Robotica (2012) volume 30, pp. 159–170. © Cambridge University Press 2011
doi:10.1017/S0263574711000452

Optimal motion planning by reinforcement learning in autonomous mobile vehicles

M. Gómez†*, R. V. González†, T. Martínez-Marín‡, D. Meziat† and S. Sánchez†

†*Departamento de Automática, Escuela Politécnica Superior, Universidad de Alcalá, Campus Universitario, 28871 Alcalá de Henares, Madrid, Spain*
‡*Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante, 03080 Alicante, Spain*

(Received in Final Form: April 4, 2011; accepted April 13, 2011. First published online: May 19, 2011)

SUMMARY
The aim of this work has been the implementation and testing in real conditions of a new algorithm based on the cell-mapping techniques and reinforcement learning methods to obtain the optimal motion planning of a vehicle considering kinematics, dynamics and obstacle constraints. The algorithm is an extension of the control adjoining cell mapping technique for learning the dynamics of the vehicle instead of using its analytical state equations. It uses a transformation of cell-to-cell mapping in order to reduce the time spent during the learning stage. Real experimental results are reported to show the satisfactory performance of the algorithm.

KEYWORDS: Adjoining; Cell-mapping; Non-holonomic systems; Optimal motion planning; Reinforcement learning.

1. Introduction

Path planning and optimal control algorithms for solving specific practical problems, such as trajectory generation for Wheeled Mobile Vehicles (WMVs) and, specifically, Car-Like Vehicles (CLVs), have been addressed in several research projects. The CLVs are non-linear dynamic systems whose motion laws have been widely studied.[1–4] These vehicles are widely used in the industry because they can carry heavy objects with only one power motor. However, the motion planning and control are difficult tasks since they are non-holonomic systems.

The problem of motion planning can be stated as follows:[3] 'Given an initial configuration and a desired final configuration of the robot, find a path, starting at the initial configuration and terminating at the final configuration, while avoiding collisions with obstacles'. Furthermore, if a cost function, for example time or energy, is minimised, the optimal motion problem is addressed. Besides solving the basic motion planning problem, intelligent vehicles must exhibit an optimal behaviour in real scenarios. This supposes an additional aim in the design of efficient algorithms for motion planning in autonomous vehicles with restricted computational resources.

* Corresponding author. E-mail: mgomez@aut.uah.es

Global optimal motion planning of CLV remains today an open problem, especially in complex tasks, such as automatic car parking. In general, research has been focused on different methods based on trajectory generation algorithms. All those methods have a common objective: to obtain a sequence of recommended real movements avoiding the obstacles present in the environment through which the vehicle is travelling.[5] The implementation of these algorithms is based on a method known as offline mode, which consists of finding a collision-free path from previous information (obstacle position, characteristics and geometrical shapes of the space where the vehicle can move, etc.) about the environment. If we want to achieve a specific trajectory, which meets a particular criterion, for example minimum time, minimum energy and shortest path, it means that the vehicle should generate an intelligent trajectory. In this context, ref. [6] describes the implementation of an A* search algorithm used for the path planning in autonomous mobile vehicles using a map of all known obstacles to determine the shortest collision-free trajectory. Therefore, the goal of these algorithms is that the CLV follows the path obtained during the offline mode calculation. Obviously, this control depends on the dynamic and mechanical characteristics of the vehicle, which is being controlled. In ref. [7], both the problem of collision detection and path planning in dynamic environments are dealt. The robot's path is controlled using kinematic-based navigation laws that depend on navigation parameters, which are tuned in real time to adjust the path of the robot.

Typical WMVs have fixed rear/front driving wheels and front steering wheels. In these kinds of vehicles, different optimal control algorithms are used to perform an intelligent trajectory generation. On the basis of Steering Angle Velocity Control Model (SAVCM), ref. [8] deals with the trajectory-planning problem of CLV with fixed rear driving wheels and front steering wheels, whose steering angle is not instantaneously and effectively realisable. The work presented by ref. [9] describes a trajectory planner composed of a trajectory generator and a control strategy planner in open loop for a high-speed WMV moving through a desired path. In this work, they use a bang–bang control strategy based on the kinematics constraint in order to obtain shortest distance solution to the optimal trajectory generation. In ref. [10], the method described is

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

also an open-loop solution that determines the time-optimal trajectories between two arbitrarily specified configurations for a non-holonomic bidirectional steered robot in the plane without obstacles. However, there are solutions[11] that are focused on the problem of stabilising reference trajectories for non-holonomic mobile robots using kinematic control models.

Other approaches are based on algorithms that generate smooth motion.[5,12] When there are no obstacles, some algorithms search the shortest path.[2] The main problem of these algorithms is that the trajectories are computed in open loop, whereas in real environments where a vehicle is subject to perturbations and uncertainty, closed-loop action is more desirable. This kind of solution could be achieved using Dynamic Programming (DP),[4,13] but although DP is efficient compared to direct search, it requires a high computational resources and a model of the motion law of the vehicle.

However, there are other techniques that do not need a model for achieving a closed-loop solution. These techniques are based on reinforcement learning where the optimal motion is estimated through an interaction between the vehicle and its environment. In ref. [14], a system that classifies the video information and the vehicle behaviour into patterns is established. They use a fast pattern-matching algorithm to decide on the required interactions with the environment, for example issuance of steering commands, in order to autonomously steer a vehicle. These techniques implement optimal control policies based on DP, where the dynamic model of the vehicle is built on line, while the algorithm is executed and the controller is generated. Other authors, such as Brunskill *et al.*,[15] propose CORL, an algorithm for efficiently learning to act in continuous-state environments. They demonstrate that their dynamics representation enables them to capture real-world dynamics in a sufficiently accurate manner to produce good performance. However, their solution depends heavily on the sample period, as any solution based on pure reinforcement learning. Also, CORL achieves near optimality stating in a precise manner when and with what probability it will be achieved.

Due to the fact that optimal control problems are generally very difficult to solve both analytically and numerically, the cell-mapping technique[1,16–22] offers an effective numerical approach to obtain global and local solutions of the non-linear optimal control problem.

Other works, related to the motion planning, look for specific algorithms (parking algorithms), but not from a global point of view. These works do not take into account any optimal criteria and they do not learn the environment from experience but they just perform a specific control. The proposed algorithm in this work can also park the vehicle considering the parking place as the goal that the vehicle has to reach in minimum time from any origin position. For example, in ref. [23], an arc path parking algorithm is proposed. The parking path is a circular arc connecting the car position to the goal point, and the tangent angle of the arc at the goal point equals to the desired car yaw angle. In ref. [24], a target parking position designation method based on parking slot marking recognition is proposed. The driver designates two seed points that are the end point of two line segments separating the target parking slot from adjacent parking slots. It can be applied to various parking slot marking types.

However, Fu *et al.*[25] and Thrun *et al.*[26] present route planning for vehicles based on given information of the urban road network. This planning searches for optimal routes to direct the vehicle through a series of checkpoints. Therefore, these works use external data from which the planning is performed.

Usually, most of the systems in which we need to apply an optimal control are continuous. Due to this fact, it is necessary to consider a discretisation of the state space when finding optimal controllers. Discretisation techniques[18,21,27] using finite element or finite difference methods applied to uniform grids are widely used and provide convergence results. However, such uniform discretisation suffers from impractical computational requirements when the size of the discretisation step is small or when the state space is of high dimension. In order to solve this kind of problems, some works use variable resolution discretisation to approximate the value function and the optimal control and compare experimentally several splitting criteria.[28]

Taking into account the advantages associated with the DP, reinforcement learning methods and cell-mapping techniques, a new optimal motion planning algorithm is proposed in this project. It is tested in a real scenario with a mobile platform. This algorithm is very robust to any change involved in the vehicle dynamics because the vehicle model is estimated in real time from received experience. The goal of this algorithm is to find an optimal controller from the previous experience in order to obtain the minimum time distance solution to the optimal trajectory generation. The results presented in ref. [21] only take into account the dynamics of the system under a simulation environment. However, the kinematics together with dynamics applying a transformation of cell-to-cell mapping in order to reduce the time spent in the learning stage have been incorporated in the proposed algorithm. In refs. [22] and [29], the algorithm is tested under simulation conditions. In ref. [30], the algorithm is tested under semi-real conditions because it was not embedded in the vehicle but it was running in a computer, which kept a Bluetooth link with the vehicle in order to send both its localisation and the new control action. Due to good behaviour under this scenario, the next step is to give the vehicle full autonomy for the execution of the algorithm and to test it taking into account different goal states and therefore different learning stages.

