Nonholonomic motion planning for car-like robots

A. Sánchez L.[2], J. Abraham Arenas B.[1], and René Zapata.[2]

[1] Computer Science Dept., BUAP
Puebla, Pue., México
{aarenas}@cs.buap.mx
[2] LIRMM, UMR5506 CNRS, 166 rue Ada 34392,
Montpellier Cedex 5, France
{asanchez, zapata}@lirmm.fr

Abstract. We consider the robot motion planning problem in the presence of non-integrable kinematic constraints, known as non-holonomic constraints. Car-like non-holonomic mobile robots are used for solving this problem, and several implemented planners are discussed. We claim that the randomized methods are more capable of efficiently solving many challenging, high-dimensional problems. Results with randomized planners are very promising to extend our work applying kinodynamic motion planning.

1 Introduction

Ever since Laumond's pioneering in 1986 [1], a lot of research works have addressed motion planning for non-holonomic systems in general and car-like vehicles in particular (see [2] for a recent and extensive review on this topic).

From a kinematic point of view, the main characteristic of wheeled robots (for instance, car-like robots) is the non-holonomic rolling without slipping constraint of the wheels on the floor, which forces the vehicle to move tangentially to its main axis. This reduction of the set of accessible velocities at each time makes the motion planning problem particularly difficult. The consequence is that the standard planning and control algorithms developed for robotic manipulators without constraints are no more applicable.

Car-like vehicles are archetypal non-holonomic systems: they can only move forward or backward in a direction perpendicular to the orientation of their rear wheels axle; besides their turning radius is lower bounded (because of the mechanical limits on the steering angle).

The paper is organized as follows: In Section 2 we review related work on motion planning approaches. Section 3 presents the model of the car-like robot. Section 4 describes two planners based on a two step approach, the Voronoi diagram-based planner and the potential field-based planner. Randomized approaches are introduced in Section 5, and two variants of PRM framework are implemented for solving non-holonomic motion planning problems. Finally in Section 6 we discuss our results and present future work.

2 Previous and related work

Non-holonomic motion planning refers to problems in which objects motions must satisfy non-holonomic constraints. It has attracted considerable interest in robotics [3], [2]. One approach for non-holonomic planning is to proceed in two stages [1]. First, one can generate a collision-free path that disregards the nonholonomic constraints and then transforms the path into an *admissible* one, i.e., a path that verifies the non-holonomic constraints.

This two-stage algorithm can be extended in various ways, which are all based on the idea of successive path transformation, but differ in what transformations to use and how to perform the transformations. Laumond, Jacobs, Taïx and Murray developed a two-stage planning strategy for the car-like robots [4]. First a path is found for the associated holonomic system (obtained by removing the non-holonomic constraints), and then this initial path is approximated by feasible path segments. Mirtich and Canny have given a path planner based on following a non-feasible path (a portion of the skeleton) [5]. They also introduced the notion of a metric based on shortest paths and the corresponding "Reeds-Shepp ball".

Barraquand and Latombe have considered non-holonomic motion planning from a different point of view; their planner finds a path by performing a systematic search through the configuration space, using potential fields methods to guide the search [6]. They find paths for car-like and trailer-like robots (in presence of obstacles) minimizing the number of maneuvers required, and their planner is able to generate paths for robots with relatively many degrees of freedom.

Švestka and Overmars [7] developed a motion planner based on probabilistic learning approch (so-called PRM for the community), this planner works with two different types of car-like robots: robots which can move both forwards and backwards, and robots which can only move forwards. Their method first generates configurations in C-space free ($\Re^2 \times [0, 2\pi)$, and then connects them using a RTR path local planning method. A RTR path is defined as the concatenation of a rotational path, a translational path, and other rotational path. A randomized strategy that has been used for non-holonomic planning is the RRT (Rapidly-Exploring Random Tree) approach [8]. Here, the non-holonomic constraints (e.g., the turning radius) are used to construct a tree of feasible paths emanating from the starting configuration.

