A Planning Method to Obtain Good Quality Paths for Autonomous Cars

Domokos Kiss, Gábor Csorvási and Ákos Nagy
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
domokos.kiss@aut.bme.hu, gabor.csorvasi@aut.bme.hu, akos.nagy@aut.bme.hu

***Abstract*—Path planning among obstacles for nonholonomic systems is a widely researched area nowadays, but it is still one of the most challenging problems in autonomous navigation. We have recently presented a rapidly exploring random tree based global planner (RTR) and a steering method (C*CS) for car-like vehicles, which uses circular and straight movements. With the aid of these two methods it is possible to obtain a feasible path, even in narrow spaces and in situations requiring non-trivial maneuvers. This paper presents an improved solution for environments, where not all obstacles are known at the beginning and these are discovered during the motion of the robot. We also introduce an extension to the presented algorithm to achieve paths of better quality, i.e. more similar to a reasonable path generated by a human driver.**

I. Introduction

Driving support systems (e.g. parking assistant, collision avoidance) are increasingly widespread nowadays. In some application these intelligent systems plan the path for the vehicle from a start position to the desired goal position. Due to the presence of obstacles this is not an easy task at all. Kinematic constraints of the vehicle are also make path planning difficult. Cars have nonholonomic constraints which cause difficulties in the control of such robots, even in the absence of obstacles. These constraints cause that a car can only move in the direction of its heading (forwards or backwards) and that the turning radius is lower bounded. For this reason parallel parking with a car is relatively difficult.

From the perspective of motion planning, any robot can be described by its *configuration*. For example, the configuration of a rigid mobile robot moving in a planar workspace $\mathcal{W} \subset \mathbb{R}^2$ can be given by $q = (x, y, \theta)$, a vector of its position and orientation in the configuration space $\mathcal{C}$. The set of not allowed configurations (e.g. because of collision with obstacles) is called the configuration space obstacle $\mathcal{C}_{obs}$, its complement is the free space $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$.

The paper is organized as follows. A brief overview of path planning methods for autonomous robots are given in Section II. In Section III, we shortly discuss our previously developed RTR and C^*CS planning methods and show how these can be applied together to obtain feasible paths in narrow environments. Section IV shows our enhanced method for sampling of the configuration space in the C^*CS algorithm. In Section VI, we present an extension to our path planning methods in order to generate human like paths. We show a planning strategy for environment, where farther obstacles are not known at the beginning in Section V. Conclusions and directions of future work are summarized in Section VII.

II. Related Work

Path planning for nonholonomic robots is not trivial even in absence of obstacles in the environment [1]. Algorithms that can solve this task are called local planners or steering methods. In case of specific wheeled robots exact methods exist for computing optimal (e.g. shortest length) local paths. These include our investigated car-like robot model [2], [3].

However, a useful planning algorithm has to generate paths in the presence of obstacles while taking into account the kinematic constraints of the vehicle as well. To the best of our knowledge, there is no general *optimal* solution available for this problem. Thus generally, if obstacles are present, one should be satisfied with a *feasible* solution which is not necessarily optimal. The majority of planning algorithms delivering a feasible solution can be grouped into two main categories. The first category consists of techniques that approximate a not necessarily feasible but collision-free initial geometric path by a sequence of feasible local paths obtained by a steering method [4]. In this case firstly a primary geometric path is planned which takes the obstacles and the shape of the robot into account, but does not care about the kinematic constraints. In a second phase, this path is iteratively subdivided and the parts are tried to be connected by a steering method. If the geometric path has a nonzero clearance ε from the obstacles, and the steering method verifies the so called *topological property*, then the approximation succeeds in finite time [5].

The second category involves sampling-based roadmap methods, which build a graph in order to capture the topology of $\mathcal{C}_{free}$ and use local steering methods to connect the graph nodes. A good survey of sampling-based planning methods can be found in [6]. The majority of these are based on random sampling of the configuration space. Their popularity arise from the fact that they do not require an explicit representation of $\mathcal{C}_{obs}$. These methods proved to be successful in many planning problems, including high-dimensional configuration spaces. For example, the Probabilistic Roadmap Method (PRM) [7] and the Rapidly exploring Random Trees (RRT) [8], [9] algorithms are common sampling based planners. The main idea of it is to incrementally build a search tree starting from the initial configuration in a way that the tree covers the free space rapidly and with gradually increasing resolution.

