Proceedings of the
2007 IEEE Intelligent Vehicles Symposium
Istanbul, Turkey, June 13-15, 2007

Navigation of Autonomous Vehicles in Unknown Environments using Reinforcement Learning

Tomás Martínez-Marín and Rafael Rodríguez

Abstract— **In this paper we propose a generic approach for navigation of nonholonomic vehicles in unknown environments. The vehicle model is also unknown, so the path planner uses reinforcement learning to acquire the optimal behaviour together with the model, which is estimated by a reduced set of transitions. After the training phase, the vehicle is able to explore the environment through a wall-following behaviour. In order to guide the navigation and to build a map of the environment the planner employs virtual walls. The learning time to acquire a good approximation of the wall-following behaviour was only a few minutes. Both simulation and experimental results are reported to show the satisfactory performance of the method.**

I. INTRODUCTION

Intelligent vehicles should exhibit an autonomous behaviour learning from experience through interaction with the environment. Furthermore, they should learn and update their internal dynamic models on-line and maximize its short term capabilities. Most of the autonomous vehicles we can find in the industry today follow a strict itinerary with a very limited interaction with the environment. In fact, the environment has been thoroughly adapted to the vehicles, which is just the proof of lack of intelligence.

Car-like vehicles are widely used in industry since they have the necessary loading capability with only a power motor. These vehicles are nonlinear dynamic systems, whose motion laws have been a well studied topic [1], [2], [3], [4]. However, the motion planning of car-like vehicles is a difficult task since they are nonholonomic systems. Our approach addresses the motion optimisation, since intelligent vehicles should provide an optimal behaviour in real scenarios with restricted computational resources.

A number of planning algorithms have been proposed for generating trajectories that give rise to smooth motion [5], [6]. These trajectories are computed in open-loop, but in real environments where a vehicle is subject to perturbations and uncertainty, closed-loop action is more desirable. Minimal length paths of simplified car-like vehicles have been characterized in [3]. This result is proven in the absence of obstacles and it is computed in open-loop. In [7] a grid-based method for motion planning in the presence of obstacles was developed. The method supposes that the vehicle model is known and the search mechanism is open-loop, although the method can be easily modified to work in a closed-loop manner. Dynamic Programming (DP) provides a closed-loop solution including obstacles [8]. Although DP is efficient compared with direct search, it requires a lot of computational resources and the vehicle model.

In this paper, we present a generic approach to learn to navigate nonholonomic vehicles by interaction with the environment. The algorithm combines reinforcement learning with some concepts of the CACM technique [9], which is based on cell-to-cell mapping techniques and Bellman's principle of optimality for continuous dynamic systems. Reinforcement learning (RL) methods provide a great advantage with respect to the cited approaches, since the optimal motion law of a car-like vehicle is estimated on-line while it is interacting with the environment. Thus, the RL controller does not require a previous vehicle model to obtain an optimal behaviour of the real vehicle. In this context, there are several applications of robot motion planning by reinforcement learning. For example, in [10] a robot docking task was solved through RL in a visual servoing framework. The optimal motion of car-like vehicles is slightly more difficult since the system is nonholonomic and the dimension of the problem is higher (3D or more).

The paper is organized as follows. Section II describes the vehicle platform. In section III we present some basic concepts of reinforcement learning. Section IV provides a brief introduction to the Cell Mapping techniques. Implementational aspects of the new algorithm are addressed in section V, tested in section VI and VII through simulations and experimentation on a real vehicle, respectively. Conclusions appear in section VIII.

II. VEHICLE MODEL

The reinforcement learning controllers have been implemented in the nonholonomic vehicle shown in Fig. 1. The vehicle is equipped with an array of infrared sensors, a laser scanner and a CMOS camera; although the image sensor has not been employed in this application. The vehicle is autonomous, using a microcontroller MPC555 to process all sensors and the RL controllers.

The state space formulation [3] of the vehicle model we will use is the following:

$$\dot{x} = v_T \cos\theta \cos\varsigma, \quad (1)$$

$$\dot{y} = v_T \sin\theta \cos\varsigma, \quad (2)$$

$$\dot{\theta} = v_T \sin\varsigma. \quad (3)$$

where v_T is the translational velocity and ς is the steering angle of the vehicle.

