2012 Intelligent Vehicles Symposium
Alcalá de Henares, Spain, June 3-7, 2012

Optimal Motion Planning based on CACM-RL using SLAM

T. Arribas, M. Gómez and S. Sánchez

Abstract— **This work aims to integrate SLAM into the path planning based on Control Adjoining Cell Mapping and Reinforcement Learning (CACM-RL) algorithm to give a total autonomy and auto-location to mobile vehicles. This way, the implementation does not depend on any external device (e.g. camera) to perform optimal control and motion planning. SLAM is performed using Particle Filtering based on the information provided by inexpensive ultrasonic sensors and odometry. A real scenario, in where some obstacles have been introduced, is used to demonstrate the efficiency and viability of the proposed technique.**

I. INTRODUCTION

PATH planning and optimal control algorithms for solving trajectory generation for vehicles and robots have been addressed in several research projects. Those mobile platforms are nonlinear dynamic systems whose motion laws have been widely studied [1-2]. However, their motion planning and control are difficult tasks since they are typically nonholonomic systems.

According to [2], motion planning can be stated as: "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, (time, distance or energy), is minimized, the optimal motion problem is addressed. Besides solving the basic 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.

Global optimal motion planning of vehicles and robots remains today an open problem. This way, research is focused on different methods based on trajectory generation strategies. All have the same objective: to obtain a sequence of recommended real movements avoiding the obstacles present in the environment through which the vehicle is travelling. Research carried out throughout history to achieve the optimal motion planning has used different solutions based on different methods: *open-loop* and *closed-loop*. *Open-loop* or *offline* approaches [1] [3-6], which find 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. The goal of these methods is to follow 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. On the contrary, *closed-loop or online* solutions [7-9] are more desirable since the vehicle is usually subjected to perturbations and uncertainty within environment.

There are other research techniques that are focus on achieving an optimal motion planning basing on a closed-loop approach but without a mathematical model of the vehicle. These techniques are based on reinforcement learning [10], where the optimal motion is estimated through an interaction between the vehicle and its environment. However, the achieved solution depends heavily on the sample period. Other approaches based on Cell Mapping techniques offer an effective numerical solution to obtain global and local solutions of the nonlinear optimal control problem [11-16]. These approaches are very adequate to solve optimal control problems due to the fact that they are very difficult to solve both analytically and numerically. Taking into account the advantages associated with the reinforcement learning methods and Cell Mapping techniques, in [15] a new optimal motion planning algorithm called CACM-RL is proposed by the authors.

Another open research topic in mobile robotics is its Simultaneous Location And Mapping or SLAM. SLAM was introduced by R. C. Smith and P. Cheeseman [17] and it tries to estimate the position and orientation of the vehicle or robot while it generates the map of the environment in parallel. Needless to say that the accuracy and the characteristics of the sensors contribute to the quality of the results provided by SLAM. Due to this fact, most of the proposed solutions based on SLAM rely on the use of high quality sensors such as laser rangefinders. Only a few works make use of low quality sensors such as sonar sensors.

In the work presented in this article, SLAM techniques based on Particle Filtering have been incorporated into CACM-RL to achieve a fully autonomous technique to perform optimal motion planning. In [19], Fast SLAM method is described, where Particle Filtering is also used to handle the paths. It is important to highlight that we have reduced the number of inexpensive sensors to only one ultrasonic sensor over a rotation mechanism, which has a degree of accuracy like 360 sensors. There are some works [18] where SLAM is achieved but using several expensive sensors (a ring of till 24 sensors). Other approaches, like [20], also use inexpensive sensors and due to the fact that the readings are not very accurate, the authors propose a robust sonar feature detection algorithm. However, the most

S. Sánchez and M. Gómez are with the Computing Engineering Department. University of Alcalá, Spain and T. Arribas is with the Signal Theory and Communications Department. Universidad de Alcalá, Spain.

remarkable aspect of this work is to use statistical functions to improve the accuracy of SLAM technique. In this sense, the cost of the used sensors is reduced.