2. Optimal Control Techniques

Optimal control techniques could be split into following two classes: (1) those based on a dynamic model of the CLV and its environment, and (2) those where the model of the system is not available. One of the approaches related to the first class and taken into account in this work are the cell-mapping techniques applied to the control of non-linear systems.[18] These techniques need to have a mathematical model of the system in order to perform an optimal control law. However, the second class is referred to techniques that learn from received experience, and therefore, once the knowledge

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

of the system has been achieved, an optimal control law can be performed. These last techniques would be widely applicable if the methods were computationally feasible and could be robustly applied to non-linear systems. They are named reinforcement learning methods.[31,32] There are two different types of these methods: indirect and direct. An indirect method relies on a system identification procedure for creating an explicit model of the controlled system and thus being able to determine the control rule of that model. Direct methods can determine the control rule without knowing the model of the system.

The new algorithm proposed for solving the optimal control problem in CLV is based on a particular case of a cell-mapping technique applied to the control of non-linear systems, combined with another technique based on the direct method of reinforcement learning.

2.1. Cell-mapping techniques

The cell-mapping techniques include, on the one hand, an efficient application of the numerical methods in order to integrate non-linear systems and, on the other, the Bellman's Principle of Optimality to find the optimal control efficiently. The result is a new optimal control method based on a cell state-space able to design efficiently optimal controllers of highly non-linear systems. The design method of controllers based on the cell-mapping techniques[18,20] can be divided into two stages:

- Obtain a family of cell-to-cell mappings that constitutes the necessary knowledge in order to calculate the optimal control laws associated with the states of the system.
- Search the optimal control laws taking into account the Principle of Optimality (intelligence searching techniques). The process finishes when an Optimal Control Table (OCT) is found, which acts as a controller.

Cell-to-cell mapping methods are based on a discretisation of the state variables of the system, defining a partition of the state space into cells.[1] A cell-to-cell mapping can be derived from the dynamic evolution of the system. In ref. [16], a cell-mapping application for the design of optimal controllers is proposed. This method (CSCM), based on the Simple Cell Mapping (SCM) technique, carries out a discretisation of both state and control variables and uses a cost function to specify the desired optimality criterion. In refs. [18] and [20], the Control Adjoining Cell Mapping (CACM) algorithm for optimal control of highly non-linear 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.[33]

Let us consider a system under the action of a P-dimensional control vector, which can only take some finite values Nu, belonging to a set U. It is assumed that the control is maintained constant during a time interval, t. When an action is applied to the system during a time interval, the system can go to a new state (image cell), remain at the same state or go to the drain (out of the state space). The knowledge of the system is given by the set of transitions to the image states. The maximum size of the set of transitions is $Nc \times Nu$, where Nc is the total number of cells or states.

The adjoining property states that the distance (D-k) between any cell and its map (image) is equal to some predefined integer value $k \geq 1$. The distance between two cells z and z' is defined as

$$\text{D-k}(z, z') = \max_j |z_j - z'_j| = k.$$

The integration time of each transition is determined adaptively to carry out the adjoining property. Such a property provides substantial improvements with respect to other optimal control techniques, since the CACM algorithm only computes the meaningful information required to obtain a good approximation to the optimal solution. Sometimes it may happen that the transition does never map into a distant-k cell, for example when it is trapped in the origin cell or it goes to a drain cell. In these cases, the algorithm stops and it changes the control action in order to go out from the cell or come into the state space.

Bellman's Principle of Optimality states a strategy to determine the global optimal solution for dynamic systems. The solution can be obtained through the well-known Bellman equation, or by addressing the problem as a shortest path search, since shortest path problems satisfy the Principle of Optimality. Both methods find the optimal policy by processing the state space backwards in time, starting at the objective state. The CACM technique carries out the shortest path search in an efficient manner, reducing both memory requirements and computation time. In continuous state-spaces, the CACM method is an attractive alternative to conventional DP since it states an efficient method for managing the approximation to the optimal behaviour without incurring a high computational cost.

When building the OCT, a cost function is defined to indicate the cost for a control action to map a cell to its image. The cost can be defined in terms of time, energy, distance or any other factor. Since a cost function is specific for a cell, it can be used as a local performance measure for controller evaluation. A global performance measure for a controller can be generated from the local measures by simply averaging all the costs on all controllable cells.

2.2. Reinforcement learning

Reinforcement learning methods only require a scalar reward (or punishment) to learn to map situations (states) in actions.[32] 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, and applied policy. The objective is to find the actions (policies) that maximise 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 optimisation problems. It was introduced by Watkins[34] and it is also described in Sutton's book[32] as well as in Kaebling's work.[35] The convergence of this algorithm towards the optimal policy p^* was proven by Watkins.[36] The accumulated reward for each pair state-action

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

$Q(s, a)$ is updated (back-up) by the one-step equation

$$\Delta Q(s_t, a_t) = \alpha(r_{t+1} + \gamma Q_{\max}(s_{t+1}, a) - Q(s_t, a_t)), \quad (1)$$

where Q is the expected value of performing action a in state s, r is the reward, α is the learning rate that controls convergence and γ is the discount factor. The discount factor makes rewards earned earlier more valuable than those received later. If the reward function is proper,[37] the discount factor can be omitted ($\gamma = 1$). The action a, with highest Q value at states, is the best policy up to instant t.

Due to the fact that our goal is to perform an optimal motion planning in minimum time, the cost function that we have to minimise is the time. The purpose is to find a control action that minimises at each time step the accumulated time from any state to the goal. The accumulated time is directly related to Q value. Q values are estimated on the basis of experience: as the robot is moving around the environment, the accumulated time (Q value) in the local state (origin) is added to the one associated with the new state. While the robot is moving, the propagation will go extending to other states. At the end of the learning process, the Q value will have been propagated to all controllable states. Therefore, the propagation is backwards, both in time and space. The algorithm solves an optimal control problem using DP.[38]

In applications dealing with real systems, Q-learning spends 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. Prioritised sweeping[39] and Dyna-Q[40] include a search control to simulate the past real experiences in a specified order. The idea is basically that for each time step, the look-up table is updated using the received experience, just like normal one-step Q-learning, and in addition N updates of Q are made using simulated experience. The N updates are random updates of previously executed state-action pairs, where the model has some recorded experience.

2.3. CACM with Reinforcement Learning

Q-learning was conceived for discrete state and action spaces. In robot motion planning applications, the state space is continuous (infinitum states), so it is mandatory to discretise the state space in order to use Q-learning as it was conceived. The state space is divided in cells, each cell containing infinitum states that will be represented as a single state by the algorithm. The inherent discretisation errors can produce a poor approximation to the optimal behaviour in complex non-linear systems, such as CLVs.

The CACM algorithm incorporates the adjoining distance mechanism to deal with the discretisation problem. In Q-learning, the transitions between states are evaluated at fixed sample times, while in CACM the transitions have to satisfy the adjoining distance condition in order to be evaluated. There are only two cases in which the adjoining distance condition may not be met according to CACM algorithm. The first is when the transition is trapped in a cell. In this case, another control action is applied in order to provoke a new transition and go out from the cell. The second is when the transition goes to the drain cell. In this case, the CACM algorithm applies a reactive function, which takes it out and continues running normally (see lines 4 and 5 of the Algorithm 1). Selecting appropriately this distance with respect to the number of cells of the system, it is possible to minimise the effect of the grid resolution and the relative length of the transitions over the optimal behaviour of the system.