The distance between the reference point (x, y) and the middle point of the driving wheels is 0.32 m. The orientation

Tomás Martínez-Marín is with the Department of Physics, System Engineering and Signal Theory, University of Alicante, Spain.
Rafael Rodríguez is with Brainstorm Multimedia, Valencia, Spain.

Fig. 1. The autonomous vehicle employed for the experiments.

of the car is denoted by θ. The two control variables of a car are the velocity v_T of the driving wheels and the steering angle ς. It is important to note that the state equations are only used in the simulations to describe the robot trajectories. In the experiments, the model is built on-line by recording the transitions between states and the immediate rewards.

Observing the state space equations, we can see that the time derivatives of the state variables only depend on the orientation variable θ. Therefore, to obtain information about the relative motion of the car (i.e., transitions between adjoining cells) it is enough to obtain the cell transitions only for the third state variable, starting each transition from the origin in the xy plane. The former is only valid if the low level controllers can keep the specified velocities on different surfaces. Using a transformation matrix (T) we can exploit this property. Thus, after learning a few local trajectories the algorithm only spends time looking for the global optimal policy.

Although the state space of the system is three-dimensional, some vehicle behaviours, such as wall following, can be specified in a two-dimensional state space. In this case, the problem is simplified by fixing the value of v_T. Then, the task is reduced to control the orientation of the vehicle (ς) in a two-dimensional state space (d, β), where d is the distance between the vehicle and the wall, and β is the relative orientation of the vehicle with respect to the wall. In our reinforcement learning experiments, we only allow three possible actions in each state (-20, 0, 20 deg). In this case the controllability of the system is more limited, since the controller cannot reduce v_T if major changes in the orientation of the vehicle are required.

III. REINFORCEMENT LEARNING

Reinforcement learning methods only require a scalar reward to learn to map situations (states) in actions [11]. As opposed to supervised learning, they do not require a teacher to acquire the optimal behaviour, they only need to interact with the environment learning from experience. The knowledge is saved in a look-up table that contains an estimation of the accumulated reward to reach the goal from each situation or state. The objective is to find the actions (policy $a = \pi(s)$) that maximize the accumulated reward in each state. Q-learning is one of the most popular reinforcement learning methods, since with a simple formulation it can address model-free optimization problems. The accumulated reward for each state-action pair $Q(s, a)$ is updated by the one-step equation

$$\Delta Q(s,a) = \alpha \left(r + \gamma max_{a'} Q(s', a') - Q(s,a) \right) \tag{4}$$

where Q is the expected value of performing action a in state s, r is the reward, α is a learning rate which controls convergence and γ is the discount factor. The discount factor makes rewards earned earlier more valuable than those received later. The action a with highest Q value at state s is the best policy up to instant t. When $t \rightarrow \infty$ the policy approximates the optimal behaviour:

$$a^* = \pi^*(s) = arg\ max_{a'} Q(s, a') \tag{5}$$

In applications with real systems Q-learning spends a long time making hundred of thousands of trials to approximate the optimal behaviour. To speed up the learning process it is necessary to incorporate some planning mechanism. Prioritized sweeping [12] and Dyna-Q include a search control to simulate the past real experiences in a specified order. In our simulations we will combine Q-learning with the search mechanism proposed in the new RL algorithm. The quality of the optimal solution has been compared with the popular Q-learning algorithm in [13].

IV. ADJOINING CELL MAPPING

Q-learning was conceived for discrete state and action spaces, where the state space is not necessarily metric. In robotic applications the state space is continuous, so it is mandatory to discretize the state space into cells. The inherent discretization errors can produce a poor approximation to the optimal behaviour in complex nonlinear systems such as mobile robots. Cell Mapping techniques were conceived in order to deal with the discretization problems in an efficient way.

Cell-to-cell mapping methods are based on a discretization of the state variables of the system, defining a partition of the state space into cells [14]. A cell-to-cell mapping can be derived from the dynamic evolution of the system. In [9] the CACM algorithm for optimal control of highly nonlinear systems is proposed. This method is based on the Adjoining Cell Mapping (ACM) technique, whose central concept is the creation of a cell mapping where only transitions between adjoining cells are allowed [15].

