Auton Robot
DOI 10.1007/s10514-006-7231-8

A purely reactive navigation scheme for dynamic environments using Case-Based Reasoning

C. Urdiales · E. J. Perez · J. Vázquez-Salceda · M. Sànchez-Marrè · F. Sandoval

Published online: 22 April 2006

Abstract This paper presents a new sonar based purely reactive navigation technique for mobile platforms. The method relies on Case-Based Reasoning to adapt itself to any robot and environment through learning, both by observation and self experience. Thus, unlike in other reactive techniques, kinematics or dynamics do not need to be explicitly taken into account. Also, learning from different sources allows combination of their advantages into a safe and smooth path to the goal. The method has been succesfully implemented on a Pioneer robot wielding 8 Polaroid sonar sensors.

Keywords Reactive navigation · Case-Based Reasoning · Learning

1. Introduction

Autonomous robotic navigation, defined as the act of finding and tracking a safe path to a goal, has been widely studied in the last decades. Much research has focused on navigation in dynamic environments, where changes are difficult to model and predict and, consequently, navigation algorithms must be capable of adapting. Both high-level planning and reactive strategies have been proposed to solve this problem, but they present complementary advantages and weaknesses: *high level planners* are efficient, but typically slow to react, while fast *reactive schemes* might usually be unefficient and prone to fall into local traps. *Hybrid systems,* supported by biological evidence (Arkin, 1998), combine both schemes to achieve a better performance. Usually, high level processing follows a deliberative pattern whereas low level control is performed in a reactive way. However, in extremely dynamic environments, navigation may rely uniquely on reactive schemes.

Reactive systems, also referred to as *behaviour based systems,* are a relatively recent development in robotics that has redirected Artificial Intelligence research (Arkin, 1995). These systems rely on coupling sensors and actuators into a set of low level primitive behaviors. Overall actions emerge from the interactions that take place between the individual behaviors, sensor readings and the world. The reactive paradigm has been successfully tested in different robots, including multilegged systems, crawlers, outdoor platforms, mobile manipulators and herds. The high modularity of these approaches makes them particularly appealing for engineers, because it facilitates growth and application of existing approaches to new domains.

One of the best known reactive schemes is the *Potential Fields Method* (PFM) (Khatib, 1986). It consists of creating an artificial repulsion field around obstacles and an attraction field around the goal. Then, paths are obtained by applying gradient descent to the resulting potential field landscape. Potential fields are simple and efficient, but they also present some drawbacks. First, it is difficult to move between close obstacles (e.g., doors). PFM also tend to present oscillations when the robot is close to obstacles. Besides, they may converge to local minima. Some non-purely reactive approaches based in a rough world model have also been proposed.

C. Urdiales (✉) · E. J. Perez · F. Sandoval
Departamento de Tecnología Electrónica, E.T.S.I,
de Telecomunicación University of Málaga,
Campus de Teatinos, Málaga, 29071 Spain
e-mail: cristina@dte.uma.es

J. Vázquez-Salceda · M. Sànchez-Marrè
Departamento de Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya,
c/Jordi Girona 1–3, 08034, Barcelona, Spain

The *Vector Field Histogram* (VFH) (Ulrich and Borenstein, 2000) uses a heading dependent histogram to represent the obstacle density so that the robot can move in the direction where there are less obstacles. It is usual to apply an extended version of VFH to include robot kinematic constraints to the method by reducing possible solutions to only a set of parametrized paths. However, in this case new opportunities cannot be discovered. The *Curvature Velocity Method* (Simmons, 1996) and the *Dynamic Window Approach* (Fox et al., 1997) combine the aforementioned robot kinematics with dynamic constraints and spatial knowledge into a goal function, which is maximized to extract moving commands. These approaches can be combined with a motion planner by providing a map to enhance their performance (Model Based Dynamic Window Approach (Fox et al., 1998)). Further refinements of the same technique include the Global Dynamic Window Approach (Brock and Khatib, 1999), which also relies on a model of the environment. The *Elastic Bands Method* (Quinlan and Khatib, 1993) modifies the trajectory of the robot, originally provided by a planner, by using artificial forces which depend on the layout of the obstacles in the way. Elastic Strips are an enhanced version of the same idea (Brock and Khatib, 2002). Similarly, the PFM can also be hibridized to include kinematics and dynamics constraints (Sekhavat and Chyba, 1999) and to avoid oscillations (Green et al., 1994). Most of these approaches depend on a global model of the environment and, hence, they are not purely reactive, but this is fairly common to avoid local minima. Their main drawback, though, is that they depend on many parameters, used to characterize kinematics and dynamics, to develop a goal function to optimize or to reduce the solution space to a tractable set. These parameters need to be optimized for each specific problem, specially if different robots are used. It is necessary to note that parameter optimization and kinematics analysis is a fairly well known problem which can be solved in a fairly straight way for a given robot. Furthermore, Minguez and Montano (2003) proposed the Ego-KinoDynamic Space based on a transformation to work in a different space with kinematic and dynamic constraints. In this space, planning algorithms can be simplified, as they do not need take kinematics into account. However, if algorithms are to be proven in robots with different kinematic and dynamic constraints, a kinematics evaluation and parameter optimization need to be performed for each different robot. The same could apply if the robot is affected by physical problems like different wheel erosion or motory desynchronization. Obviously, it would be desirable to achieve a reactive system capable of adapting to these circumstances with minimal human intervention and, if possible, with no analytical calculation, so that it could run in a different robot after a minimal, intuitive adaptation stage.