III. The RTR+C*CS Path Planner

Our global path planner (RTR) and steering method (C*CS) have been presented recently [10], [11]. The global path designed by RTR consist of translational motion (T) and turning

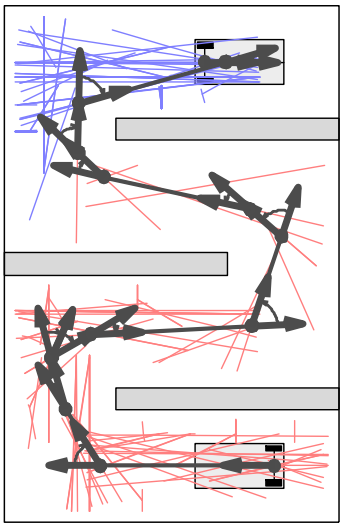

Fig. 1. A result of the RTR planner through a narrow corridor

in plane (R). Therefore the path is not feasible for car-like robots, but a good starting point for the steering method, because the path is collision free and the translational primitives are feasible for car-like robots. The global path is approximated by the C*CS local planner, which uses straight segments and circular arcs of lower bounded (but not necessary minimal) radii. The purpose of the approximation is to eliminate in-place turns while avoiding collisions.

A. RTR Global Planner

The RTR planner is based on the well-known Rapidly exploring Random Trees (RRT) algorithm [12]. RTR is similar to a bidirectional RRT method, because it also builds two tree from the start and goal configuration (*start tree*, *goal tree*). However, the phases (sampling, vertex selection and extension) of the RTR planner are different from RRT.

The first difference to RRT can be found in the sampling step. It returns a guiding position p_G instead of a configuration, which can be treated as a one-dimensional continuous set of configurations, from which any element can serve as local goal in the tree extension step.The vertex selection step returns the configuration in the existing tree which has the smallest *position* distance to p_G. This step uses a simple Euclidean metric, hence no special configuration space metrics are needed.

The main difference to the RRT method can be found in the tree extension step. It performs some primitive rotate–translate motion pairs, guided by p_G. A rotation is applied to reach the orientation pointing to p_G, and a consecutive translation is performed in both forward and backward directions until the first collision. If a collision occurs during the rotation, the rotation is tried again in the other turning direction. An example for the resulting RTR-path can be seen in Fig. 1, the two trees are depicted as well by different colors.

B. C*CS Steering Method

The main idea behind the C*CS local planner it to use circular and straight movements as path primitives [13], [14]. With the help of these primitives we can arrive from a q_I initial

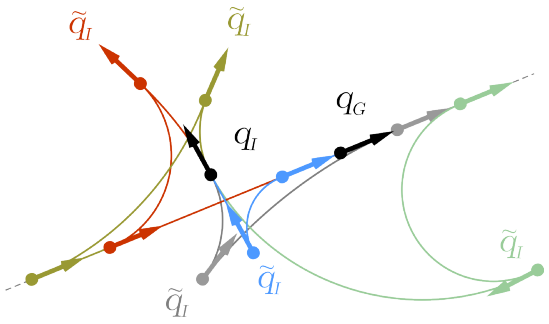

Fig. 2. There are many C^*CS paths between two configurations

configuration to any q_G goal. To simplify the notation we will use the letter C for the circular arcs and S for the straight segments in the sequel. It can be easily show that without taking into account the curvature bound of a car-like robot and assume that $\theta_I \neq 0$ the goal configuration can be reached as follows. With a tangential arc we turn to an intermediate goal $\tilde{q_G}$ which is in the line of the q_G. After that only a final straight segment is needed to arrive in the goal.