The new algorithm, CACM-RL, is represented in the so-called Algorithm 1. It has been implemented as a model-based reinforcement learning method, where the model is generated from experimentation. In general, model-based reinforcement learning finds better trajectories and manages changes in the environment, for example new obstacles. Also, the goals are reached more efficiently than in direct reinforcement learning.[32]

Algorithm 1 CACM-RL

```
1:  Initialise Q-Table(s,a) y Model_Table
2:  x ← current state
3:  s ← cell(x)
4:  IF      s ∈ drain or s ∈ goal or s ∈ safety_area
5:  THEN    F_reactive(x)
6:  ELSE    IF      D-k-adjoining(x,x')
7:          THEN    Q-Table(s,a)← s',r
8:                  Model_Table ← IT(x,x')
9:          a ← policy(s)
10:         Execute action a on the vehicle
11:         Observe the new state x' and r
12: UNTIL the end of the learning stage
13: FOR all (s,a), repeat N times
14:   x̄' ← Model_Table, DT(x,x')
15:   s̄' ← cell(x̄')
16:   Q-Table(s,a) ← s̄', r
```

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

- *Q-Table(s, a)*: It is the Q-factor table, where the accumulated reward for the (s, a)-pair is saved. From this table, the optimal policy a^* is obtained as follows:

$$a^*(s) = \arg .\max_a Q - \text{Table}(s, a)$$

  The *Q-Table(s, a)* is updated according to Eq. (1), where the learning rate, α, has been set to 0.4 and the discount factor, γ, to 1.

Although α is set to a constant value, the algorithm has the ability to adjust it automatically by means of an average reward of each of the visited states. The discount factor makes rewards earned earlier more valuable than those received later. If the reward function is proper,[37] as in our case where we have a minimum time problem, the discount factor can be omitted ($\gamma = 1$):

- *Model-Table(s, a)*: It is the local vehicle model. It contains an average of local transitions for each velocity (as many as cells) and for each control action. The transitions have

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

to satisfy the D-k adjoining property to assure a good approximation to the optimal policy.
- *policy(s)*: It selects a specific policy to estimate the model (*Model-Table*) and to exploit the best policy acquired.

During the learning stage of Algorithm 1, only exploration is done. Exploration can be defined as the used strategy to select control actions, which the agent believes have the best long-term effect. A successful agent obtains rewards by performing actions, it has tried in the past and found to be effective in producing rewards. But to discover such actions, it has to try actions that it has not selected before. The exploration is the key to learning the optimal action policy by means of the determination of the next control action. In order to get a good learning, it is necessary that all control actions for each cell (state) are applied during the exploration. The more times the agent chooses a control action, the better learning will be. There are several strategies used for exploration. In our case, we use a random selection due to the fact that it gives us a good performance; all actions are chosen enough number of times for achieving the transition knowledge from the model.

Exploitation is the technique by means of which the selected control actions during learning are the best long-term. This means that the selection is done using a criterion that let us chose the best action in that moment. Therefore, the exploitation is the key to optimum performance. Moreover, in some cases exploration is combined with exploitation (i.e. e-greedy or dyna-Q) in order to speed up the learning process. The speed of the learning process is accomplished by choosing that control action that achieves the greatest long-term benefit.

Due to the fact of applying a cell-to-cell mapping transformation, the learning is so fast that is not worth exploitation. A random control action is chosen for the exploration when a transition is provoked. In this way, convergence is guaranteed because all control actions have been selected several times[32]:

- *IT(x,x')*: It is the operator that transforms a global transition *x-x'* into a local transition. When all possible local transitions have been performed, we can conclude that the learning stage has finished because using the operator *DT(x,x')* we can transform local transitions into global and in the same way, we are able to have the knowledge of the whole state space for each control action being applied. In order to perform a real approximation in the transformation to local transitions, the real value for each variable is averaged out and filtered. The transformation of global transitions in X_2, X_3 and X_4 state variables consists of translations and rotations to a local point. However, the local value of the velocity is estimated.
- *f_reactive(x)*: There are some real situations where the vehicle could go off the state space (drain state), reach the goal or reach the safety area associated with any obstacle. In these cases, it is necessary to implement a reactive function in order to continue the learning, either coming into the state space or moving the vehicle to another position.
- D-k-*adjoining(x,x')*: According to CACM technique, we detect the transition distance between the current state x and the previous one x', by means of this function. The results presented in this work have been achieved using a distance equal 1.
- *cell(x)*: It transforms a continuous state x into a discrete value s (cell).

Related to this combination of techniques, the contributions carried out in our work are focused on applying them in a CLV. According to CACM-RL algorithm, each EPISODE is equivalent to one training session. In each training session, the agent explores the environment, gets the accumulated time (reward) until it reaches the goal state or goes to the drain state. The purpose of the training is to enhance the 'brain' of the controller that is represented by *Q-table*. The more the learning stage extends, the more the Model-Table approximates its real values.

The controller that we generate using Algorithm 1 is non-linear. We are able to perform an optimal control of a non-linear system (WMV) without linearising the model of the vehicle. One of the strong points of cell-mapping techniques (and therefore in the new algorithm) is to be able to perform optimal control, both in linear and non-linear systems using the same concepts and strategies.

2.3.1. Convergence and controllability of the Algorithm 1. In optimal control problems, control actions are usually restricted to be between a lower and an upper bound. In general, if the optimal control switches from one extreme to the other at certain times (sometimes it could switch in between the bounds), it is referred to as a bang–bang control. In our case, we have a bang–bang problem because we want to obtain a pathway in minimum time. The problem described in our work considers two control actions and up to four state variables. In this case, we can also apply the bang–bang theory. The bang–bang solution has an associated switching curve, so if we graphically represent the steering action of our problem on the X–Y plane, we obtain the shape of an S-shaped curve. In general, this curve is called 'S-shaped switching curve'.Therefore, the performance of the proposed algorithm can be evaluated from this curve between maximal control values.

The S-shaped switching curve is typical in bang–bang control algorithms. Figure 1 shows this curve, which has been generated considering the goal located at the coordinate origin and with velocity and orientation equal to zero. As we can see in Fig. 1, the 'S-shaped switching curve' separates the XY-plane into several subsets (red, yellow, blue and cyan), where the steering action assumes either a right turning or a left one. Figure 1 shows five trajectories that the vehicle generates in order to reach the goal in minimum time. Independently of where the robot is located, the trajectories shall always reach the switching curve. Once it is reached, the robot will follow it. In practice, we can see that our solution approximates to the optimal controller.

The trajectories in the first quadrant starts moving back (both start with an orientation equal to zero), turning to the right (pink) until the robot reaches the switching curve (due to the discretisation effect, our algorithm approximates the switching curve to the border between the cyan and blue areas), where it begins to turn to the left (black) following the border. With this last movement, it reaches the goal.

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

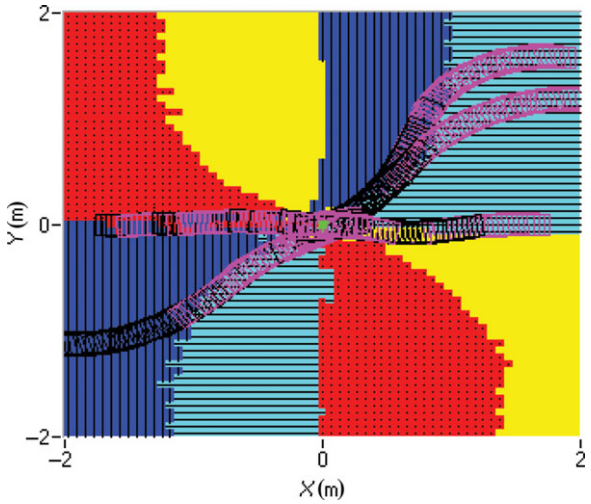


Fig. 1. (Colour online) S-shaped switching curve. This Figure shows the S-shaped switching curve that has been generated with a constant velocity and orientation (0°) in the *XY*-plane, and the goal has been placed in the origin of the state space (0, 0, 0, 0).