II. WHEELED MOBILE VEHICLE

Two different platforms have been used in this work. Both are based on LEGO® Mindstorms® NXT 2.0 kit. Taking into account that both platforms have different architectural topology, and due to the fact that the CACM-RL algorithm does not need the mathematical model of the platform, the comparison between both platforms can be done easily using the same control algorithm. The robots used to solve the challengers have been built using the LEGO NXT platform and the architecture chosen is a differential robot with two-wheel drive and a third free wheel. The other chosen vehicle is a conventional four-wheel car with two steering wheels and two-wheel drive, model known as Ackermann vehicle.

In Figure 1 the Ackermann and Differential platform are shown. On the left, the vehicle steering is implemented using Ackermann steering geometry with the ability of rotate up to ±22 degrees (ψ). The rotation radius (Φ) is defined by the distance between axes (L) and the steering angle, (ψ). The second platform shown in Figure 1 on the right is a differential Tribot with two traction and steering wheels and a third freewheel responsible for maintaining the balanced system in the plane. This kind of building structure is very common in robotics due to its simplicity and maneuverability, being able to rotate up 360 degrees without displacement.

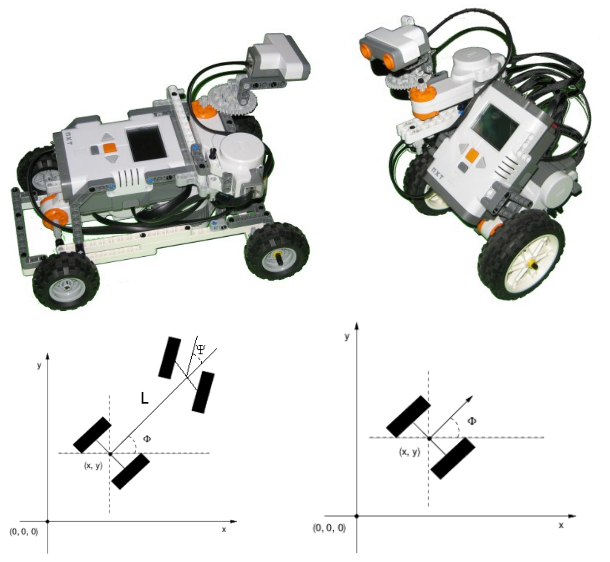

Fig. 1. Wheeled Ackermann and Wheeled Differential Tribot vehicles.

The auto-location of both platforms is estimated from the odometry equations, shown in Table II and III. The odometry is based on the information provided by the servomotors' internal encoders and by the sonar sensor. The encoders provide measurements that are very sensitive to accumulate errors (imperfections, landslides or kidnapping). This makes necessary to introduce some external measurements to correct this problem. Some expensive devices like GPS, laser or visual external processing have demonstrated to be useful in this issue, but in this work an inexpensive mechanism based on sonar detection has been selected.

III. SLAM DESCRIPTION

SLAM is a technique used by robots and autonomous vehicles to build up a map within an unknown environment or to update a map within a known environment keeping track of their current location. Two independent processes bind in a loop to obtain the auto-location are used. Additionally, a feedback between both processes is performed to enhance the accurate in the mapping and auto-location.

In SLAM, the mapping is the first task that has to be solved. To do so, it is necessary to establish the sensors and the methodology to measure the relative distance between the objects and the vehicle. The sensor arrangement can be objective (out of the vehicle) described in [15], or subjective (on the vehicle). Typical sensors used in SLAM are based on sonar or infrared devices (short distance), video camera (medium and long distances) and laser detector (medium and long distances). Each sensor will require a specific processing technique to obtain the relative distance between the vehicle and the objects that will shape the map.