In this paper we propose a purely reactive navigation scheme based on Case-Based Reasoning (CBR). CBR is a learning and adaption technique that helps to solve current problems by retrieving and adapting past experiences, which are stored in the *casebase* in the form of *cases*. CBR is used here to let a robot learn how to react in any given situation, so that it can adapt itself to any configuration or environment change. As proven by our experiments, the main advantage of the proposed approach is that there is no need to study the robot kinematics nor the environment. It will be proven that it also achieves some resistance against sensor errors and provides progressively more efficient ways of avoiding obstacles in the path.

2. A CBR reactive navigation strategy

Case-Based Reasoning is a reasoning, learning and adaptation technique to solve current problems by retrieving and adapting past experiences (Schank and Abelson, 1977; Aamodt, 1994). CBR systems become more efficient by remembering old solutions given to similar problems and adapting them to fit new problems. The new problem together with its new solution becomes a new *case* that can also be stored in the *casebase* to be used later. A CBR system cycle to solve a new problem consists of four steps: (i) *retrieve* the most similar stored *case* to the new current *case*; (ii) *adapt* its solution to the new current *case*; (iii) *evaluate* the results of the proposed solution; (iv) *learn* from the new experience. Consequently, when creating a CBR agent, design decisions often concern: (i) which is the *casebase* or *case* library structure; (ii) how to describe the problem to solve within a particular *case* structure; (iii) how retrieval process and similarity assessment between *cases* can be evaluated; (iv) how to adapt the old solution to solve the new current problem, (v) how to evaluate the success of the proposed solution, and (vi) what to learn and how to learn from solved problems (new experience gain).

CBR has been used in navigation before, usually for global path planning in static environments (Branting and Aha, 1995; Fox and Leake, 1995). There are also approaches for global planning in dynamic environments based on a topological map of an a-priori known environment. For instance, in Haigh and Veloso (1995) new opportunities cannot be discovered when the environment changes unless the topological map is reorganized regularly. Kruusmaa (2003) proposes a grid-based CBR global path planning method to overcome the aforementioned problem. However, she concludes that CBR-based global navigation is beneficial only when obstacles are large and dense and only few solutions exist. Otherwise, the solution space may become too large. Other CBR based methods focus on

non-pure reactive navigation (Likhachev and Arkini, 2001; Ram and Santamaria, 1993), but they basically rely on accumulating experience over a time window while navigating in a given environment to obtain an emergent global goal seeking-behaviour. Hence, they are not adequate for highly dynamic environments where the layout of obstacles changes too often. In this work, the *casebase* acts as a collection of sense/act pairs, in a purely reactive sense. It does not gather information about a given environment, but simply how to act when a given sensor/goal configuration is detected and, consequently, it is valid for any environment. The acquired experience simply covers how a robot with a given physical structure would act at a given time instant to reactively reach a goal in a place where there are obstacles at certain positions. It must be noted that this scheme does not provide a path to the goal, but a particular action at each time instant; the path is the emergent behavior resulting from the combination of several cases.

2.1. The sense/act pair: *Case* structure

A CBR system depends on the *case* library organization, which can either be a flat memory or a hierarchical one. Flat memories retrieve the *case* best matching the input *case*. Including new cases is simple, but it is necessary to keep a bounded number of them to grant adequate response times. Hierarchical memories provide more efficient retrieval, but keeping them in optimal conditions requires an overhead on the *case* library organization and the retrieval process could miss some optimal cases by searching a wrong area of the memory. Consequently, a flat memory has been chosen in this paper.