The adjoining property states that the distance D_k between the current cell and the previous cell is equal to some integer value k equal or greater than 1. For our RL controller, we will define the adjoining property in terms of the continuous states $\mathbf{x}$ and $\mathbf{x}'$ as follows:

$$D_k(\mathbf{x}, \mathbf{x}') = \max_j |\frac{x_j - x'_j}{h_j}| = k, \quad (6)$$

where x_j indicates the j-th component of the state $\mathbf{x}$ and h_j is the cell size of the j-th dimension.

In Q-learning the transitions between states are evaluated at fixed sample times, while with our RL controller the transitions have to satisfy the adjoining distance condition in order to be evaluated. By appropriate selection of this distance with respect to the number of cells, it is possible to minimize quantization effects and better approximate the optimal behaviour of the system.

V. THE RL ALGORITHM

The new algorithm has been implemented as a model-based reinforcement learning method, where the back-ups are made by simulation following the shortest path search backward in time, starting from the goal state. In direct reinforcement learning, the back-ups are only made by experimentation, which is suitable when the back-up time for experimentation compared with simulation is not very high. In general, model-based reinforcement learning find better trajectories and manages changes in the environment (e. g. obstacles) and the goals more efficiently than direct reinforcement learning.

The RL algorithm deals with several data structures for organizing the available information and storing the partial results of the learning process. These structures are the following:

- $Q(s, a)$: is the Q-value table where the accumulated reward for the (s, a)-pair is saved. From this table, the optimal policy is obtained according to 5.
- $M(k, x, a)$: is the local vehicle model. It contains an average of relative transitions of the car transformed to local coordinates during the learning process. The transitions have to satisfy the $D - k$ adjoining property to assure a good approximation to the optimal policy.
- $Queue$: contains the states to be updated in the correct order to find the optimal policy in only one sweeping.
- $policy(s, M, Q)$: selects a ϵ-greedy policy to estimate the model (M) and to exploit the best policy acquired (Q).
- $dist(s, s')$: is the maximum norm between states s and s'.

The RL algorithm that implements the concepts described above is presented in Fig. 2(see [11] for notation details). The state is represented in the algorithm by a real valued vector x, which is converted to the discrete state s (integer index) by the function $cell()$. In our experiments, uniform discretization was used with 41 cells per variable (see Table I for full details of the RL parameters). The function $Dk{-}adjoining()$ is used to determine whether the adjoining property has been satisfied. The index s is used to update the Q-table, and x is used to update the function M. Since the controller uses noisy data from an image sensor, the function

Initialize $Q(s,a)$, $M(k,x,a)$ and $first(Queue) \leftarrow$ goal					
	$x \leftarrow$ current state				
	$s \leftarrow cell(x)$				
	IF	s = sink or goal			
	THEN	$reactive(x)$			
	ELSE	$a \leftarrow policy(s, M, Q)$			
		Execute action a			
		Observe resultant state x' and reward r			
		IF	D_k-adjoining(x, x')		
		THEN	FOR $k = 1, 2, ..., K$		
				$M(k, x, a) \leftarrow x', r$	
				$k = min\{K, dist(goal, s)\}$	
				$s' \leftarrow first(Queue)$	
				FOR all (s, a) that led to s'	
					$\bar{x} \leftarrow M(k, x', a)$
					$s \leftarrow cell(\bar{x})$
					Update $Q(s, a)$ using Eq. 4
					Insert s in $Queue$ sorting by $max_a Q(s, a)$
				UNTIL N times	
				If $Queue$ is empty: $first(Queue) \leftarrow$ goal	
UNTIL training terminated					

Fig. 2. Reinforcement Learning Algorithm.

$M()$ estimates the state of the system by filtering before storing it. In our experiments an average filter was used.

For the wall following behaviour the aim of the controller is to move the vehicle from any initial position inside the region of interest to a goal position (an objective distance from the wall) through a minimum-time trajectory. A trial finishes when the vehicle moves outside of the state space (sink cell) or when it enters in the goal. Then, the function $reactive()$ moves the vehicle in the opposite direction, until some starting position inside the state space is reached.

The function $policy()$ selects an action for each transition of the system. The RL controller selects the actions randomly to explore most of the state space during training. By changing the function $policy()$ it is possible to implement other controllers. In the update rule 4, the learning rate α is variable, falling inversely with the number of transitions and the discount factor is fixed to $\gamma = 1$.