In this work, SLAM is implemented using a single sonar sensor. The mapping and auto-location processes developed are joined by the Particle Filtering technique, merging the odometry data obtained from the vehicle wheels tachometer, with the sonar detection and the map. The following paragraphs describe the tasks that the SLAM algorithm implements:

Mapping: is the process to represent the real world with a digital map. It is based on the information obtained from the sonar sensor using the Line Feature Detection technique described in [20]. In this work, we employ a single rotating sonar sensor instead of an array of sensors. Once the Line Map is built from the sonar detection, a process of exploding converts the line map (see Figure 2 on the left) in a Gaussian likelihood Map (see Figure 2 on the right).

Auto-Location: the state space estimation is based on the recursive implementation of *Monte Carlo statistical signal processing*, also known as *Particle Filtering*. In this article, the position and speed is referenced into a statistical map created from the digital map built with the information provided by the sonar detection. The match to the location on (x, y) plane is done thanks to the information provided by the odometry sensors and sonar distance measurement from the real world.

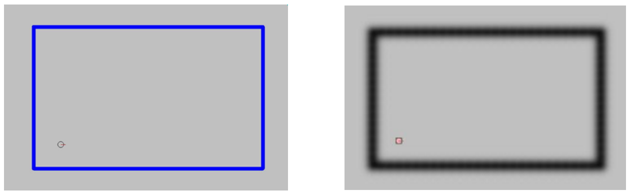

Fig. 2. Line Map and Likelihood Map.

Particle Filtering: is the sequential Bayesian estimation for nonlinear dynamic state space models. It involves recursive estimation of filtering and predictive time varying signals. The goal with this technique is the estimation of the state space variables from the distributions of unobserved time varying signals based on noisy observations. Particle Filtering algorithm is extensively described in [21]. Figure 3 describes graphically the workflow of Particle Filtering implementation, referencing in parentheses the algorithm tasks implemented in Table I.

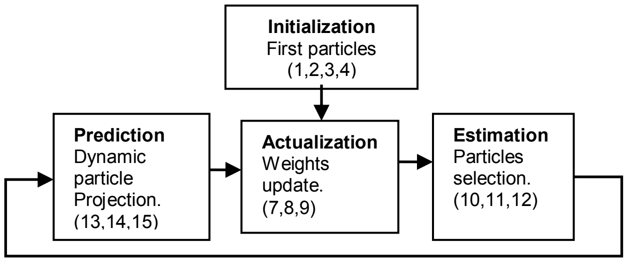

Fig. 3. Workflow of the Particle Filtering algorithm.

In our case, the unobserved internal variables: position, x, y; orientation, ϕ: velocity, v, and the noisy observations are obtained from the odometry process and external measure (sonar device). Let x_n be the unobserved state space variable of a dynamic system. In this case, $p(x_n|x_{n-1})$ is the probability distribution where x_n is the state variable in the instant n and x_{n-1} contains the previous state. If y_n is the noisy observation of x_n, $p(y_n|x_n)$ is the observation equation distribution conditioned on the unknown state of x_n, which is estimated, and $p(x_0)$ is the known initial probability distribution. Then, from the Bayes theorem:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

and

$$P(B) = \sum_{j=1}^{n} P(B|A_j)P(A_j)$$

Described in state space and assuming that $y_{0:n} = \{y_0, y_1, \dots, y_n\}$ as all the observations up to time n:

$$p(x_n|y_{0:n}) = \frac{p(x_n|y_{0:n-1})p(y_n|x_n)}{\int p(x_n|y_{0:n-1})p(y_n|x_n)dx_n} \qquad (1)$$

The predictive state of x_{n+1}can be expressed as:

$$p(x_{n+1}|y_{0:n}) = \int p(x_{n+1}|x_n)\, p(x_n|y_{0:n})dx_n \qquad (2)$$

The Particle Filtering approximates the predictive distributions (1) and (2) by Gaussian densities using the Particle Filter algorithm described in Table I.