Within flat memories, the most commonly used representation is a *feature-value vector* representation, where a *case* description can be regarded as a vector in an N-dimensional space. This is naturally suitable to represent a sense/act pair. The main concern of reactive navigation is to reach a goal point in a safe way using local data. Hence, a problem instance or *case* consists of the position of the goal and obstacles with respect to the robot, while the output is basically which direction to follow in order to safely reach the goal.

Specifically, the proposed *case* instance includes *the goal direction* and the instant readings of *all on-board sonar sensors*. Besides, since robots are not necessarily holonomic, the heading direction of the robot is also included in the *case*. Figure 1 shows the *case* parameters, including its output: the solution heading direction. Further *case* features will be discussed later.

Polaroid sonar sensors present a wide uncertainty arc, so detected obstacles cannot be precisely located. We achieve a rough spatial integration by evaluating the readings of all on-board sensors at a given instant but, nevertheless, some errors may occur. These errors are filtered out of the system

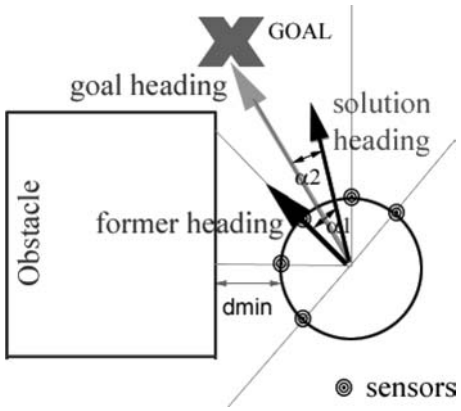

Fig. 1 *Case* definition

by continuously triggering the reactive navigation module. Since false echoes and wrong readings tend to provoke spurious sharp direction changes, unless they last for a while, they are filtered out of the global response of the system because of inertia. Similarly, in complex areas where sonar readings change often, the platform tends to slow down for the same reason.

2.2. Retrieval process and *case* similarity assessment

In flat memories, retrieval consists of matching all *cases* in the *casebase* against the current one. The best similar *case* is usually selected typically by a nearest-neighbour (NN or *k*-NN) algorithm (Watson, 1996), where an evaluation function usually combines all partial matchings through a *case* attribute into a full dimensional partial-matching between *cases*.

We have experimentally checked that when the different features of a *case* are related, as in this case, qualitative matching outperforms quantitative one, as illustrated in the example in Fig. 2. Quantitative matching would return the *case* in Fig. 2(c) as the most similar to the one in Fig. 2(a) given the same goal position, even though it is clear that the *case* in Fig. 2(b) is a better choice to determine where to go. Thus, in this work we rely on the *Tanimoto* distance (Deichsel and Trampisch, 1985), which is less sensitive to differences in a single feature than to global ones:

$$D(X_i, X_j) = 1 - \frac{X_i^T \cdot X_j^T}{|X_i|^2 + |X_j|^2 - X_i^T \cdot X_j^T} \tag{1}$$

It is important to note that minor differences between sensors readings usually correspond to slight robot shiftings rather than to different situations. Thus, such differences should not involve creation of new *cases*. Since it has been proven (Sànchez-Marrè et al., 1999) that it is better to combine discrete and continuous attributes in CBR systems to

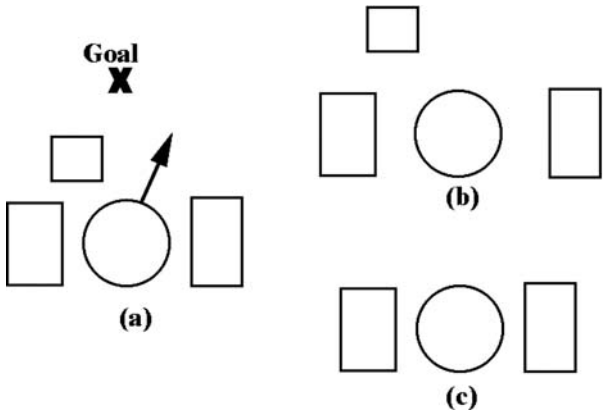

Fig. 2 *Case* selection: (a) stored *case*; (b) input *case* 1; (c) input *case* 2

fully exploit the domain knowledge, the problem instance can be improved by discretizing the sensor readings. There are many discretization techniques (Dougherty et al., 1995), but all of them roughly belong either to automatic partitioning of parameter domain or predefined partitions. In the first case, methods rely either on equal width interval binning or using clustering methods to group partitions relative to the *case* frequency. In the second case, an expert can define reasonable partitions. Since the robot and the sonar behaviour is known, we have predefined a partition. For our purely reactive layer, far obstacles are not important, but the closer an obstacle is, the more dangerous the situation is. Thus, we discretize sonar readings into 5 non-equal intervals: (i) critical (0–20 cm); (ii) near (20–50 cm); (iii) medium (50–100 cm); (iv) far (100–150 cm); and (v) no influence (more than 150 cm). These readings are valid for Polaroid sonar sensors and reactive navigation in a medium scale indoor environment. In large scale situations a different discretization would be required.

