRT step with two different values of d_{tol}. For $d_{tol} = l/80$, RRT generated 2738 nodes. For a bigger value of d_{tol}: $d_{tol} = l/8$, RRT generated 1453 nodes. Figure 8 displays the trajectory generated by our method for $d_{tol} = l/8$ and after running the trajectory optimization method. Let us notice that this second step requires few iterations compared to the number of nodes produced by RRT.

V. Conclusion

In this paper, we have proposed a method that significantly improves performance of trajectory planning algorithms based on the expansion of exploration trees in the configuration space by integrating input functions. The method applies a trajectory optimization technique initially developed for controlling the motion of nonholonomic mobile robots. The main advantage



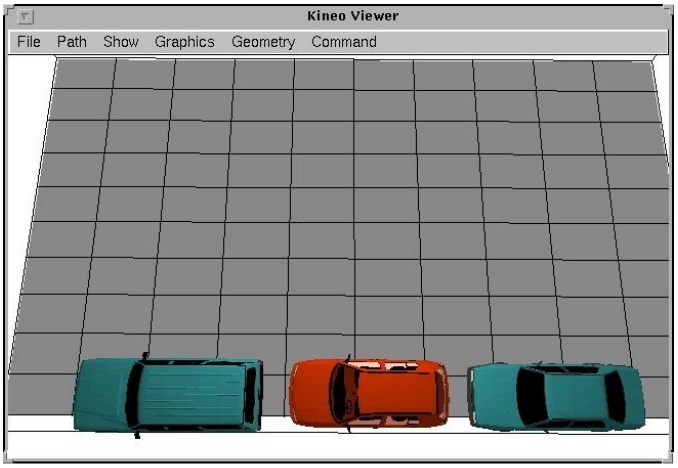


Fig. 4. Parallel parking problem for a car-like robot. The initial configuration (top) and the goal configuration (bottom). The velocity is equal to 0 for both configurations.

of the approach proposed in this paper is that it can be applied to any dynamic system.

ACKNOWLEDGMENT

This work has been partially supported by the European Project MOVIE (IST-2001-39250).

REFERENCES

[1] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[2] P. Cheng, E. Frazzoli, and S. LaValle. Exploiting group symmetries to improve precision in kinodynamic and nonholonomic planning. In *International Conference on Intelligent Robots and Systems*, pages 631–636, Las Vegas, NE, October 2003. IEEE/RSJ.

[3] B.R. Donald, P. Xavier, J.F. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM*, 40:1048–1066, 1993.

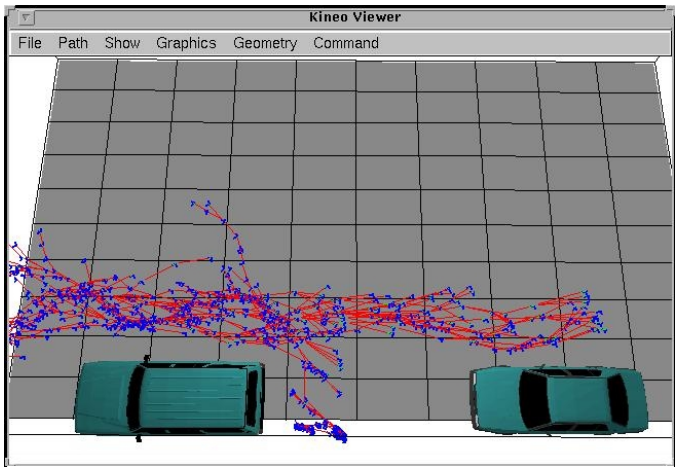


Fig. 5. Graph generated by RRT to solve the parallel parking task.

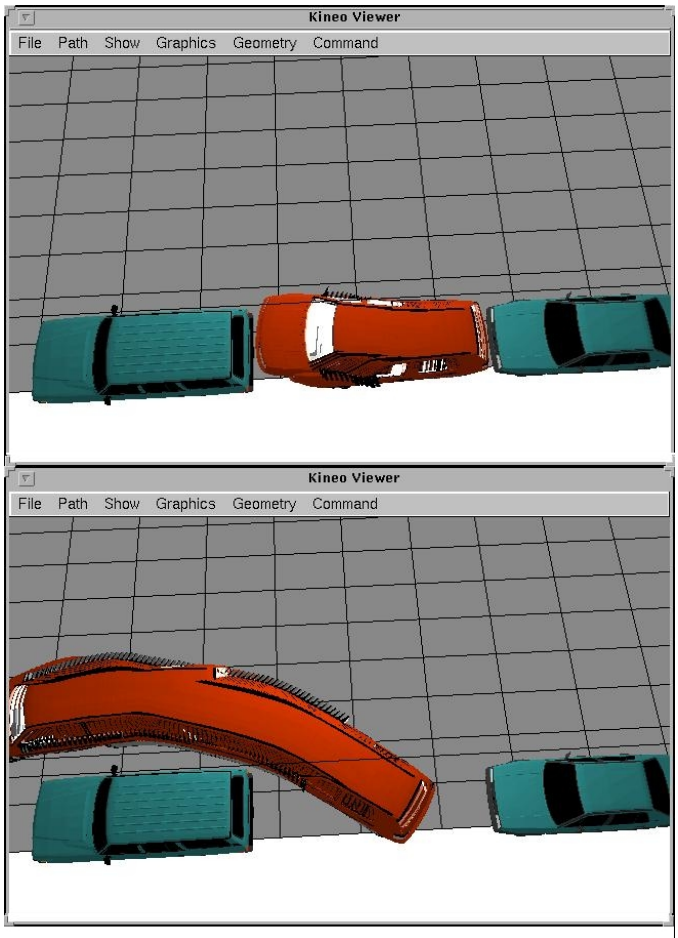


Fig. 6. The two trajectories produced by RRT expansion step. The end of the first trajectory does not fit exactly the beginning of the second trajectory.