The basic idea of Monte Carlo methods is to represent a distribution of a random variable by a collection of samples (particles).

If $N(x;\mu,\Sigma)$ is the density of a Gaussian random variable X, and assuming that at n=1, we have $p(x_1|y_0) = N(x_1;\mu_1,\Sigma_1)$ projected by the movement model, we will update this value in next measure, y_n, according to the following rule based on (2):

$$p(x_n|y_{0:n}) \simeq \frac{N(x_n;\mu_n,\Sigma_n)\, p(y_n|x_n)}{\int p(x_n|y_{0:n-1})p(y_n|x_n)dx_n}$$

where we can consider C_n as a normalized constant:

$$C_n{=}(\int p(x_n|y_{0:n-1})p(y_n|x_n)dx_n)^{-1}.$$

Then,

$$p(x_n|y_{0:n}) \simeq C_n\, N(x_n;\mu_n,\Sigma_n)\, p(y_n|x_n)$$

Finally, as it only propagates the mean and covariance vectors of the Gaussian density using importance sampling, it can be approximated by the following expression:

$$\overline{p}(x_n|y_{0:n}) \simeq N(x_n;\mu_n,\Sigma_n)$$

TABLE I
PSEUDOCODE OF THE PARTICLE FILTERING

```
1:  Draw samples $(x_0)^i\ p(x_0)$ $i = \{1,\dots,N\}$
2:  Compute $\mu_1 = \sum_{i=1}^N x_0^{(i)}$
3:  Compute $\Sigma_1 = \sum_{i=1}^N(\mu_1 - x_0^{(i)})(\mu_1 - x_0^{(i)})^T$
4:  Approximate $p(x_1|y_0) \simeq N(x_1;\mu_1,\Sigma_1)$
5:  FOR $k=1$ to $K$ (total span of time)
6:    FOR $i=1$ to $N$ (total number of particles)
7:      Draw samples $\{x_k^{(i)}\}_{i=1}^N \sim N(x_k = x_k^{(i)};\mu_k,\Sigma_k)$
8:      Compute weights $w_k^{(i)} = p(y_k|x_k^{(i)})$
9:      Normalize weights $w_k^{(i)} = \frac{w_k^{(i)}}{\sum_{j-1}^N w_k^{(j)}}$
10:     Estimate mean $\mu_k = \sum_{i=1}^N w_k^{(i)} x_k^{(i)}$
11:     Estimate covariance
        $\Sigma_k = \sum_{i=1}^N w_k^{(i)}(\mu_k - x_k^{(i)})(\mu_k - x_k^{(i)})^T$
12:     Draw samples $\{x_k^{(i)}\}_{i=1}^N \sim N(x_k = x_k^{(i)};\mu_k,\Sigma_k)$
13:     Draw predictive samples
        $\{x_{k+1}^{(i)}\}_{i=1}^N \sim p(x_{k+1} = x_k^{(i)})$
14:     Estimate predictive mean $\mu_{k+1} = \sum_{i=1}^N w_k^{(i)} x_{k+1}^{(i)}$
15:     Estimate predictive covariance
        $\Sigma_{k+1} = \sum_{i=1}^N w_k^{(i)}(\mu_{k+1} - x_{k+1}^{(i)})(\mu_{k+1} - x_{k+1}^{(i)})^T$
16:   END FOR $i$
17: END FOR $k$
18: $x_k = \mu_k$
```

Finally, the Particle Filtering implementation has to be tuned considering parameters like number of particles or a threshold in particle selection. Both are required to adjust the maximum permitted error or the convergence time of the estimation. These parameters are related to the maximum velocity of the system or the maximum acceptable error.

In our vehicles the position coordinates x, y and θ are estimated from two sources, the internal odometry provided by the encoders and the external location obtained from the sonar. Table II describes how to obtain the vehicle coordinates from the linear accumulated advance from the right and left wheels in a differential Tribot vehicle, and Table III describes how to obtain the vehicle coordinates from the linear accumulated advance from the rear wheels and the steering angle in an Ackermann vehicle.