In [9] Ferbach combines the two-stage approach and a so-called variational approach. It can be used for small-time controllable system. First, a collision-free path is generated. Then the non-holonomic constraints are introduced progressively.

3 Model of the car-like robot

We consider a Cartesian frame embedded in the plane. The vehicle is located with the coordinates (x, y) of the reference point, the direction θ is the angle

between the x-axis and the main axis of the robot (see Fig. 1). The robot is then completely defined as a point (x, y, θ) in the configuration space $\Re^2 \times S^1$.

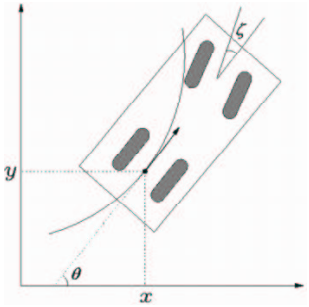

Fig. 1. Model of a car.

Car-like robot behavior is described by the following differential system:

$$(\Sigma) \begin{cases} \dot{x} = \cos\theta \cdot u_1 \\ \dot{y} = \sin\theta \cdot u_1 \\ \dot{\theta} = \frac{1}{R} u_2 \end{cases} \tag{1}$$

with $|u_1(t)| = 1$ and $|u_2(t)| \leq 1$, where u_1 and $\frac{u_2}{R}$ are respectively the linear and angular velocity of the car. Hence, without loss of generality, we assume that $R = 1$. The equations of the system (Σ) relate the derivatives $\dot{x}$, $\dot{y}$, and $\dot{\theta}$ and one cannot integrate these relations. Such a differential system expresses kinematic constraints which characterize the non-holonomic nature of the robot.

3.1 Shortest paths for a car-like robot

The study of the shortest paths for a car-like robot has an history. It has first been addressed without considering the presence of any obstacle. The pioneering result has been achieved by Dubins [10] who characterized the shape of the shortest paths when $u_1 \equiv 1$ (the robot moves always forward). More recently, Reeds and Shepp have provided a sufficient family of 48 shortest paths for the car-like robot moving forward and backward ($|u_1| \equiv 1$) [11]. Shortest paths for Dubin's car are a finite sequence of at most 3 pieces consisting of straight line segments or arcs of circle with radius 1. Optimal paths for Reeds&Shepp's car are constituted by a finite sequence of at most five elementary pieces which are either straight line segments or arcs of a circle of radius 1. A comprehensive overview on optimal trajectories for mobile robots is presented in Souères's chapter [2].

4 Approximating holonomic paths: a two-stage approach

Everything being in place, we may now define a first non-holonomic motion planning scheme for small-time controllable systems. It consists in approximating

a collision-free (holonomic) path by a sequence of collision-free admissible ones. The algorithm itself is then very simple:

- **Step 1**: Plan a collision-free path with the geometric path planner. If no such path exists, the algorithm stops: there is no solution.
- **Step 2**: Perform subdivisions on the path until all end-points can be linked to their neighbors by an admissible collision-free path.

A natural question to ask about this path-transformation method is whether it is always possible to transform a collision-free path into an admissible path that ensures the non-holonomic constraints. The answer is yes for car-like robots [1]. The convergence of Step 2 is guaranteed as soon as the steering method verifies the topological property (see Laumond's chapter in [2] for more details). Then the completeness of the algorithm only depends on the completeness of the geometric planner that computes a first collision-free path [1]. Next two subsections present both geometric planners voronoi-based and potential field-based.

4.1 Voronoi diagram-based planner

This geometric planner consists of retracting C-space free (C_{free}) into its Voronoi diagram. This diagram has the interesting property of maximizing the clearance between the robot and the obstacles. It proceeds as follows:

1. Compute the Voronoi diagram $Vor(C_{free})$,
2. Compute the points q_s and q_g and identify the arcs of $Vor(C_{free})$ containing these two points,
3. Search $Vor(C_{free})$ for a sequence of arcs $A_1, ..., A_p$ such that $q_s \in A_1$, $q_g \in A_p$ and, for all $i \in [1, p-1]$, A_i and A_{i+1} share an endpoint,
4. If the search terminates successfully, return q_s, q_g and the sequence or arcs connecting them; otherwise return failure.