2.3. *Case* evaluation and *casebase* reorganization

When defining the CBR system, it is also necessary to provide a way to determine how good a solution is, so that the worst *cases* can be periodically pruned from the *casebase* and, hence, retrieved cases are efficient. It is important to note that in a purely reactive layer, only factors at hand at a given time instant can be evaluated to estimate the efficiency of a *case*. Thus, we cannot rely on global factors such as the distance required to reach a given goal. First, in order to avoid oscillations, one of the evaluation factors is the angle difference between the current robot direction and the output one. Besides, it is interesting not to get too close to obstacles to avoid possible collisions. This does not mean that the robot cannot move close to obstacles, but simply that a solution is better if it is not necessary to do so. It is important to note that if only these two factors—*closeness to obstacles* and *soft direction changes*—are considered, the robot might try to move too far from the goal as long as there were no obstacles nearby and its curvature was stable. Thus, a third evaluation factor to avoid this behaviour is to keep the goal ahead. This third factor is *the angle between the direction to the goal and the resulting one*.

Figure 1 presents all the attributes required to define a *case*. The input instance includes the sensors readings, the robot heading and the goal direction. Then, the *case* is evaluated as a function of the distance to the closest obstacle $d_{\min}$, the angle between the current heading and the output heading α_1 and the angle between the goal direction and the output heading α_2. These factors are combined into an efficiency rank E ranging from 0 to 1:

$$E = \frac{k_1 e^{-C_1 \cdot \frac{1}{d_{\min}}} + k_2 e^{-C_2|\alpha_1|} + k_3 e^{-C_3|\alpha_2|}}{k_1 + k_2 + k_3} \tag{2}$$

k_1, k_2 and k_3 being weight constants that are equal to 1 in this paper. At this point, it would be possible to simply remove all *cases* whose efficiency is lower than a threshold. However, this approach implies that some situations that cannot be solved efficiently—e.g. navigation in a tight corridor where obstacles are always close to the agent—would never be learnt. Thus, the *casebase* is organized by splitting it into classes using a classic MaxMin algorithm. The prototypes of those classes become the cases of the *casebase*. The prototype of a given class x is calculated as:

$$P_x = \sum_{\forall i \in x} E_i case_i \tag{3}$$

The prototype of a class resembles more the most efficient *cases* in that class because E is used as a weight factor. However, classes consisting uniquely of non-efficient *cases* are not removed from the *casebase* as desired.

It is necessary to point out that a MaxMin clustering algorithm only depends on a threshold $CS_{\max}$ which controls the maximum size of a class. $CS_{\max}$, however, is not critical and can be broadly chosen. If $CS_{\max}$ is high, classes are large and the *casebase* presents few *cases*. Otherwise, classes are small and less adaptation is required. In this paper, all tests were performed for $CS_{\max}$ equal to 500, resulting in less than 300 prototypes, to keep a bounded *casebase*.

3. Training, learning and adaptation

Learning is a key cognitive task of CBR systems. There are two major learning approaches in CBR: *learning by observation* and *learning* by *own experience*. Learning by observation (van Lent and Laird, 1998) happens when the *case*

library is seeded with a set of initial *cases*, either by an expert or by direct observation of real data. Learning by own experience (Schank, 1982) is done after each cycle of the reasoner: if the proposed solution has been a succesful one, it can be stored and used for future reference. The reactive scheme proposed in this paper can use both approaches.

Learning by observation is achieved by a module that learns by observing a human, who guides the robot using a keyboard in an obstacle field. Although the robot could also learn by observation from other reactive techniques like the PFM or the DWA, learning from humans has the advantage that a human driver easily adapts to any situation and provides a flexibility to act one way or another in similar situations, coming close to obstacles if necessary or performing sharp direction changes if required. The human driver implicitly takes into account the kinematics and dynamics of the problem, as well as the nature of the environment, so when the robot learns *cases* from a human guided path, such factors are included in the *casebase* in a natural way. It is important to take into account that the human is not bound to drive the robot in a perfect way, so some stored cases may be far from optimal or even erroneous. However, as stored cases are ranked and combined according to their efficiency in the casebase, bad cases are progressively filtered out of the response of the agent and mutations provide better efficiency to the existing ones if necessary.