Unfortunately if the radius of the tangential arc is smaller than the minimal turning radius of the car or it is initially parallel to the line of the goal no feasible path can be found. To help that an intermediate $\tilde{q_I}$ configuration has to be reached where a feasible path is given. It is shown that we have infinite possibility to choose one of this configuration. The set of configuration, which is reachable from the initial configuration, is called Arc Reachable Manifold (ARM [15])

IV. Improvements in Sampling Phase

Among obstacles infinite possibility is useful, thus there is more possibilities to find a feasible path, and the resulted paths can be more ”prettier“ as well. Although in the implementation we have to use a form of sampling method. Naturally if the sample size is bigger the calculation takes more time, but if only few sample is taken, we can miss the possible solution.

In our early implementation we used a Shukarev-grid based sampling method, but this had more drawbacks. The minimally needed sample size was depended on the actual environment, and a lot of calculation was unnecessary for example in far spaces, or next to the car which is unreachable for an non-holonomic car-like robot.

Our new method is based on the behaviour of these robots, and only the closed environment is sampled. To avoid the unnecessary calculation caused by the unreachable area at the two sides of the robot, we sampled the curvature of the possible circular arcs from 0 to κ_max, where $\kappa_max = \frac{1}{r_min}$. Also the distance from the robot is sampled from 0 to a chosen d_max. Our results shows that the $d_max = 2\dot{r}_min$ could be a suitable choice. The first step results tangential circles on both side of the robot and the second step results co-centric circles as seen in image 3. The intersections of these circles are the sample points. The extra time needed for the calculation of the intersection is relatively small compared to the whole process.

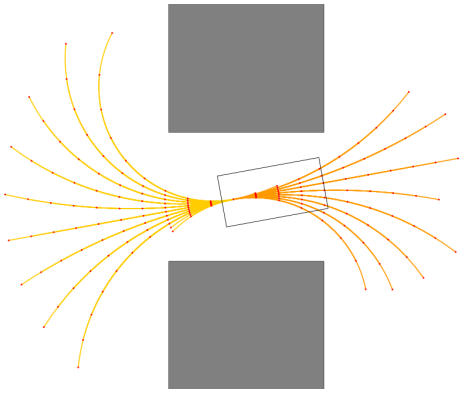

Fig. 3. Arc Reachable Manifold in an environment with two obstacles

V. Planning in Dynamic Environment

Our presented path planning method requires global information about the environment. During collision detection in the RTR and the C*CS algorithm, all obstacles of the environment are checked. In most real situation we do not have global information of the environment or the obstacles of the environment change their position. In both cases the planner algorithm has to be run repeatedly if the environment is changed and the previously generated path is collided.

Consider the following scenario. A 3D laser scanner is located on the top of a car-like robot. This sensor provides the information about the obstacles near the car for the planning algorithm. In our simulation we model this situation with the following way. The robot only detects the obstacles near the current configuration within a fixed radius. In every simulation iteration the program checks whether an obstacle within the sensing radius is discovered. If the new obstacle collide with the path of the robot, the path will be re-planned. After that, the robot steps forward to the next path configuration.

The algorithm of the simulation process can be seen in Algorithm 1. The $\mathcal{E}$ indicates the structure of the environment. This structure contains the geometrical representation of the obstacles. The $\mathcal{P}_{global}$ and the $\mathcal{P}_{approx}$ indicate the global path generated by RTR planner and the approximated path feasible for car-like robots (C*CS planner). Both path are described with a configuration list. The loop variable of the simulation is i and we run the simulation for L iteration.

One of the most important factors for this simulation scenario are planning time and the quality of the path. The RTR and the C*CS planners run time definitely determinate the planning time. In this Section we will focus on planning time and path quality. The quality of the path will be discussed in Section VI too.

We compare three difference planning strategy for dynamic environment. The strategies differ in re-planning method when the path is collided:

- *Global Re-Planning*: The path is re-planned from the robot current configuration to the goal configuration.
- *Local Re-Planning*: The path is re-planned from the robot current configuration to the end of the collided path segment.