Step 1 requires $O(n \log n)$ time, Step 2 takes $O(n)$ time and Step 3 also takes $O(n)$ time. Thus the overall time complexity of this algorithm is $O(n \log n)$. The most expensive step is the Step 1, but it depends on C_{free} only.

4.2 Potential field-based planner

The planner is based on the systematic use of hierarchical bitmap to represent the robot workspace. The "distributed representation" makes it possible to define simple and powerful numeric potential field techniques. The principle of this approach is to construct *collision-avoiding attractive potential fields* over the workspace. Each of these potentials applies to a selected point in the robot, called a *control point*, and pull this point toward its goal position among the obstacles. The workspace potentials are then combined into another potential

[1] A complete algorithm is guaranteed to find a collision-free path if one exists; otherwise it returns failure.

function defined over the configuration space of the robot. This new potential attracts the whole robot toward its goal configuration.

Each workspace potential is computed over the bitmap representation of the workspace (at some resolution) in such a way that it has no other local minimum than the desired final position of the robot's control point it applies to. Therefore it can be regarded as a numeric navigation function. We used the NF1 algorithm introduced in [12] for computing workspace potential field. NF1 computes the workspace potential for every control point by using a wavefront technique. The complexity of NF1 is linear in the number of points of the bitmap description and constant in the number and shape of the obstacles.

4.3 Experimental results

The basic idea of both algorithms is to generate a path that is feasible with respect to the curvature and non-holonomic motion constraints by approximating a first path found by any geometric planner. In order to link intermediate configurations along the path generated by the geometric planner, we use the shortest path from the set of shortest feasible paths in the absence of obstacles, as given by Reeds and Shepp [11]. Additional intermediate configurations are computed by recursively subdividing the geometric path whenever a collision is detected along a given feasible subpath. Hence, we start trying to reach the goal directly from the initial configuration by the shortest feasible path. If this path intersects an obstacle, the configuration halfway along the solution path of the holonomic system is used as a subgoal. The problem is thus subdivided into two subproblems. If, at any point, the minimal length path between subgoals is not collision-free, then we recursively subdivide the holonomic path. When all of the necessary subdivisions are completed, the concatenation of all feasible paths will be collision-free and will respect non-holonomic and curvature constraints.

The Voronoi diagram method computes in a single shot a roadmap that completely represents the connectivity of the C_{free}. However, the method is limited to low-dimensional C-space. Fig. 2 shows an example using the Voronoi diagram-based planner, in the left picture the holonomic path is shown, and in the right one, the feasible non-holonomic path is shown.

A significant drawback of the potential field-based planner is that it induces paths that typically graze the obstacles in the workspace. However, it is possible to use a more involved workspace potential function computed by the algorithm NF2 [12] [2]. Fig. 3 presents an example using this planner, left picture shows a holonomic path and left one the feasible path for the same scene.

One can perform an "optimization" routine to remove extra maneuvers and reduce the length path. Both planners are implemented in Java and the tests were performed on an Intel © Pentium III processor based PC running at 866 Mhz with 128 Mb RAM.

[2] The complexity of NF2 is slightly higher than that of NF1.

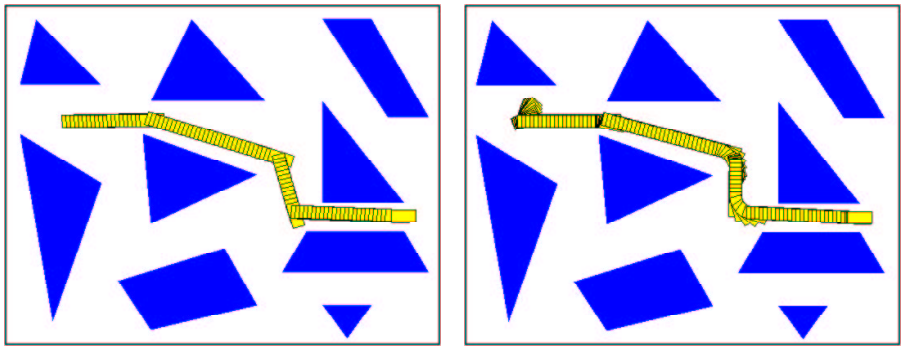

Fig. 2. A holonomic path (left picture) is computed for a complicated scene, the right figure shows the corresponding feasible non-holonomic path.