In order to learn by observation, a human supervisedly guides the robot through a path. Each time the Tanimoto distance between the *case* retrieved from the *casebase* and the current *case* instance (robot heading, goal direction and sensor readings) is larger than a threshold U_{learn}, the *case* is stored in the *casebase*. During this stage, the output of the *case* is obviously not used. Our experiments seem to point out that the robot learns better when different drivers provide different routes to approximately the same destination than when all this training is supervised by a single person.

Learning by own experience is achieved when the robot is moving on its own. In absence of obstacles, the robot tries to reach the goal in a straight way. However, as soon as obstacles are detected nearby, the proposed reactive system is triggered and the most similar *case* in the *casebase* is retrieved. Then, it is evaluated if the Tanimoto distance between the current *case* and the retrieved one is larger than U_{learn}. If the distance is lower, the solution of the retrieved *case* is adopted. Otherwise, the *case* is adapted, used and stored.

In order to solve a new situation, a simplified version of the PFM is used: the goal acts as an attractor and obstacles act as repulsors, so that the combination of all involved vectors provides the resulting trajectory. However, to take into account previous experiences, the resulting direction of the retrieved *case* is also included in the vector combination. Obviously, the so created *case* might not be suitable either. Naturally, in these cases the input instance changes fast. Thus, the reactive module is immediately retriggered and more *cases* are created, where each new *case* is progressively more adapted, until a safe course to the goal is reached.

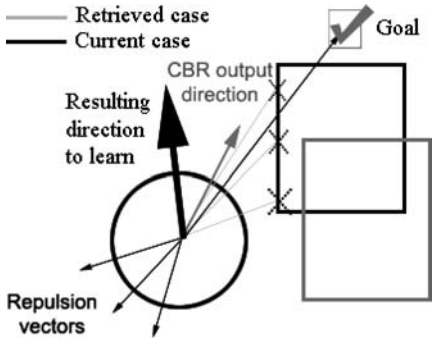

Fig. 3 Learning by experience

Figure 3 shows an example of learning by own experience. When the robot is moving towards the goal, a nearby obstacle on the right, printed in black, is detected. The most similar *case* stored in the *casebase* corresponds to a situation where an obstacle was also on the right, but in a different position. The location of the obstacle in the retrieved *case* is printed in gray in the figure and it can be observed that the solution of the *case*, also printed in gray, is not valid for this new situation because the robot would collide with the obstacle. Naturally, this is detected because the sensor readings are quite different in the current *case* and the retrieved one. Thus, a new *case* is created by handling the obstacle as a repulsor and the goal as an attractor, as represented by the thin black arrows in Fig. 3. After adding these vectors to the solution of the retrieved *case*, a new heading solution is obtained. If further correction is needed, more *cases* are generated. As explained in the previous section, the best adapted cases contribute more to the prototype of their class and, hence, to the future behavior of the agent, than the less adapted ones.

It needs to be observed that supervised training is not necessary in the proposed system. If no training is provided, the robot will try to reach the goal following a straight line (case 0). When obstacles are detected, this single *case* will start to mutate and generate new cases, that will themselves be mutated until an efficient *casebase* is achieved. It has been experimentally checked, though, that convergence is faster if supervised training is provided. It takes longer for an untrained system to achieve short, safe and smooth trajectories.

It can be observed that no explicit considerations about hardware structure, kinematics, sensor errors or environmental factors have been formulated. Thus, if any of these factors changes, the system simply adapts through navigation to the new configuration. This is an interesting feature because it makes the system valid for different robotic platforms

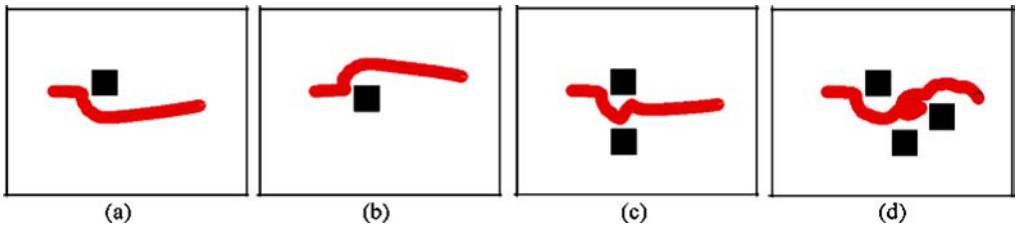

Fig. 4 Performance without learning for: (a) isolated obstacle on the left; (b) isolated obstacle on the right; (c) two obstacles; (d) three obstacles

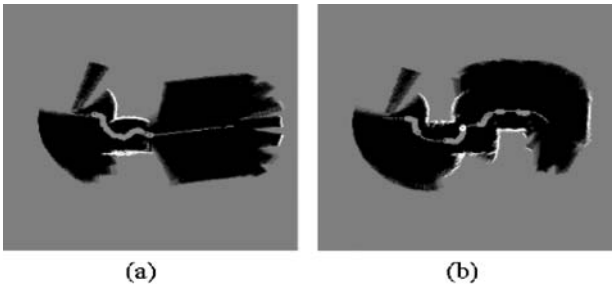

Fig. 5 Performance after non-supervised learning: (a) 2 obstacles; (b) 3 obstacles

without any reprogramming and also spares evaluation of kinematics during the design stage.