The trajectory in the third quadrant starts in a position located in the blue area and it goes ahead (the robot starts with an orientation equal to 180°), turning to the left (black) until it reaches the border for turning to the left (pink) in order to reach the goal.

At last, there are two trajectories that start at the border between quadrants 1–4 and 2–3, one with an orientation equal to 180° (quadrant 1–4) and the other one equal to 0° (quadrant 2–3). Both trajectories go straight reaching the goal. During their way and due to the fact that they are at a border, the trajectories cancel the effects of steering produced by the border.

The quality of the controller implemented by Algorithm 1 is measured from its controllability. The controllability concept that we use in our work is different from the one used in classical control theory. A cell is controllable if starting at the centre point of such a cell (initial state), the system evolves reaching some point inside the objective cell (final state). Thus, the percentage of controllable cells is the number of controllable cells with respect to the number of cells inside the region of interest that do not belong to obstacles. A highly optimised controller should be able to control the largest possible region of a cell space. Since different cell resolutions result in a different number of cells for a cell space, the total number of controllable cells depends on the cell resolution. Controllable cell space percentage is a good measure of controllability and is independent of cell resolution when the cell state space is uniformly discretised.

Thus, the method for evaluating the controllability consists of verifying for each state if the goal is reached. Taking into account this consideration, the temporal evolution of the controllability is represented in Fig. 2. The controllability is measured in different times during the execution of the proposed algorithm.

The solid line represents the evolution of the controllability when no obstacles are considered. The maximum controllability is reached in about 5.5 min, achieving a controllability value of 99%. On the other hand, the dashed

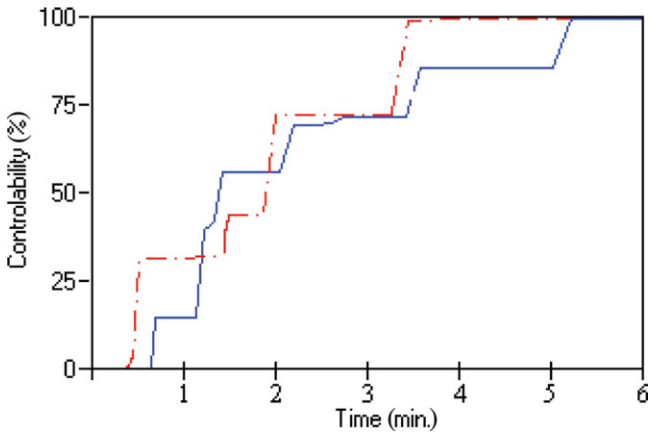


Fig. 2. (Colour online) Temporal evolution of the controllability. The controllability grows according to the progress of the learning time.

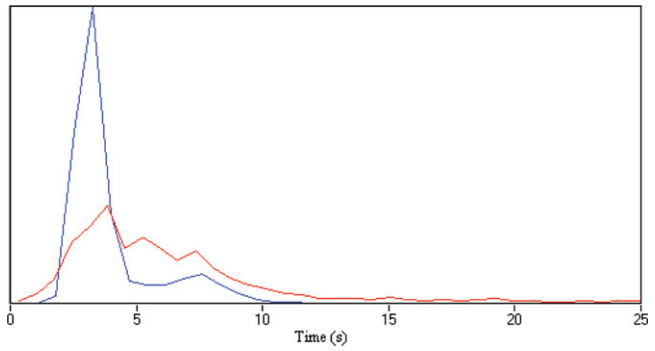


Fig. 3. (Colour online) Histograms showing the average optimal time to reach a goal. The red line corresponds to an adjoining condition equal to 1 and the blue line equal to 3.

line shows how the controllability with obstacles grows depending on the time. In this case, the percentage of the controllable cells is similar to the previous because the total number of cells has been reduced due to the fact that obstacle cells have not been considered. This means that the learning time is lower and the maximum controllability is achieved in about 4 min.

2.3.2. Average optimal time. The optimal time associated with the controller could be evaluated by calculating for each cell, the time required to reach the goal. If we average these times, we get the average optimal time of the system. Regardless of the characteristics of the system, if we use CACM-RL, we can obtain a histogram whose shape and trend will be similar in any case. For example, Fig. 3 shows two histograms, where the time taken by each of the states to reach the goal is represented. By means of this graphic, the optimality of the time can be globally evaluated.

One of the histograms represented in Fig. 3 is performed using an adjoining condition equal to 1 (red line), and the other one with an adjoining condition equal to 3 (blue line). We can see that the larger the adjoining condition, the optimal times are concentrated in a smaller time interval. Also, it can be stated that the average optimal time is lower, the greater adjoining condition, and therefore, it tends to the optimal value. Thus, the average optimal time indicates the approximation to the optimal behaviour of the system.

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

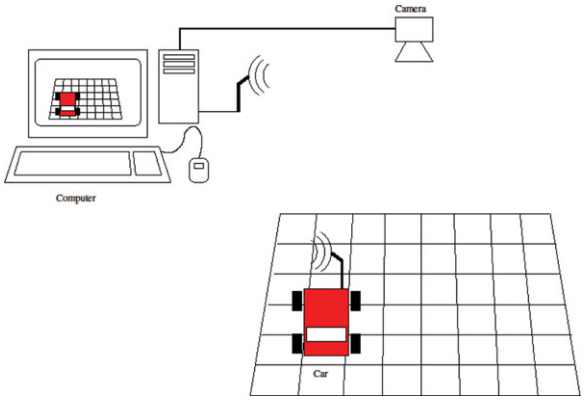


Fig. 4. (Colour online) Real scenario used in the tests.



Fig. 5. (Colour online) 1/8 scale vehicle for testing the proposal new optimal control technique.

3. Working Environment

The scenario used in the tests carried out in order to verify the capacity and behaviour of the algorithm is shown in Fig. 4. We have to take into account that the algorithm runs on a non-holonomic 1/8 scale vehicle (Fig. 5). The scenario has been set up in order to have the localisation of the vehicle (position and orientation) centralised in a Personal Computer (PC), which is continuously receiving the localisation from the camera. When the algorithm needs the state of the vehicle (velocity and localisation), it requests the localisation to the PC by means of a Bluetooth link, and it reads the velocity from the encoder coupled to the traction motor.

The control software developed has been designed in compliance with all constraints and memory limits of the hardware, which is based on the LEON3 processor. The platform integrates a time processor unit in order to perform timing control actions independently from the rest of operations and several sensorial systems for monitoring the value of the used state variables.

It is assumed that the concerned vehicle is moving on a horizontal plane. The two rear wheels are driven via a single motor through a differential device to get the same driving velocity. The two front steering wheels are controlled by a single axis to obtain the same steering. The body of the CLV is perfectly rigid and the wheels are rigid too with the same spoke. It is supposed that the wheels are uniform discs with a negligible thickness. The state variables that have been used in this work are included in the following vector: $X = \{X_1, X_2, X_3, X_4\}$, specified in Table I.

Table I. State variables for real tests.

Symbol	State variable
$X_1 = v$	Velocity
$X_2 = x$	X Cartesian coordinate
$X_3 = y$	Y Cartesian coordinate
$X_4 = \theta$	Orientation

Concept tests have been performed taking into account that the vehicle can move in a limited area. This area is in the field of vision of a CMOS camera in charge of obtaining the localisation and orientation of the vehicle in real time.