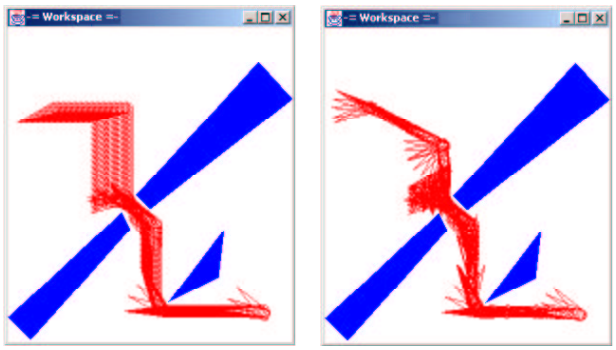

Fig. 3. In the left figure, a holonomic path in a complicated scene is shown, the right figure presents its associated feasible path. The workspace is modeled by a grid of 100×100 pixels.

5 Randomized non-holonomic motion planning

Recent results have been provided by applying a new general paradigm in motion planning [13]. This approach is splited into two phases: the *learning phase* and the *query phase*. In the learning phase, a probabilistic roadmap is incrementally constructed in C_{free}; this roadmap is a graph where arcs correspond to simple collision-free paths between the nodes. These simple motions are computed using a fast local method. In the query phase this roadmap can be used to find paths between different pairs of configurations.

Such a scheme applies for non-holonomic systems as soon as the local method is a steering method verifying the topological property. Because of the probabilistic nature of the algorithm it is difficult to analyze it. The algorithm is not complete. But fortunately for most applications the algorithm is probabilistically complete, that is, when the running time goes to infinity, the probability that a solution can be found goes to 1 (assuming a solution exists). Next subsections

present two variants of PRM-based planners for car-like robots: 1) a classical PRM, and 2) a lazy PRM.

5.1 A classical PRM planner

A car-like robot has special so-called non-holonomic constraints than restricts its motion. For example, a car cannot move sideways. Still the configuration space is 3-dimensional because, given enough space, the car can get in each position in any orientation. Using a simple straight-line (many PRM planners use this type of local method) interpolation for the local method no longer leads to valid paths for the robot. So we need to use a different local method. We use Reeds & Shepp paths as local method: given two argument configurations a and b, if the shortest path connecting a to b intersects no obstacle, the method succeeds and returns this path, otherwise a failure is returned.

5.2 A Lazy PRM approach

A recent PRM variant called the Lazy-PRM has been proposed for the problem of answering single planning queries efficiently [14], [15]. The goal of this variant is to minimize the number of collision checks. The algorithm builds a roadmap, whose nodes are the user-defined start and goal configuration, initially we assume that all nodes and edges in the roadmap are collision-free, and searches the roadmap for a shortest path between these configurations. If a collision with the obstacles occurs, the corresponding nodes and edges are removed from the roadmap and a new shortest path search procedure is applied, or new nodes and edges are added to roadmap. The process is repeated until a collision-free path is found. The resulting planner is sometimes very efficient in comparison to the original PRM. This presents a shift from the multiple query philosophy of the original PRM, and returns to the single query philosophy which was used in earlier planners. The advantage of the Lazy-PRM is that the collision checking is only performed as needed. Thus, all edges do not have to be collision checked as in the case of the original PRM.