4. Experiments and results

The proposed reactive algorithm has been tested on a Pioneer robot equipped with a frontal set of 8 Polaroid sonar sensors. The robot includes a compass, which is used in combination with odometry to obtain the approximate position of the goal with respect to the robot to determine if the goal has been reached. We also use evidence grids (Moravec, 1998) in this section to show where obstacles are from the sonar point of view. In these grids, obstacles are printed in white, free space is presented in black and unexplored areas appear in gray. It is very important to note, though, that the reactive system does not use these grids at all, and they are used only as a visualization tool.

In order to evaluate the results in this section, it needs to be stated that the robot moved at different speeds depending on the situation. During supervised training, the supervisor could make the robot go faster or slower depending on the difficulty of the trajectory. This did not affect learning, as only local information is acquired at a given time instant. On autonomous navigation, we set a maximum speed of 0.6 m/s, as sonars are slow and safety could not be granted for larger velocities. However, speed is implicitly controlled by the nature of the system. When the robot reaches an area where obstacles exist, the reactive system generates changes of direction that provoke a decrease in the translational speed of the robot. If the area is very crowded or complex, the reactive system is triggered very often. The inertia of the system filters some direction changes out of the emergent behavior of the robot, but, nevertheless, it slows down. We experimentally checked that this velocity reduction is more drastic in areas where no suitable cases are available in the *casebase* and, hence, the robot needs to adapt. Under these circumstances, the velocity of the robot drops down drastically, as proposed direction changes are so sharp that the robot does not have time to perform them when a new direction is already proposed by the CBR. Nevertheless, this velocity reduction, in fact a side-effect of the approach, allows safe adaptation to unknown and potentially dangerous situations. After adaptation, though, the robot may go through similar layouts in a much faster way, but speed changes still depend on the environment structure and it may range from 0.2 m/s in narrow corridors and cluttered areas to our maximum 0.6 m/s in situations considered to be safe for the robot, where minimal direction changes are required.

4.1. Training and tests in simulation

First, we tested the algorithm under simulation to keep a completely controlled scenario where the position of the obstacles and the departure and arrival locations of the robot could be kept exactly the same and also to keep performance not affected by sonar and localization errors. The goal of this set of experiments was to evaluate the learning process.

Initially, storage of new cases is not allowed to check how the algorithm works without learning. Figure 4 shows the performance of the robot in four experiments, when the only *case* stored in the *casebase*—to move ahead to the goal in absence of obstacles—is continuously adapted. When the robot deals with a single obstacle, the trajectory is basically correct, even though sharp direction changes occur when the robot detects the obstacle (Figs. 4(a) and (b)). However, in more complex environments significant oscillations (Fig. 4(c)) and loops may occur until a way out of a complex situation is found (Fig. 4(d)). However, if learning by own experience is allowed, the performance of the robot is quite improved (Fig. 5). Locations where the *casebase* is triggered are printed in gray over the evidence grid, and adapted cases are printed in white. It can be observed that: (i) almost no

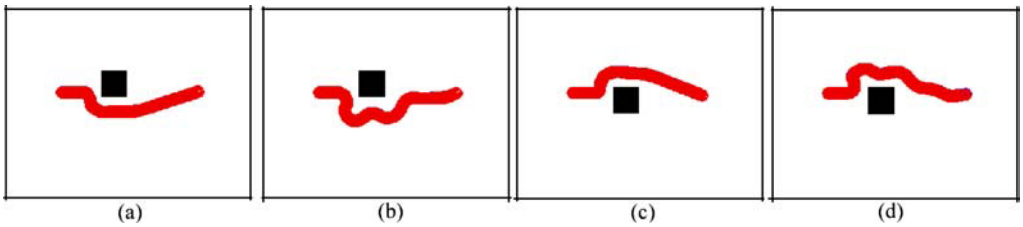

Fig. 6 (a–d) Human supervised learning for an isolated obstacle on the left and right