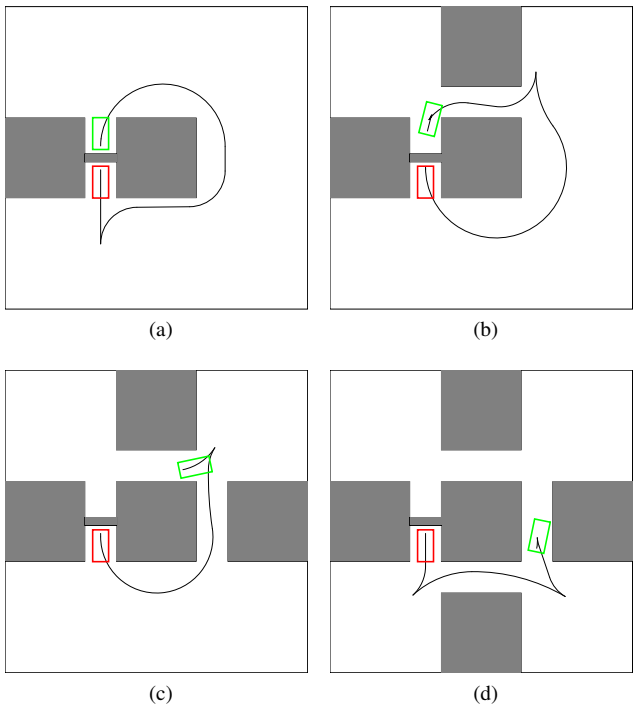

Fig. 4.

- *Global Re-Planning with Tree Reservation*: Same as the first strategy, but the goal tree in the RTR algorithm is remain.

Fig. 5 shows *Global Re-Planning* strategy for two situation. In these figures, we marked the configuration of the robot when it detects a new obstacle, the collided path segment and the goal configuration as well. In the first scenario (see Fig. 5a, 5b), the path takes from the lower, left corner to the lower, right corner. We can see, that planning to the goal configuration makes a natural path for car-like robots in this situation. However, in the next scenario (Fig. 5c, 5d) this planning method regenerates a long path segment unnecessary. In this situation, replanning only the collided segment would cost less computing capacity and makes more simple path as well.

The *Local Re-Planning* strategy is shown in Fig. 6. In the first situation (Fig. 6a, 6b), this method generates complicated path, because the planner tries to connect the new path segment to existing one. In this strategy, there is an extra parameter, which determinate the distance from the end point of the collided segment to the end point of the generated segment. If this parameter set to infinity, the strategy will be *Global Re-Planning* method. In the second scene (Fig. 6c, 6d), the local planning makes a simple, natural solution and do not re-plan unnecessary.

The *Global Re-Planning with Tree Reservation* method generates path to the goal configuration like the *Global Re-Planning* algorithm. The tree reservation is an improvement in the RTR algorithm. After the first run of the global planner, the RTR does not rebuild the entire *goal tree*, instead it uses the tree built in the first iteration. As a disadvantage of this improvement, the planner has to remove the collided nodes

TABLE I. RUN TIMES FOR DIFFERENT PLANNING STRATEGIES

Scene	Global Re-Pl.	Local Re-Pl.	Global Re-Pl. with T. R.
Scene 1	64.9375ms	56.1083ms	44.375ms
Scene 2	7.2250ms	7.625ms	6.425ms
Scene 3	8.40875ms	8.27795ms	12.3391ms