TABLE II
DIFFERENTIAL TRIBOT ODOMETRY ALGORITHM

1:	Mr = lineal advance right wheel in encoder units.
2:	Ml = lineal advance left wheel in encoder units.
3:	*Dl = (Ml - Dl_1) * P* ‘*(lineal displacement (in meters))*
4:	*Dr = (Mr - Dr_1) * P* ‘*(P: m/encoder unit conversion)*
5:	*Dl_1 = Ml*
6:	*Dr_1 = Mr*
7:	*Dc = (Dl + Dr) / 2*
8:	*dθ = ((Dr - Dl) / B)* ‘*(B: intra-wheel distance)*
9:	***θ***= *θ +dθ*
10:	***x***=*x+Dc*cos(θ)*
11:	***y***=*y+Dc*sin(θ)*

TABLE III
ACKERMANN ODOMETRY ALGORITHM

1:	Ψ = Steering angle
2:	M= Lineal traction advance encoder.
3:	*Dc=(M-Dc_1)*P* ‘*(P: m/encoder unit conversion)*
4:	*Dc_1=M*
5:	*dθ = Sin(Ψ) * Dc/ L*
6:	***θ*** = *θ +dθ*
7:	***x*** = *x + Dc * Cos(θ) * Cos (Ψ)*
8:	***y*** = *y + Dc * Sin (θ) * Cos(Ψ)*

SLAM Algorithm: the iterative processes of sonar detection, Particle Filtering and mapping update is synchronized by the execution of an algorithm that evaluates the quality of the estimation based on the statistic variance of the auto-location vector (x, y, speed, ϕ). The SLAM algorithm is described in the following steps:

1: Information from the sensors is used to draw the first reference segments. No statistical evaluation can be used.
2: Once the map is initialized, like in Figure 2 on the left, the algorithm starts the likelihood edges map generation, like in Figure 2 on the right, from the Line Map. This process converts the discretized Line Map into a kernel point map, where each point will be characterized by the position and orientation.
3: The sonar detection process scans the perimeter by echo and stores each position and orientation location.
4: The vehicle coordinates (Auto-Location vector) are projected to the expected position according to the advance and the current orientation.
5: The Particle Filtering algorithm process the recent information obtained in step 3 locating the (x, y, ϕ) vehicle location in the max likelihood place.
6: The local vehicle (x, y, ϕ) coordinates are updated according to the maximum likelihood place from step 5.
7: The echo sonar detection stored in 3 is updated according to the correction in 6.
8: The Line Map information is updated with the last echo sonar detection stored in 3 and corrected in 7.
9: Go to step 2.

IV. CACM-RL DESCRIPTION

The Cell Mapping techniques include, on the one hand, an efficient application of the numerical methods in order to integrate nonlinear systems and, on the other, the Bellman’s Principle of Optimality to find the optimal control efficiently. Designing controllers based on the Cell Mapping techniques [12-16] results to be an efficient optimal control method for nonlinear systems. 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. A cell-to-cell mapping can be derived from the dynamic evolution of the system. In [12-14], solutions based on cell mapping techniques for the design of optimal controllers are proposed. In [14], the method called Control Adjoining Cell Mapping (CACM) is implemented and it consists of the creation of a cell mapping where only transitions between adjoining cells are allowed.

It is necessary to define a control vector, which can only take some finite values, N_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 may go to a new state, 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 from different origin states. The maximum size of the set of transitions is N_c x N_u, where N_c is the total number of cells or states.

Reinforcement learning methods only require a scalar reward (or punishment) to learn to map situations (states) in actions [10]. 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 state, and applied policy. The objective is to find the actions (policies) that maximize the accumulated reward in each state.