The vehicle should move forward and backward to avoid possible obstacles in the way. For that reason, during the training phase two RL controller are built: one for forward motion and the other for backward motion. Thus, employing both controllers the vehicle is able to avoid obstacles and turn around (see experimental results in Fig. 4) in a natural manner. This controller is suitable to explore unknown environments creating a vehicle model and a map which later can use a three-dimensional RL controller or a CACM controller [13]. In this way, the path planning can be improved using a control architecture with two levels of abstraction. On the other hand, the vehicle can reach any position and orientation

State variables: 2. (1661 states)	x_1: -1 $\leq d \leq$ 1 m. Cells: 41 x_2: -100 $\leq \beta \leq$ 100°. Cells: 41
Objective state:	(d, β) = (0 m, 0°)
Control variables: 2. (3 actions)	u_1: -0.2 $\leq v_T \leq$ 0.2 m/s. (u_2: -20 $\leq \varsigma \leq$ 20°.)
Sampling time:	T_s: 0.1 sec.
Reward:	r = 100 if goal r = -20 if sink r = -n (T_s) otherwise
Adjoining distance:	D-2

TABLE I

PARAMETERS IN THE RL ALGORITHM ON THE REAL VEHICLE.

using the virtual wall concept. A virtual wall is created by manipulating the vehicle sensors in such a way that although the RL controller remain unchanged, the vehicle can navigate on a free space without any physical guide.

VI. SIMULATION RESULTS

The algorithm proposed has been tested in simulation using the vehicle model given in section II. The state space needed to learn the wall following behaviour is two-dimensional (d, β). The variables values and parameters are depicted in Table I.

The optimal behaviour of the RL controller is depicted in Fig 3. We can observe that the controller approximates the time-optimal behaviour: bang-bang control with the classical *switching curve* in the centre. The training time was 100 seconds for all simulations. This short learning time was enough to obtain a very good approximation of the optimal motion behaviour for the RL algorithm. In Fig 4 the wall following behaviour of the vehicle is shown. In particular, we can see that when the vehicle reaches a corner it cannot turn around without hitting the wall. Instead, it has to go back in order to correct the orientation. This simple mechanism allows the vehicle to follow walls with corners and avoid obstacles.

VII. EXPERIMENTATION RESULTS

In this section we will present the experiments carried out on the real vehicle described in section II. The first step was to train the vehicle along a wall to acquire the optimal behaviour. The training time was 5 minutes. In order to reduce the computation time the steering actions have been limited to three $(-20°, 0°, 20°)$.

To test the performance of the proposed method on a real autonomous vehicle, we have considered moving the vehicle in a closed corridor. When the vehicle reaches the obstacle, which closes the corridor, it has to turn back in a very narrow space. To succeed, it uses two different controllers, one to go forward and the other to go backward. Fig. 5 shows some images of the vehicle trajectory. The former task is quite difficult, since the steering range of the vehicle is extremely narrow $(-20° \leq \varsigma \leq 20°)$. The vehicle can also turn back in the corridor without any physical wall, as shown in Fig. 6, where the corridor is closed by a virtual wall.

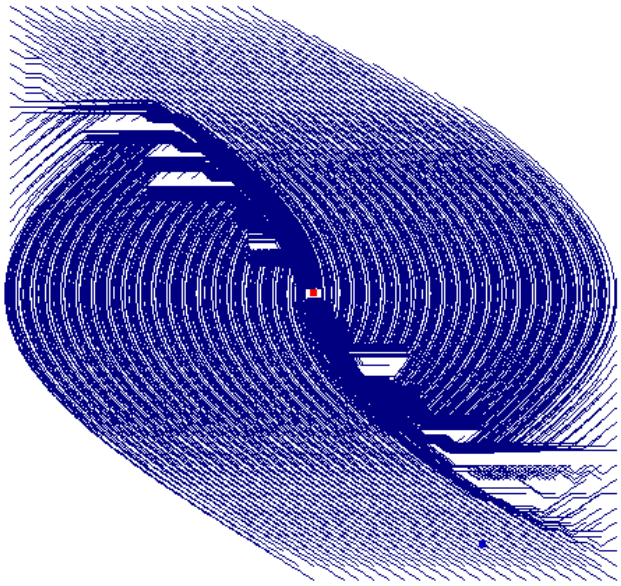

Fig. 3. State space (d, β) showing the controllable trajectories.