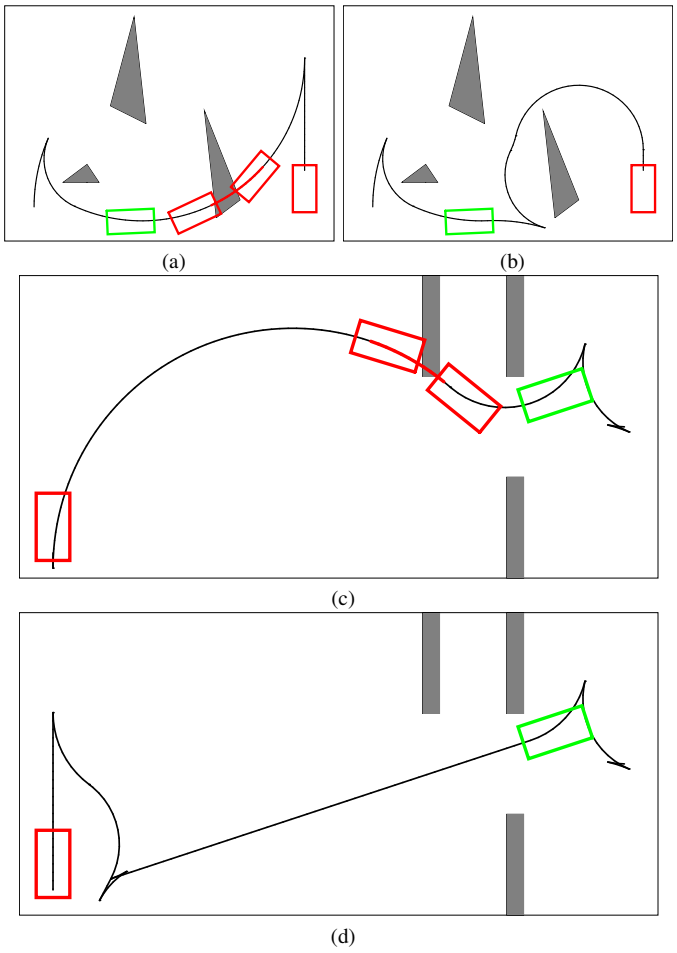

Fig. 5.

and edges from the *goal tree* and it takes time to check the whole tree.

In TABLE I we compare the three different planning strategies run time. We have run twenty measurement for every scene, strategy, and averaged the results. The algorithms were implemented in C++ and tested on a notebook computer with Intel i7-3630QM processor running at $2.4GHz$ and having 8 GB RAM.

VI. IMPROVEMENTS OF THE RTR GLOBAL PLANNER

The final path is greatly depends on the global path. Because of the "random" nature of the RTR global planner the resulted paths could be very far from a path which is planned by a human driver. To improve the global planner it is necessary to define the [qualities] of a path. We tried to minimize the total path distance, the numbers of segments and the absolute turning angle, and to maximize the distance from the obstacles.

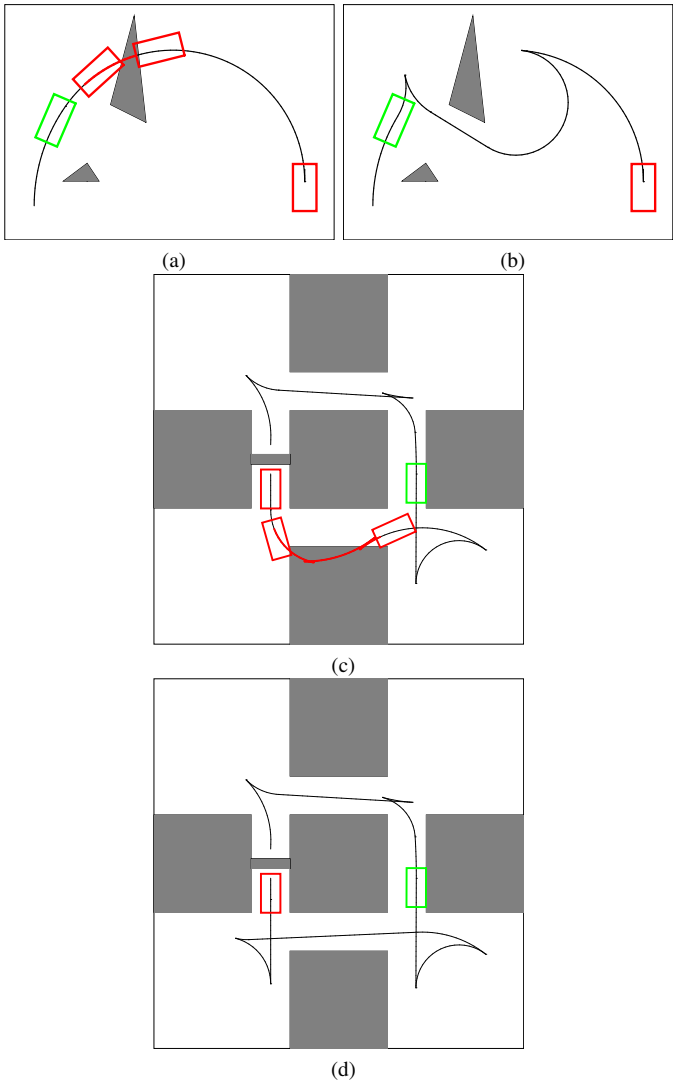

Fig. 6.