The new algorithm called CACM-RL proposed by the authors in [15] combines the Cell Mapping techniques and the reinforcement learning approaches in order to conceive a single efficient optimal control algorithm. In this work, we have integrated a SLAM approach in CACM-RL in order to give total autonomy the mobile vehicle. CACM-RL deals with different data structures to store the partial results of the learning process. These structures are described below of Table V.

TABLE V
CACM-RL ALGORITHM

```
1:  Initialise Q-Table(s,a) and Model_Table
2:  x ← current state
3:  x' = x
4:  s ← cell(x)
5:  s' = s
6:  REPEAT
7:    IF s'∈drain or s'∈goal or s'∈safety_area
8:    THEN f_reactive(x')
9:    ELSE IF Dk-adjoining(x,x')
10:           THEN  Q-Table(s,a) ← s',r
11:                 Model_Table ← IT(x,x')
12:   x = x'
13:   a←policy(s)
14:   Execute action a on the vehicle
15:   Observe the new state x' and r after Ts
16:   s' ← cell(x')
17:   s ← cell(x)
18: UNTIL the end of the learning stage
    REPEAT
19:   FOR each (s,a)
20:     x ← state(s)
21:     x' ← DT(x)
        s' ← cell(x')
22:     Q_Table(s,a) ← s',r
23: UNTIL EPISODES < N_EPI
```

- *Q_Table(s,a)* is where the accumulated reward for each pair state-action, *Q(s,a)*, is saved. From this table, the optimal policy *a** is obtained. The table is updated according to the one-step equation [15].
- *Model_Table(s,a)* is the local model of the vehicle. It contains an average of local transitions for each velocity (as many as cells), and for each control. The transitions have to satisfy the adjoining property to assure a good approximation to the optimal policy.
- *policy(s)* selects a specific policy to estimate the *Model Table* and to exploit the best policy acquired.
- *IT(x,x')* 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. In order to perform a real approximation in the transformation to local transitions, the real value for each variable is averaged out and filtered.
- *DT(x)* transforms local transitions into global and in the same way, we are able to have the knowledge of the whole state space for each control being applied.
- *f_reactive(x')* lets the vehicle continue the learning, either coming into the state space or moving it to another position. The former is given when the vehicle goes off the state space (drain state) and the latter when the vehicle reaches the goal or the safety are associated to any obstacle.
- *Dk-adjoining(x,x')* detects the transition distance between the current state x' and the previous one, x. The results presented in this work have been achieved using a distance equal 2.
- *cell(x)* transforms a continuous state, *x*, into a discrete value, *s* (cell).
- *state(s)* transforms a discrete value, *s*, into a continuous state, *x*.

V. RESULTS

A software application named "Control Central Station (CCS)" and a physical environment has been developed in order to test the algorithm described in this work. The CCS is connected via Bluetooth with the vehicle. According to the challenge and the vehicle state, the CACM-RL algorithm is supervised in real time by the application.

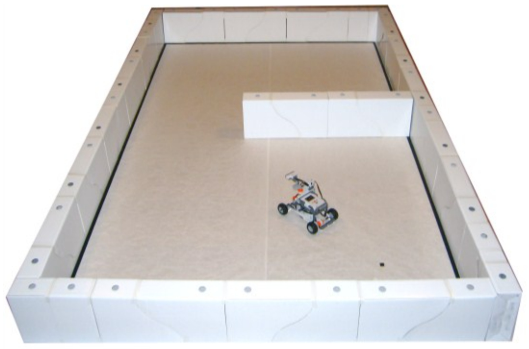

Fig. 4. Perimeter of the test scenario.

The physical environment is defined by the perimeter of a limited rectangle area, with height enough to represent a reflecting surface for the sonar sensor. The rectangle area selected 1.6 x 2.6 m contains a 2 m^2 surface with a 30 cm perimeter of guard, divided by one or two mobile partitions useful to introduce complexity of the route (see Figure 4).