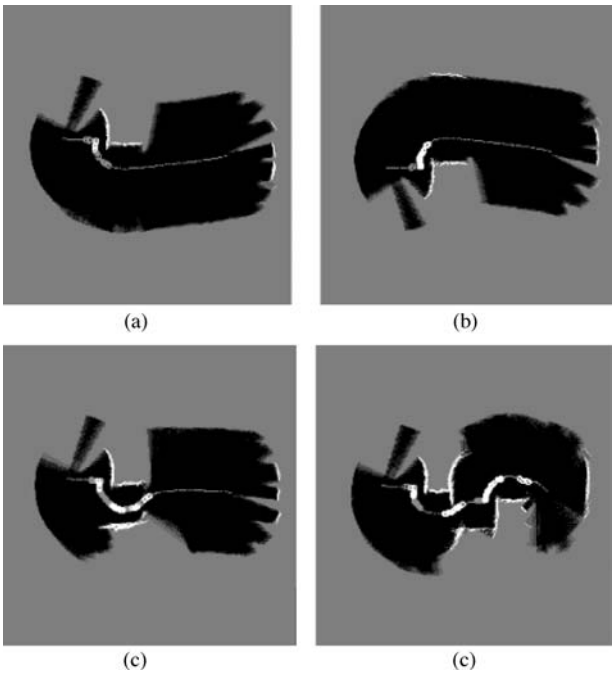

Fig. 7 Performance with human supervised learning for: (a) isolated obstacle on the left; (b) isolated obstacle on the right; (c) two obstacles; (d) three obstacles

further cases are stored; (ii) trajectories are smoother than in Figs. 4(c) and (d); and (iii) no loops appear. It is important to note, though, that this is possible because of the absence of sonar errors under simulation.

A second test focused on learning by observation. In this case, a human guided the robot through the paths in Figs. 6(a)–(d). This test had a double goal: (i) to check if supervised training improves the performance of the robot with respect to non-supervised training and; (ii) to evaluate the potential impact of *cases* learnt from bad, oscillating routes. Hence, the training paths are purposefully non-smooth. Figure 7 shows four tests after learning *cases* through the trajectory in Figs. 6(a) and (b) (obstacle on the left). During these tests, learning by own experience is not allowed. It can be noted that in Fig. 7(a) only a couple of cases are mutated (white circles). This occurs because the robot was forced to move too close to the obstacle during training and some cases corresponded to unsafe situations that are adapted when the robot moves on its own. Figure 7(b) shows an example where the robot must leave the obstacle on the right. Now many new *cases* need to be adapted because the robot was only trained to leave obstacles on the left but the emergent behavior is, nevertheless, the expected one. The experiment in Fig. 7(c) is also interesting. As expected, the robot must adapt cases when it detects obstacles at both its sides, but it can be observed that now the trajectory does not present as many oscillations as in Figs. 4(c) and 5(a). It can also be appreciated that the oscillations in Fig. 6(b) are not cloned in these experiments and, hence, that it is not necessary to be too cautious with training. The same conclusions can be extracted from the test in Fig. 7(d).

We included a PFM in the robot for comparison purposes, as an example of a typical approach where only local information is used. Figure 8 shows how the trajectories resulting of the PFM are basically efficient. They oscillate more than the ones returned by our reactive scheme after human supervised training (Fig. 7), but are smoother. Thus, we checked if we could improve the system by learning from the PFM. Figure 9 shows the performance after learning only how to leave an obstacle on the left from PFM (Fig. 8(a)). As in the previous test, only a few cases are mutated when the situation is similar to the learnt one (Fig. 9(a)). However, it is interesting to note that, while during human training cases were mutated because the robot had learnt to move too close to obstacles (Fig. 7(a)), here they are mutated to avoid oscillations. The situation in Fig. 9(b) is more interesting. The robot should