The velocity X_1 is obtained from the position given by the encoder coupled to the traction motor. The rest of variables (X_2, X_3, X_4) are acquired in real time by the camera. Thanks to the fact that the software embedded in the camera can track different colours, two different colour cards have been located on the top of the vehicle in order to capture its position (X–Y). The orientation (θ) is performed from the centre of each colour card.

3.1. Implementation of the new algorithm on board the vehicle

The software application in charge of the implementation of the new algorithm has been developed under a real-time operating system, RTEMS 4.8, running on LEON3 processor according to the 32-bit SPARC-V8 architecture. The application consists of two tasks: *request_position* and *position_ready*. Both tasks are launched from two different *Interrupt Service Routine (ISR)*: ISR_A and ISR_B associated with two external interrupts. The ISR_A is performed periodically and activated in each sample period. It is in charge of the following:

- Steering servo control.
- Traction motor control.
- Reading of the velocity of the vehicle.
- Launch the *request_position* task.

The goal of the *request_position* task is to ask the PC shown in Fig. 4 for the localisation of the vehicle. On the other hand, the *position_ready* task is sporadically activated by the ISR_B when the localisation data arrive at the vehicle or any command from the PC is received. This task also implements both the learning stage and the exploitation of the controller in order to generate motion planning. These two actions are coded by means of two commands sent from the PC by the operator. The operator can introduce a number of iterations in which the learning must be completed through the Man Machine Interface (MMI) of the PC. When it is completed, *Q-table* is updated.

Although an interaction between the operator and the embedded software on the vehicle has been necessary during the development of the tests shown in this work, the embedded software design is ready to be completely autonomous in order to be applied to a real mobile platform.

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

Table II. Value range and cells for each of state variables.

State variables	Cells	Range
X_1	3	$-1.5 \leq X_1 \leq 1.5$ m/s
X_2	9	$-0.9 \leq X_2 \leq 0.9$ m
X_3	13	$-1.3 \leq X_3 \leq 1.3$ m
X_4	31	$-\pi \leq X_4 \leq \pi$ rad

Table III. Values for each control action.

Control action	Values		
Voltage in traction motor	−18 V	0 V	18 V
Steering angle	−23°	0°	23°

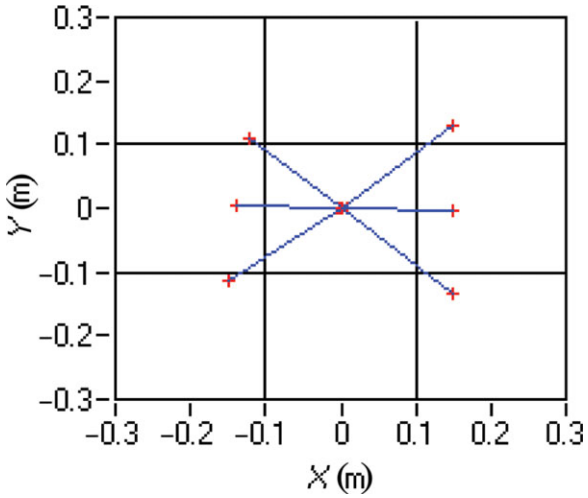


Fig. 6. (Colour online) Example of a model achieved during the learning stage in an experiment.

4. Results

In this section, the results obtained after applying the proposed algorithm are presented. The main achieved goal is to show some concept tests in order to extrapolate the good results to real vehicles. The state space taken into account for the execution of the new algorithm uses the state variables specified in Table I. Both value range and cells associated with each variable are indicated in Table II. The control actions used by the algorithm are specified in Table III.

The sample period and the size of the cells have been performed taking into account the hardware constraints of the camera and response time of the traction motor. As an example of a model generated during the learning stage in an experiment is shown in Fig. 6. The local coordinates from which the model has been created is the origin state $\{X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 0\}$.

Next, different real tests are presented. Two different scenarios have been considered: the first one is shown in several videos[41] (Figs. 7, 8 and 9) and we can see how the vehicle performs a trajectory planning of an autonomous way in a free obstacle environment. The second one represented in Fig. 10 shows three trajectories that the vehicle performs in the same previous environment but adding obstacles. All tests have been performed using CACM-RL algorithm. The error associated with each trajectory is not accumulative because the optimal control is obtained in order to reach the goal in each of the cells.

In Fig. 7(a) and (b), the vehicle starts going into reverse and after that it turns to the left and goes ahead (Fig. 7(c)). In this movement, the car comes out of limited area (Fig. 7(c)). Applying the reactive function of the proposed algorithm, the car starts going into reverse and it comes into the Cartesian space again. In this moment, it starts going straight ahead (Fig. 7(d) and (e)). When it reaches the state of the Fig. 7(e), it starts going into reverse turning to the right (Fig. 7(f)). Then, it goes ahead turning to the left (Fig. 7(g)) approaching the goal. Due to the discretisation and sample period, the car goes into reverse turning to the right in order to start going ahead turning to the left reaching the goal (Fig. 7(h)).

In Fig. 8, due to the negative initial orientation (Fig. 8(a)), the car starts going into reverse turning to the right (Fig. 8(b) and (c)) in order to reach a state in which it has to go ahead turning to the left (Fig. 8(d) and (e)) for achieving the same orientation than in the goal (Fig. 8(f)). Next, it reaches the goal going into reverse straight (Figs. 8(g) and (h)).

In Fig. 9, the vehicle starts going into reverse straight due to the initial orientation equal to 0° (Fig. 9(a) and (b)). After

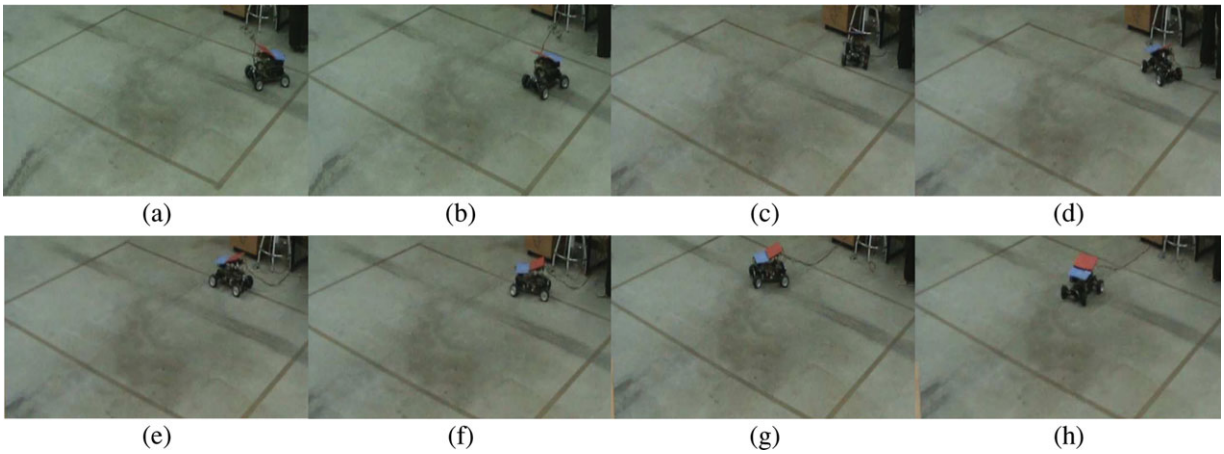


Fig. 7. (Colour online) Test case, where the origin state is equal $\{v = 0,\ x = -0.46\,\text{m},\ y = 1.25\,\text{m},\ \theta = 142.1°\}$ and the goal state $\{v = 0,\ x = 0,\ y = 0,\ \theta = 0\}$.