The CCS is used to show in real time the state of the vehicle, to define the test, to compute the CACM-RL control plane and to keep the vehicle under control, according the CACM-RL plane. In Figure 5, a dual image is shown, where it is possible to see the operational view and a virtual reproduction drawing the trajectory with a rectangle shape.

The CACM-RL algorithm is tested with both vehicles solving the same challenge, so the optimal trajectory depends on the maneuverability of each vehicle, as Figure 5 shows. The left picture shows the Ackermann vehicle and the right picture the differential vehicle.

Two different challenges have been chosen to test the algorithm. Two routes with obstacles and a final parking maneuver have to be performed by both vehicles in minimal length and time. On the left part of Figure 5, Ackerman vehicle starts turning left until it reaches the top of the central obstacle and manouvers backward to get the correct angle for going forward and turn again to park backward in the right angle. Comparing with the right part of Figure 5, the differential vehicle turns before advancing, obtaining the right angle without displacement in each turn.

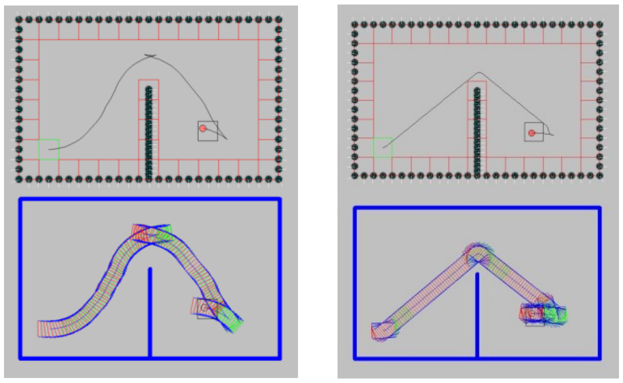

Fig. 5. Operational and Virtual Environment with the SLAM obtained by Ackermann Vehicle and Tribot Differential Vehicle.

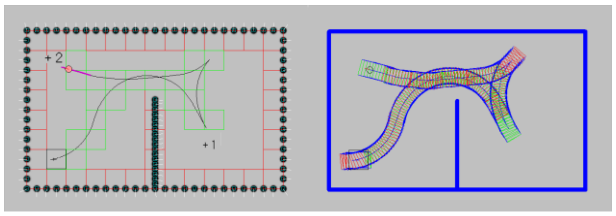

Fig. 6. Multi-trajectory and iterative routes [22-23].

The combination of this test in an infinite loop allows us to check whether the auto-location algorithm is able to ensure that the cumulative error is fixed. Figure 6 on the left shows a sequential multi-trajectory route.

VI. CONCLUSIONS AND FUTURE WORK

In this project, the CACM-RL algorithm combined with the SLAM technique has demonstrated to be a feasible complete solution to control mobile vehicles in unknown environments whereas, indoors or outdoors, with no dependencies on the vehicle model or the holonomicity condition. SLAM limitations are set by the constraints of the CACM-RL algorithm (cells size). In comparison with other control methods, the most important characteristic is the ability to perform a non–linear adaptation in order to optimize any kind of problem. The non-linear control of state space variables developed in this work is extensive to other dynamic variables, like speed, forces, etc. Other points of interest of this method is its ability to learn and adapt the attitude control in real time, avoiding dynamic obstacles or considering changes in the system, like variable overhead, dirt or wear.

ACKNOWLEDGMENT

This work has been supported by the MICINN under the grant AYA2009-13478-C02-02 and by LEGO-Electricbricks that has provided the vehicle prototype and the development environment.