5.3 Experimental results

The implementation is in Java and the tests were performed on an Intel © Pentium III processor based PC running at 866 Mhz with 128 Mb RAM. Fig. 4 shows an example of a scene with narrow passages. We have executed both variants of PRM using the same scene, this is with the purpose of comparing the running time of both methods. We remember that classical PRM consumes certain time of computing during the learning phase, but later in the query phase, it can resolve multiple queries "instantaneously". On the other hand, lazy PRM is a single-query planner, but in addition due to the local method it also uses certain time in the generation of the graph that unlike classic PRM the time is smaller (lazy PRM assumes that all paths are collision-free). We see that after

only 16.72 sec of learning (63 nodes), the path was computed in 0.58 sec. But lazy PRM used only 50 nodes and the path was found in 0.46 sec. Initially we use the proposed strategy in [14] for the iterative search and deletion nodes and edges (e.g., for the lazy PRM approach), but we found it to be less efficient for our computed examples. We also chose to run the A^* algorithm only once, and performed collision detection during the search.

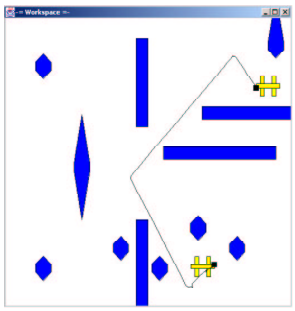

Fig. 4. A car-like robot navigating in narrow corridors.

6 Discussion and future work

Comparing the different results obtained with the planners, we can affirm that randomized approaches are more adapted to solve problems that contain complex environments and robots with many degrees of freedom. The resolution of motion planning problems that involve mobile manipulators will be possible. We believe the implications of our work put a new twist on the understanding of various motion planning algorithms. For a perspective, consider the following methods in light of our work.

- Potential field methods attempt to improve search by using heuristics to escape local minima [12]. For our implementation, the search is even performed over an implicit grid. Thus, it can be considered as a lazy PRM on a grid lattice, with A^* search replaced by the potential field method.
- The basic PRM proposed the use of substantial preprocessing to construct a roadmap based on random sampling in an attempt to reduce the exponential number of samples needed for a grid-based approach.
- The lazy PRM provides a kind of missing link between classical grid search and the basic PRM. The spirit is much like classical search (assuming only one execution of A^* algorithm), but occurs over a randomly-generated graph, instead of an implicitly defined lattice.

Using the PRM framework, one can derive a customizable version. For example, solving non-holonomic motion planning problems when the car-like robots are different turning radii.

6.1 Towards randomized kinodynamic planning

The primary difficulty with existing techniques is that although they are powerful for standard motion planning, they do not naturally extend to general problems that involve differential constraints. The potential field-based planner is efficient for holonomic planning, depends heavily on the choice of a good heuristic potential function, which could become a daunting task when confronted with obstacles and differential constraints.

In the PRM approach, a graph is constructed in the configuration space by generating random configurations and attempting to connect pairs of nearby configurations with a local method that will connect pairs of configurations. For planning of holonomic systems or steerable non-holonomic systems, the local planning step might be efficient; however, in general the connection problem it can be as difficult as designing a non-linear controller, particularly for complicated non-holonomic and dynamical systems. PRM technique might require the connections of thousands of configurations or states to find a solution, and if each connection is akin to a non-linear control problem, it seems impractical for solving problems with differential constraints. Furthermore, the classical PRM is designed for multiple queries.

RRT approach [8], offers benefits that are similar to those obtained by successful randomized planning methods (for instance, PRM approach) but apply to a much broader class of problems.

For example, in Fig. 5 is depicted an example for a car-like vehicle under differential constraints. The basic idea is to use control-theoretic representations, and incrementally grow a search tree from an initial state by applying control inputs over short time intervals to reach new states. Each vertex in the tree represents a state, and each directed edge represents an input that was applied to reach the new state from a previous state. When a vertex reaches a desired goal region, an open-loop trajectory from the initial state is represented. We have used the Motion Strategy Library for this example [8].