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

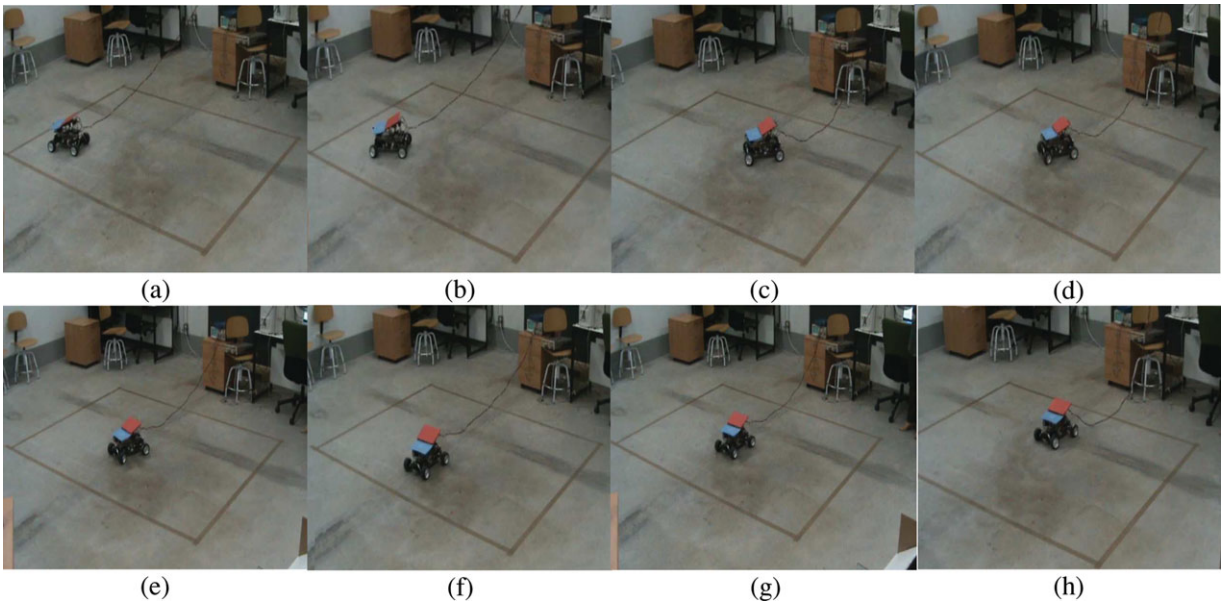


Fig. 8. (Colour online) Test case, where the origin state is equal $\{v = 0,\ x = 0.81\,\text{m},\ y = -0.94\,\text{m},\ \theta = -68.7^{\circ}\}$ and the goal state $\{v = 0,\ x = 0,\ y = 0,\ \theta = 0\}$.

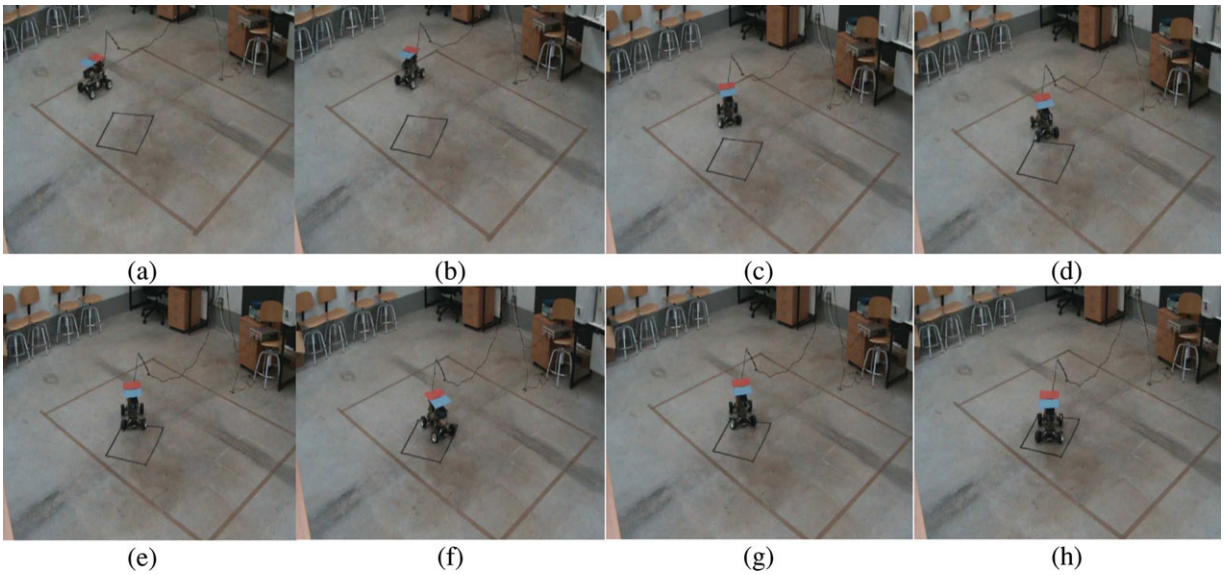


Fig. 9. (Colour online) Test case, where the origin state is equal $\{v = 0,\ x = 0.46\,\text{m},\ y = -1.2\,\text{m},\ \theta = 0^{\circ}\}$ and the goal state $\{v = 0,\ x = 0.86,\ y = -0.05,\ \theta = 24.65^{\circ}\}$.

that it goes ahead turning to the left in order to approach the goal (Fig. 9(c)–(e)). Later, it maneuvers (Fig. 9(f) and (g)) into the goal (Fig. 9(h)).

Figure 10 shows three trajectories in a scenario where several obstacles in the Cartesian space have been added. It can be noted that the vehicle is located at three different initial positions and the goal state (located at the centre of the rectangle) is reached in minimum time dodging the obstacles that it meets in the way. In this case, the security area around each obstacle is marked with a yellow border. Both value range and cells associated with each variable are indicated in Table II. The control actions used by the algorithm are specified in Table III. The best way to check how the vehicle dodges the obstacles is to paint the path that it does. For this, the vehicle sends its state to a remote computer via Bluetooth in order to paint the trajectory in the scenario where the obstacles have been mapped.

4.1. Comparison with other techniques

During the introduction, we review several types of optimal control methods, which could be classified into two groups: online and offline methods. The offline methods used for optimal motion planning consist of finding a collision-free path from previous information (obstacle position, characteristics and geometrical shapes of the space where the vehicle can move, etc.). That means that after learning, when the vehicle has to perform planning, any change involved in the environment would impact directly in planning. Thus, it is likely that the path followed by the vehicle is not optimal. If we focus on the new proposed algorithm, it can be configured

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

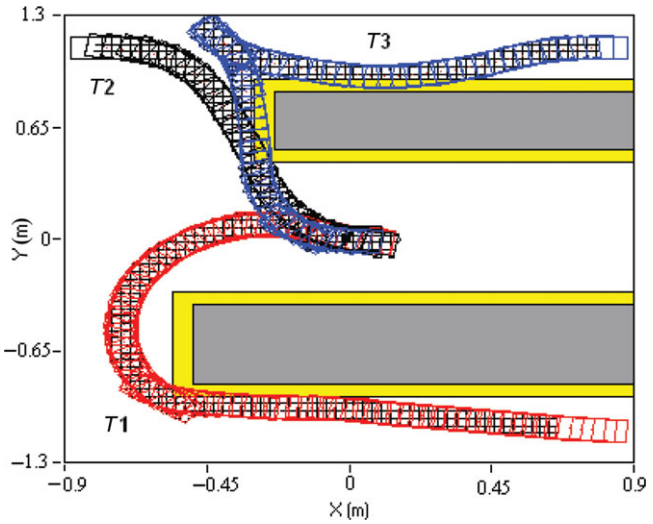


Fig. 10. (Colour online) Test case in a scenario with obstacles, where the vehicle starts from three origin states corresponding to the three trajectories: $\{v = 0,\ x = 0.75\,\text{m},\ y = -1.1\,\text{m},\ \theta = 180^\circ\}$, $\{v = 0,\ x = -0.75\,\text{m},\ y = 1.1\,\text{m},\ \theta = 0^\circ\}$ and $\{v = 0,\ x = 0.75\,\text{m},\ y = 1.1\,\text{m},\ \theta = 0^\circ\}$ in order to reach the same goal state $\{v = 0,\ x = 0,\ y = 0,\ \theta = 0^\circ\}$.

as an online or offline method. In case of working as an online method, it can be doing learning and planning concurrently and, if it works as an offline method, it performs planning from the knowledge acquired during the learning in offline mode. However and due to CACM-RL algorithm works in closed loop, during the planning it can be learning while control actions are acting on the vehicle, but also it can correct the errors arising as a result of any change due to the fact that the error associated with each trajectory is not accumulative: the optimal control is obtained in order to reach the goal in each of the cells.