Algorithm 1 The simulation method for planning in dynamic environment

$\mathcal{P}_{\text{global}} \leftarrow$ GLOBAL_PLANNER$(\mathcal{E})$
$\mathcal{P}_{\text{approx}} \leftarrow$ LOCAL_PLANNER$(\mathcal{E}, \mathcal{P})$
for all $i = 1$ to L **do**
  if CHECK_OBSTACLES$(\mathcal{E})$ **then**
    if CHECK_COLLISION$(\mathcal{E}, \mathcal{P})$ **then**
      $\mathcal{P}_{\text{global}} \leftarrow$ GLOBAL_PLANNER$(\mathcal{E})$
      $\mathcal{P}_{\text{approx}} \leftarrow$ LOCAL_PLANNER$(\mathcal{E}, \mathcal{P})$
    end if
  end if
end for

VII. CONCLUSIONS AND FUTURE WORK

ACKNOWLEDGEMENT

This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013) organized by VIKING Zrt. Balatonfüred. This work was partially

supported by the Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fond through project eAutoTech (grant no.: KMR_12-1-2012-0188).

REFERENCES

[1] J.-P. Laumond, *Robot Motion Planning and Control*, ser. Lecture Notes in Control and Information Sciences. Springer, 1998, vol. 229.

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

[3] P. R. Giordano, M. Vendittelli, J.-P. Laumond, and P. Souères, "Nonholonomic distance to polygonal obstacles for a car-like robot of polygonal shape," *IEEE Trans. Robot.*, vol. 22, pp. 1040–1047, 2006.

[4] J.-P. Laumond, P. E. Jacobs, M. Taïx, and R. M. Murray, "A motion planner for nonholonomic mobile robots," *IEEE Trans. Robot. Autom.*, vol. 10, pp. 577–593, 1994.

[5] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*, ser. Lecture Notes in Control and Information Sciences, J.-P. Laumond, Ed. Springer, 1998, vol. 229.

[6] S. M. LaValle, "Sampling-based motion planning," in *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available online at http://planning.cs.uiuc.edu/.

[7] L. E. Kavraki, P. Svetska, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, pp. 566–580, 1996.

[8] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep., 1998.

[9] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, pp. 378–400, 2001.

[10] A. Nagy, G. Csorvási, and D. Kiss, "Path planning and control of differential and car-like robots in narrow environments," in *2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Herlany, Slovakia, January 2015, pp. 103–108.

[11] G. Csorvási, A. Nagy, and D. Kiss, "Rtr+c*cs: An effective geometric planner for car-like robots (accepted, publishing in progress)," in *Proceedings of the 16th International Carpathian Control Conference (ICCC'2015)*, Szilvasvarad, Hungary, May 2015.

[12] D. Kiss and G. Tevesz, "The RTR path planner for differential drive robots," in *Proceedings of the 16th International Workshop on Computer Science and Information Technologies CSIT2014*, Sheffield, England, September 2014.

[13] D. Kiss, "A path planner for car-like robot maneuvering in narrow environments," in *Proceedings of the Automation and Applied Computer Science Workshop, AACS'2013*, Budapest, Hungary, June 2013, pp. 82–93.

[14] D. Kiss and G. Tevesz, "A steering method for the kinematic car using C*CS paths," in *Proceedings of the 2014 15th International Carpathian Control Conference (ICCC)*, Velké Karlovice, Czech Republic, May 2014, pp. 227–232.

[15] J. Minguez and L. Montano, "Extending collision avoidance methods to consider the vehicle shape, kinematics, and dynamics of a mobile robot," *IEEE Trans. Robot.*, vol. 25, pp. 367–381, 2009.