REFERENCES

1. J. A. Reeds and R. A. Shepp, “Optimal paths for a car that goes both forwards and backwards,” Pacific Journal of Mathematics, 145(2): 364-393, 1990.
2. J.-C. Latombe, “Robot Motion Planning,” Kluwer Academic, 1991.
3. L. Rankin and C. D. Crane III, “A multi-purpose off-line path planner based on an A* search algorithm,” In Proceedings of The ASME Design Engineering Technical Conferences, Irvine, California, 1996.
4. B. Qin, Y. C. Soh, M. Xie and D. Wang, “Optimal trajectory generation for wheeled mobile robot,” In Proceedings of the 5th International Conference on Computer Integrated Manufacturing, Singapore, 1: 434-444, 2000.
5. F. Lamiraux and J. P. Laumond, “Smooth Motion Planning for Car-Like Vehicles,” IEEE Transactions on Robotics and Automation, 14(4): 498-502, 2001.
6. T. Fraichard and J. M. Ahuactzin, “Smooth Path Planning for Cars,” In Proceedings IEEE International Conference on Robotics and Automation, Seoul, 2001.
7. J. Barraquand and J. C. Latombe, “On nonholonomic mobile robots and optimal maneuvering,” Revue d’Inteligence Artificielle, 3(2): 44-103, 1989.
8. J. Borenstein, Y. Koren, “The Vector Field Histogram – Fast obstacle avoidance for mobile robots,” IEEE Journal of Robotics and Automation. 7(3): 278-288, 1991.
9. K. Konolige, “A Gradient Method for Realtime Robot Control,” in Proceedings of Intelligent Robots and Systems, 1: 639-646, 2000.
10. R. S. Sutton and A. Barto, “Reinforcement Learning: An introduction,” MIT Press, 1998.
11. C. S. Hsu and R. S. Guttalu, “An unravelling algorithm for global analysis of dynamical systems: an application of cell-to-cell mapping,” Journal of Applied Mechanics, 44: 940-948, 1980.
12. S. Hsu, “A discrete method of optimal control based upon the cell state space concept,” Journal of Optimization Theory and Applications, 46(4), 1985.
13. M. Papa, H. M. Tai and S. Shenoi, “Cell Mapping for Controller Design and Evaluation,” IEEE Control Systems Magazine, 52-65, 1994.
14. P. J. Zufiria and T. Martínez-Marín, “Improved Optimal Control Methods based upon the Adjoining Cell Mapping Technique” Journal of Optimization Theory and Applications, 118(3): 654-680, 2003.
15. M. Gómez, R.V. González, T. Martínez-Marín, D. Meziat and S. Sánchez. “Optimal Motion Planning by Reinforcement Learning in Autonomous Mobile Vehicles,” Robotica, 2011, doi: 10.1017/S0263574711000452.
16. P. J. Zufiria and R. S. Guttalu, “The adjoining cell mapping and its recursive unravelling. Part I: Description of adaptive and recursive algorithms,” Nonlinear Dynamics, Springer Netherlands, 4(4): 204-226, 1993.
17. R. C. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” IJRR, 5(4): 56–68, 1986.
18. J. D. Tardós, J. Neira, P. M. Newman, J. J. Leonard. “Robust Mapping and Localization in Indoor Environments using Sonar Data” International Journal of Robotics Research, 21(4): 311-330, 2002.
19. M. Montemerlo and S. Thrun, “A Scalable Method for the Simultaneous Localization and mapping Problem in Robotics,” Springer Tracts in Advanced Robotics, vol. 27, 2007.
20. J. Choi, S. Ahn and W. K. Chung. “Robust Sonar Feature Detection for the SLAM of Mobile Robot,” International Conference on Intelligent Robots and Systems, 3415-3420, 2005.
21. J. H. Kotecha and P. M. Djuric, “Gaussian Particle Filtering,” IEEE Transactions on Signal processing, 51(10), 2003.
22. CACM-RL Ackermann trajectory in video sample: http://www.youtube.com/watch?v=9RjSBpCgZdM
23. CACM-RL Multi Trajectory Planning in video sample: http://www.youtube.com/watch?v=F4O600pamhs