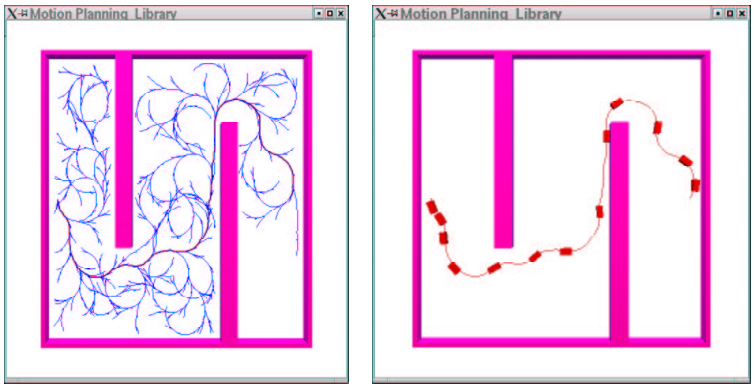

Fig. 5. The car can move forward only, and a 2D projection of the RRT is shown.

One can formulate the kinodynamic problem [16] as motion planning in state space that has first-order differential constraints and obstacle-based global constraints. Kinodynamic planning is treated as a motion planning problem in a higher dimensional state space, which serves the same role as the configuration space for basic motion planning. Kinodynamic is a topic of future work and research.

References

1. Laumond, J-P. Feasible trajectories for mobile robots with kinematic and environmental constraints. In Proc. Int. Conf. on Intelligent Autonomous Systems (1986) 346-354
2. Laumond, J-P(Editor). Robot motion planning and control. Lecture notes in control and information sciences 229. Springer (1998)
3. Latombe, J-C. Robot Motion Planning. Kluwer Academic Publishers (1991)
4. Laumond, J-P., Jacobs, P. E., Taïx, M., Murray, R. M. A motion planner for nonholonomic mobile robots. IEEE Transactions on Robotics and Automation. Vol 10, No. 5 (1994) 577-593
5. Mirtich, B., Canny, J. Using skeletons for nonholonomic path planning among obstacles. Proc. IEEE Int. Conf. on Robotics and Automation (1992)
6. Barraquand, J., Latombe, J-C. On non-holonomic mobile robots and optimal maneuvering. Revue d'Intelligence Artificielle. Vol 3, No. 2 (1989) 77-103
7. Švestka, P., Overmars, M. H. Motion planning for carlike robots using a probabilistic learning approach. The International Journal of Robotics Research. Vol 16, No. 2 (1997) 119-143
8. LaValle, S. M., Kuffner, J. J. Rapidly-exploring random trees: Progress and prospects. Workshop on the Algorithmic Foundations on Robotics (2000)
9. Ferbach, P. A method of progressive constraints for nonholonomic motion planning. IEEE Transactions on Robotics and Automation. Vol 14, No. 1 (1998) 172-179
10. Dubins, L. E. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. American Journal of Mathematics, Vol 79 (1957) 497-516
11. Reeds, J. A., Shepp, R. A.: Optimal paths for a car that goes both forward and backwards. Pacific Journal of Mathematics. 145(2) (1990) 367-393
12. Barraquand, J., Latombe, J-C. Robot motion planning: A distributed representation approach. The International Journal of Robotics Research, Vol 10, No. 6 (1991) 628-649
13. Kavraki, L. E., Švestka, P., Latombe, J-C., Overmars, M. H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation. Vol 12, No. 4 (1996) 566-579
14. Bohlin, R., Kavraki, L. E. Path planning using lazy PRM. Proc. IEEE Int. Conf. on Robotics and Automation (2000)
15. Sánchez, L. A., Zapata, R., Arenas, B. J.A. Motion planning for car-like robots using lazy probabilistic roadmap method. Lecture Notes in Artificial Intelligence series (Vol 2313). Springer Verlag (2002) 1-10
16. Donald, B., Xavier, P., Canny, J., Reif, J. Kinodynamic motion planning. Journal of the ACM, Vol. 40, No. 5 (1993) 1048-1066