According to Fig. 2, a short learning time was enough to obtain a very good approximation of the optimal motion behaviour for the CACM-RL algorithm. The high number of controllable cells demonstrates the good performance of the proposed method. However, if we apply Q-learning method alone (reinforcement learning), the optimal motion behaviour strongly depends on the transition time. If the transition time is not adequate, the final behaviour can be very poor. Therefore, the proposed strategy improves the computation time and offer better results than cell mapping and reinforcement learning techniques separately.

Both percentage of controllable cells and the average optimal time (see 'Convergence and controllability' and 'Average optimal time' sections) are two indexes that can provide a robust test of performance of non-linear dynamic systems, instead of testing the performance just for a few initial states. After performing several tests with CACM-RL and Q-learning separately, we have created the Table IV as a summary, which shows a comparison between the new CACM-RL algorithm and Q-learning taking into account the previous indexes of performance and studying the effect of the relative length of transitions over the optimal behaviour of the system. The column D-k/T_s of the Table IV refers to the adjoining condition associated with CACM-RL algorithm (D-k) or the sample period used in Q-learning method (T_s).

Table IV. Results of average optimal time and controllable cells.

Control technique	D-k/T_s	Average optimal time (s)	Controllable cells (%)
CACM-RL	D-1	t_1	cc_1
	D-2	$t_2 \ll t_1$	$cc_2 \gg cc_1$
	D-3	$t_3 < t_2$	$cc_3 \sim cc_2$
Q-learning	T_{s1}	t'_1	cc'_1
	$T_{s2} > T_{s1}$	t'_2	$cc'_2 > cc'_1$
	$T_{os} > T_{s1}$	t'_3	$cc'_0 > cc'_2$
	$T_{s3} > T_{os}$	t'_4	$cc'_3 \ll cc'_0$

Analysing the data value contained in Table IV, we can conclude the following statements: regarding CACM-RL, as adjoining condition increases, so does the controllability ($cc_2 \gg cc_1$). However, there comes a time when it is not worth continuing increasing the adjoining condition due to the fact that controllability scarcely changes ($cc_3 \sim cc_2$), and nevertheless, the computation time increases. The average optimal time decreases as the adjoining condition increases, but like before, there comes a time when that decrease is minimum and the adjoining condition is not worth increasing it.

When working with Q-learning, it makes no sense to talk about adjoining condition but sample period. We can easily see in Table IV that the controllability is very dependent on it. Starting from a low sample period, the fact of increasing it does lets us increase the controllability ($cc'_2 > cc'_1$, $cc'_0 > cc'_2$), but when the optimal sample period (T_{os}) is reached, other increases can provoke a reduction of the number of controllable cells ($cc'_3 \ll cc'_0$). In general, the average optimal time does not improve necessarily if sample period increases.

Regarding Q-learning, finding the optimal sample period (T_{os}) enables us to come close to the optimal time (t'_3) to reach the goal from a specific initial state. Moreover, when adjoining condition increases the time to reach the goal comes close to the optimal, too. According to Table IV, t_3 is closer to t'_3 than t_1 and t_2. One of the difficulties in Q-learning is to find the optimal sample period in a precise way. Therefore, using CACM-RL with an adjoining condition equal to 3 enables us to come close to the optimal time more easily than with Q-learning.

With regard to controllable cells and using CACM-RL, we can achieve a high percentage (∼80%) even with an adjoining condition equal to 1. However, if the adjoining condition increases to 2 or 3 the controllability can rise to more than 90%. In case of using Q-learning, the only case in which we can be sure to have a high controllability (about 90%) is when applying the optimal sample period but, as said before, it is difficult to find it. Therefore, only cc'_0 is similar to cc_2 or cc_3.

Due to the fact that CACM is based on cell-mapping techniques, it needs to know the whole model of the system (set of transitions) and after, from this information, the optimal control is performed. Also, these techniques use fix distances in transitions, which improve the quality of the controller. In the other hand, reinforcement learning techniques do not need the model of the system, but it

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

learns from received experience. They do not use the distance concept for provoking the transitions of the system. Our solution integrates the goodness of both techniques.

5. Conclusions

In this work, we have presented a new optimal control algorithm based on CACM control technique together with a specific reinforcement learning method (Q-learning with pseudorandom planning). We have demonstrated, with the inclusion of concept tests, that the behaviour of the algorithm is a good approximation to the global optimal solution through the adequate selection of the adjoining distance.

The new algorithm has been successfully employed for optimal motion planning of a CLV. Unlike other path planning solutions,[7,10,11] this approach provides a closed-loop solution including forward and backward motion. The technique is robust to changes in the vehicle parameters, since the vehicle model is estimated online and the optimal motion law is updated.

Most of the research focused on vehicle guidance is based on different location and navigation techniques using a previously known map or simultaneous localisation and mapping. Both methods together with their control algorithms have become key components in vehicle navigation. However, these techniques do not take into account the dynamics of the vehicle and they do not learn from the experience but they use information provided by external localisation sensors in order to plan a trajectory in real time.

Next, the advantages derived from the implementation of the new algorithm for optimal motion planning in autonomous mobile vehicles are specified:

1. *Absence of a mathematical model*. The more important difference with respect to CACM is the absence of a mathematical model from which the learning can be obtained. Instead of it, the agent interacts with the environment in order to implicitly create a model of the system/environment.
2. *Sample period is not critical*. The sample period used for the execution of the final controller is not critical. Independently of the chosen value, the controllability is not affected because it is automatically adjusted.
3. *Adaptability to other non-linear-dynamic systems*. The proposed algorithm is valid for other unstable-unmanned vehicles. The more critical part of the new algorithm is the selection of the state variables, which characterise the system. The core of the algorithm is common to any dynamic system.

Our CACM-RL algorithm provides some advantages regarding other algorithms based on reinforcement learning, such as the criticality of the sample period or its adaptability to other unstable-non-linear-dynamic systems. CACM-RL algorithm always gives a solution reaching the goal and this solution will tend to the optimal solution as the cell size is reduced. Our algorithm always works in closed loop applying the optimal control-action at each moment depending on cell, where the vehicle is located.

Acknowledgments

This work has been supported in part by the MICINN under the grant AYA2009-13478-C02-02, the Generalidad Valenciana under Project GV07/214, and the University of Alicante under Project GRE07/4P.