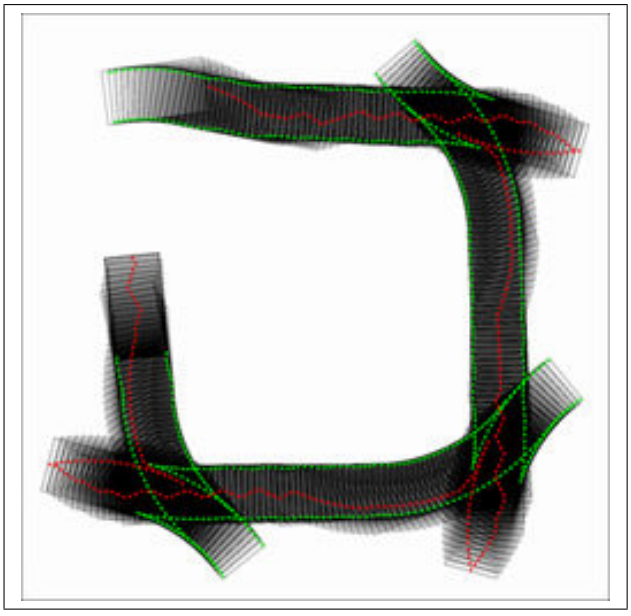

Fig. 4. Wall following behaviour. The vehicle goes from the starting position (top-left) to the final position turning clockwise.

VIII. CONCLUSION

The new RL approach has been successfully employed for navigation of nonholonomic vehicles. The vehicle learns in a few minutes a wall-following behaviour to explore the environment. In contrast with conventional RL techniques (Q-learning, Dyna-Q, Prioritized Sweeping), the algorithm does not need to use function interpolation to find a close to optimal behaviour in continuous state spaces. Furthermore, this approach provides a closed-loop solution in the presence of obstacles including forward and backward motion. The result provides a good approximation of the optimal motion

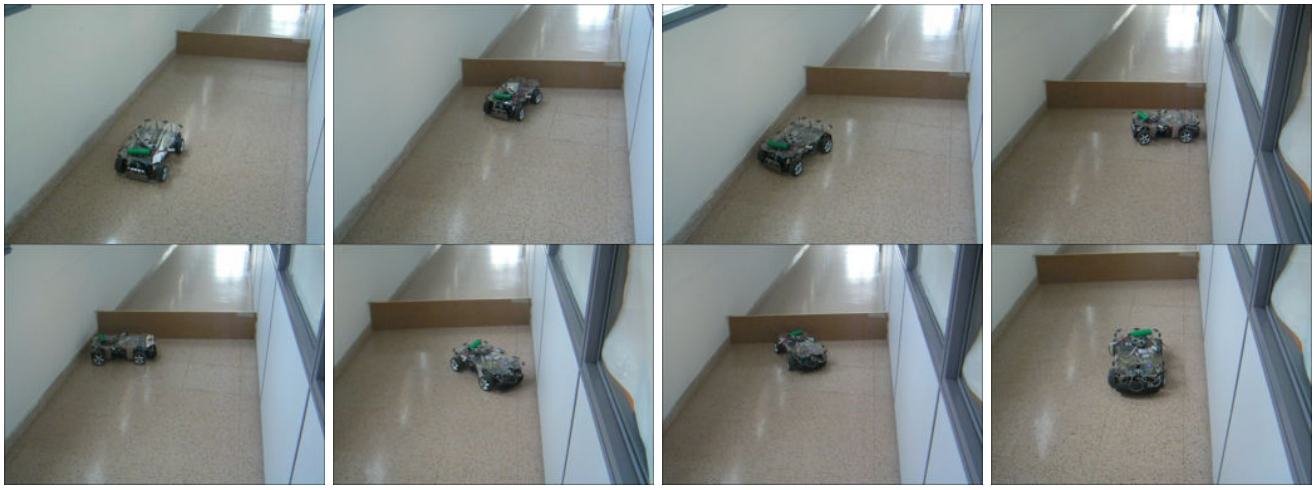

Fig. 5. Sequence of images showing the wall following behaviour learned on the real vehicle.