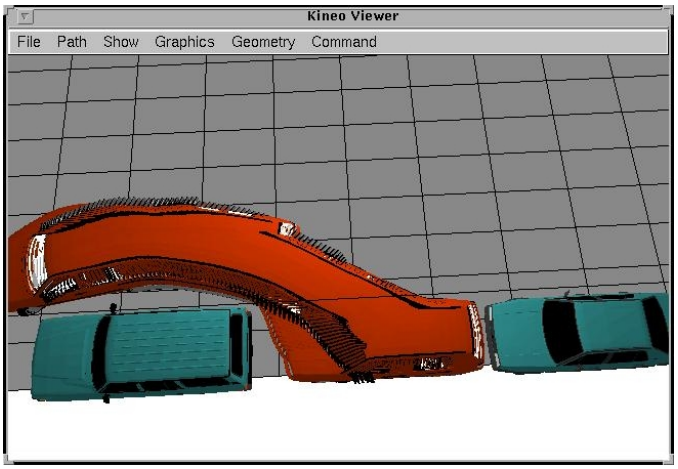


Fig. 7. Graph generated by RRT to solve the parallel parking task.

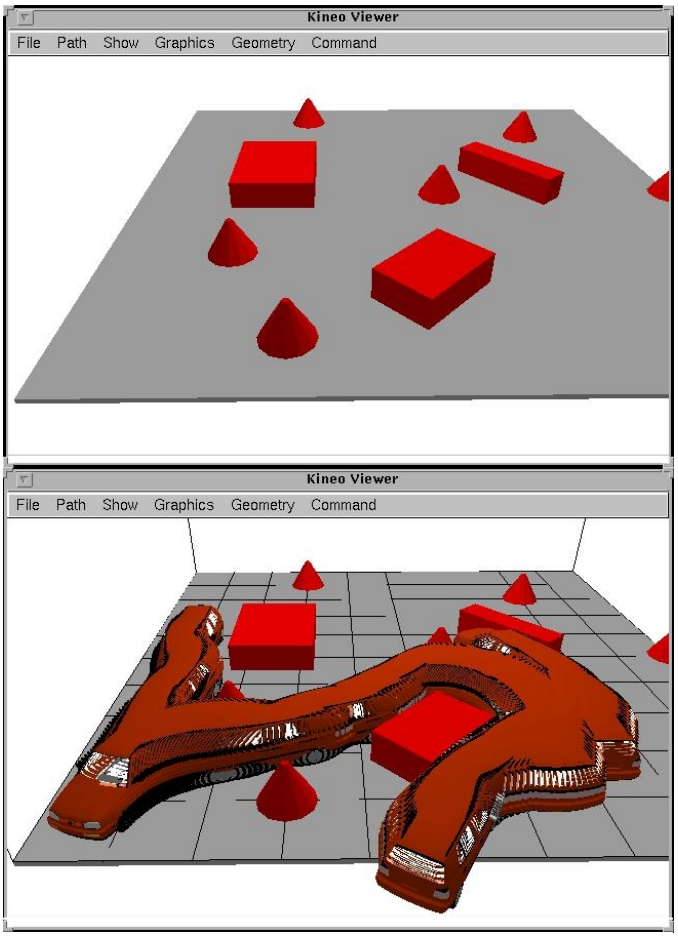


Fig. 8. Exploration of a maze (top). trajectory computed by our method: RRT exploration with $d_{tol} = l/8$ and trajectory optimization (bottom)

[4] F. Lamiraux and D. Bonnafous. Reactive trajectory deformation for nonholonomic systems: Application to mobile robots. In *International Conference on Robotics and Automation*, pages 3099–3104, Washington D.C., May 2002. IEEE.

[5] F. Lamiraux, J.-P. Laumond, C. VanGeem, D. Boutonnet, and G. Raust. Trailer-truck trajectory optimization for airbus a380 component transportation. *IEEE Robotics and Automation Magazine*, (Robotic Technologies applied to Intelligent Transportation Systems), 2003. to appear.

[6] F. Lamiraux, S. Sekhavat, and J.-P. Laumond. Motion planning and control for hilare pulling a trailer. *IEEE Transactions on Robotics and Automation*, 15(4):640–652, August 1999.

[7] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[8] A. Scheuer and T. Fraichard. Continuous curvature path planning for car-like vehicles. In *International Conference on Intelligent Robots and Systems*, pages 997–1003, Grenoble, France, September 1997. IEEE/RSJ.

[9] S. Sekhavat, P. Rouchon, and J. Hermosillo. Computing the flat outputs of engel differential systems - the case study of the bi-steerable car. Arlington VA, June 2001.

[10] S. Sekhavat, P. Švestka, J.-P. Laumond, and M. Overmars. Multilevel path planning for nonholonomic robots using semi-holonomic subsystems. *International Journal on Robotics Research*, 17:840–857, 1998.