References

1. C. S. Hsu and R. S. Guttalu, "An unravelling algorithm for global analysis of dynamical systems: An application of cell-to-cell mapping," *J. Appl. Mech.* **44**, 940–948 (1980).
2. J. A. Reeds and R. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pac. J. Math.* **145**(2), 364–393 (1990).
3. J.-C. Latombe, *Robot Motion Planning* (Kluwer Academic, USA, 1991).
4. M. Gómez, T. Martínez and S. Sánchez, "Optimal Trajectory Generation Using the Simple Cell-Mapping Method for Wheeled Mobile Vehicles," *Proceedings of Seminario Annual de Automática, Electrónica Industrial e Instrumentación (SAAEI)*, Universidad de Alcalá, Spain (2002) pp. 1–5.
5. F. Lamiraux and J. P. Laumond, "Smooth motion planning for car-like vehicles," *IEEE Trans. Robot. Autom.* **14**(4), 498–502 (2001).
6. L. Rankin and C. D. Crane III, "A Multi-purpose off-line Path Planner Based on an A* Search Algorithm," *Proceedings of the ASME Design Engineering Technical Conferences*, Irvine, California (1996) pp. 1–10.
7. F. Belkhouche, "Reactive path planning in a dynamic environment," *IEEE Trans. Robot*. **25**(4), 902–911 (2009).
8. B. Qin, Y. C. Soh, M. Xie and D. Wang, "Optimal Trajectory Generation for Wheeled Mobile Robot," *Proceedings of the 5th International Conference on Computer Integrated Manufacturing*, Singapore (2000) vol. 1, pp. 434–444.
9. B. Qin, Y. C. Soh, D. Wang and M. Xie, "Trajectory Generation for Velocity-Varying Wheeled Mobile Robot," *Proceedings of the 6th International Conference on Applications of Advanced Technologies in Transportation Engineering*, Singapore (2000).
10. H. Wang, Y. Chen and P. Souères, "A geometric algorithm to compute time-optimal trajectories for a bidirectional steered robot," *IEEE Trans. Robot.* **25**(2), 399–413 (2009).
11. P. Morin and C. Samson, "Control of nonholonomic mobile robots based on the transverse function approach," *IEEE Trans. Robot.* **25**(5), 1058–1073 (2009).
12. T. Fraichard and J. M. Ahuactzin, "Smooth Path Planning for Cars," *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea (2001) pp. 3722–3727.
13. J. Barraquand and J. C. Latombe, "On nonholonomic mobile robots and optimal maneuvering," *Revue d'Inteligence Artificielle* **3**(2), 44–103 (1989).
14. M. Krödel and K-D. Kuhnert, "Reinforcement Learning to Drive a Car by Pattern Matching," Computer Science, *Proceedings of the 24th DAGM Symposium on Pattern Recognition*, Lecture Notes in Computer Science, vol. 2449, Springer, Berlin (2002), pp. 322–329.
15. E. Brunskill, B. R. Leffler, L. Li, M. L. Littman and N. Roy, "Provably efficient learning with typed parametric models," *J. Mach. Learn. Res.* **10**, 1955–1988 (2009).
16. S. Hsu, "A discrete method of optimal control based upon the cell state space concept," *J. Optim. Theory Appl.* **46**(4), 547–569 (1985).
17. M. Papa, H. M. Tai and S. Shenoi, "Cell Mapping for controller design and evaluation," *IEEE Control Syst. Mag.* **17**, 52–65 (1997).
18. T. Martínez-Marín, Diseño de controladores óptimos combinando técnicas de Cell-Mapping y redes neuronales para el control de sistemas dinámicos no lineales *Ph.D. Dissertation* (Madrid, Spain: Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT), Universidad Politécnica de Madrid, 1999).

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press

19. F. Song and S. M. Smith, "Cell-state-space-based search," *IEEE Control Syst. Mag*. **22**(4), 42–56 (2002).
20. P. J. Zufiria and T. Martínez-Marín, "Improved optimal control methods based upon the adjoining cell mapping technique," *J. Optim. Theory Appl.* **118**(3), 654–680 (2003).
21. M. Gómez, T. Martínez, S. Sánchez and D. Meziat, "Optimal control applied to Wheeled Mobile Vehicles," *Proceedings of the IEEE International Symposium on Intelligent Signal Processing*, Universidad de Alcalá, Spain (2007) pp. 83–88.
22. M. Gómez, T. Martínez-Marín, S. Sánchez and D. Meziat, "Optimal Control Applied for Wheeled Mobile Vehicles Based on Cell Mapping Techniques," *Proceedings of the IEEE Intelligent Vehicles Symposium*, Eindhoven University of Technology, The Netherlands (2008) pp. 1009–1014.
23. M. F. Hsieh and Ü. Özgüner, "A Parking Algorithm for Autonomous Vehicle," *Proceedings of the IEEE Intelligent Vehicles Symposium*, Eindhoven University of Technology, The Netherlands (2008) pp. 1155–1160.
24. H. G. Jung, D. S. Kim, P. J. Yoon and J. Kim, "Two-Touch Type Parking Slot Marking Recognitions for Target Parking Position Designation," *Proceedings of the IEEE Intelligent Vehicles Symposium*, Eindhoven University of Technology, The Netherlands (2008) pp. 1161–1166.
25. L. Fu, A. Yazici and Ü. Özgüner, "Route Planning for OSU-ACT Autonomous Vehicle in DARPA Urban Challenge," *Proceedings of the IEEE Intelligent Vehicles Symposium*, Eindhoven University of Technology, The Netherlands (2008) pp. 781–786.
26. S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian and P. Mahoney, "Stanley, the robot that won the DARPA grand challenge," *J. Field Robot.* **23**(9), 661–692 (2006).
27. H. J. Kushner and P. Dupuis, "Numerical Methods for Stochastic Control Problems in Continuous Time," **In:** *Applications of Mathematics* (Springer-Verlag, Berlin, 1992).
28. R. Munos and A. Moore, "Variable Resolution Discretization in Optimal Control," *Mach. Learn.* **49**, 291–323 (2002).
29. M. Gómez, T. Martínez-Marín, S. Sánchez and D. Meziat, "Integration of cell-mapping and reinforcement learning techniques for motion planning of car-like robots," *IEEE Trans. Instrum. Meas.* (Special Issue) **58**(9), 3094–3103 (2009).
30. M. Gómez, L. Gayarre, T. Martínez-Marín, S. Sánchez and D. Meziat, "Motion Planning of a Non-Holonomic Vehicle in a Real Environment by Reinforcement Learning," *Proceedings of the International Work-Conference on Artificial Neural Networks (IWANN '09),* Lecture Notes in Computer Science, vol. 5514, Springer, Berlin (2009), pp. 813–819.
31. R. S. Sutton, "Reinforcement Learning Architectures," *Proceedings of the International Symposium on Neural Information Processing (ISKIT '92)*, Fukuoka, Japan (1992).
32. R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction* (MIT Press, USA, 1998).
33. P. J. Zufiria and R. S. Guttalu, "The adjoining cell mapping and its recursive unravelling. Part I: Description of adaptive and recursive algorithms," *Nonlinear Dyn.* **4**(4), 204–226 (1993).
34. C. J. C. H. Watkins, Learning from Delayed Rewards *Ph.D. Dissertation* (Cambridge, England: Cambridge University, 1989).
35. L. P. Kaebling, M. L. Littman and A. W. Moore, "Reinforcement learning: A survey," *Artif. Intell. Res.* **4**, 234–285 (1996).
36. C. J. C. H. Watkins and D. Peter, "Technical note: Q-learning," *Mach. Learn.* **8**, 249–292 (1992).
37. D. P. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming* (Athenea Scientific, USA, 1996).
38. R. E. Bellman, *Dynamic programming* (Princeton University Press, USA, 1954).
39. Moore and C. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Mach. Learn.* **13**, 103–130 (1993).
40. R. S. Sutton, "First Results with Dyna, an Integrated Architecture for Learning, Planning, and Reacting," **In**: *Neural Networks for Control* (MIT Press, 1990).
41. M. Gómez, "Website of Research Works Related to Motion Optimal Planning and Reinforcement Learning Applied to Autonomous Mobile Vehicles," Videos Online: http://atc1.aut.uah.es/∼mariano/Research/OptimalControl_research.html (2010).

https://doi.org/10.1017/S0263574711000452 Published online by Cambridge University Press