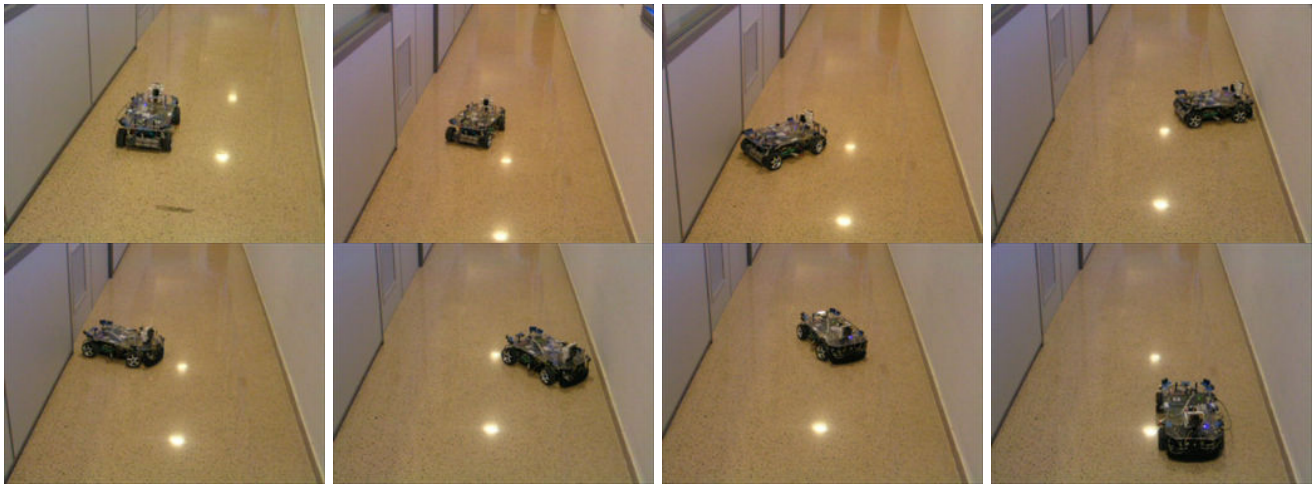

Fig. 6. Sequence of images showing the wall following behaviour with a virtual wall closing the corridor.

through the adequate selection of the adjoining distance.

This controller proposed is suitable to explore unknown environments creating a vehicle model and a map of the environment that can use a three-dimensional RL controller. Thus, the path planning can be improved using a control architecture with two levels of abstraction.

REFERENCES

[1] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic, 1991.

[2] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion*. MIT Press, 2005.

[3] J. A. Reeds and R. A. Shepp, "Optimal path for a car that goes both forward and backward," *Pacific J. Math*, vol. 145, no. 2, pp. 367–393, 1990.

[4] D. Gu and H. Hu, "Neural predictive control for a car-like mobile robot," *Int. J. of Robot. and Autonomous Systems*, vol. 39, no. 2-3, pp. 73–86, 2002.

[5] F. Lamiraux and J. P. Laumond, "Smooth motion planning for car-like vehicles," *IEEE Trans. Robot. Automat.*, vol. 17, no. 4, pp. 498–502, 2001.

[6] T. Fraichard and J. M. Ahuactzin, "Smooth path planning for cars," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, Seoul, 2001, pp. 21–26.

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

[8] ——, "On nonholonomic mobile robots and optimal maneuvering," *Revue d'Inteligence Artificielle*, vol. 3, no. 2, pp. 77–103, 1989.

[9] P. Zufiria and T. Martínez-Marín, "Improved optimal control methods based upon the adjoining cell mapping technique," *Journal of Optimization Theory and Applications*, vol. 118, no. 3, pp. 657–680, 2003.

[10] T. Martinez-Marin and T. Duckett, "Fast reinforcement learning for vision-guided mobile robots," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, Barcelona, 2005.

[11] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[12] A. Moore and C. Atkeson, "Priortized sweeping: Reinforcement learning with less data and less time," *Machine Learning*, vol. 13, pp. 103–130, 1993.

[13] T. Martinez-Marin, "On-line optimal motion planning for nonholonomic mobile robots," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, Orlando, 2006, pp. 512–517.

[14] C. Hsu, "A discrete method of optimal control based upon the cell state space concept," *Journal of Optimization Theory and Applications*, vol. 46, no. 4, 1985.

[15] P. Zufiria and R. Guttalu, "The adjoining cell mapping and its recursive unraveling, part i: Description of adaptive and recursive algorithms," *Nonlinear Dynamics*, vol. 4, pp. 204–226, 1993.