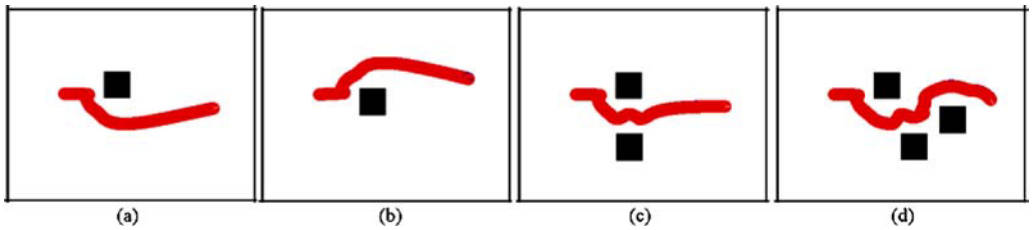

Fig. 8 (a–d) Robot trajectories generated by a pure Potential Fields algorithm

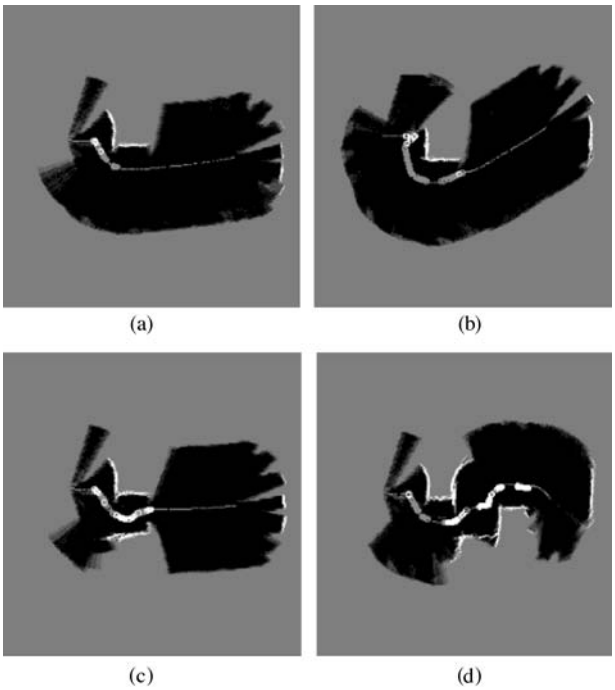

Fig. 9 Performance with potential field supervised learning for: (a) isolated obstacle on the left; (b) isolated obstacle on the right; (c) two obstacles; (d) three obstacles

leave the obstacle on the right this time but, since it learnt to leave it on the left, it chooses a longer trajectory to follow its training rather than adapting. The examples in Figs. 9(c) and (d) show the performance of the robot for two and three obstacles, respectively. It can be observed that the trajectories in these cases are smoother than the ones in Fig. 7, basically because the robot does not move so close to obstacles, but they also oscillate less than the ones in Fig. 8.

It also seemed interesting to combine the experience learnt from humans and from PFM and check if the robot showed a better performance. Figure 10 shows the performance of the robot after the training in Figs. 6 and 8. In this case, white circles are used to mark locations where the retrieved *case* was learnt from PFM, while gray circles represent human *cases*. The robot mostly chooses PFM cases to avoid getting too close to obstacles and human cases in locations where potential fields might provoke oscillations. Naturally, one of the main advantages of controlling the learning process is that it is easy to evaluate what the robot needs to learn and provide the required teaching. Thus, further improvements could be achieved by supervisedly teaching the robot how to move around a corner like the one conformed by the last two obstacles on the right of the map.

A final simulation test consisted of forcing a situation that a pure PFM could not handle by moving obstacles progressively closer until the robot refused to move between them with PFM (Fig. 11(a)). We checked that our system could not

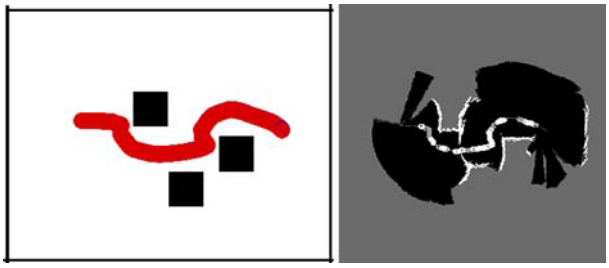

Fig. 10 Reactive trajectory for three obstacles with supervised learning from humans (gray circles) and Potential Fields (white circles)

either solve the situation without training (Fig. 11(b)). However, when it was trained either by own experience (Fig. 4), by a human (Fig. 6), or by PFM (Fig. 8), it solved the situation (Figs. 11(c–e)). This proves that the reactive system is capable of learning in a fast way, even from scratch, to enhance the global performance of the robot.

4.2. Real learning and tests

A second set of experiments similar to the previous one was conducted in the real world. The laboratory where tests were performed is not a large environment, so we set the reactive range to a radius of 0.5 m around the robot to work only with "critical" and "near" obstacles because walls were at medium range. Thus, obstacles farther than 0.5 m did not trigger the reactive layer. Despite this triggering threshold, when the reactive module is triggered all detected obstacles have influence in the resulting solution.

Figure 12(a) shows our human supervised learning stage, where three different persons guided the robot around a single obstacle isolated in the middle of a room. No restrictions were imposed to the drivers. Thus, they could follow long or short trajectories, perform sharp turns if necessary and get as close to obstacles as desired. 160 *cases* were stored after 6 routes. We explicitly avoided supervised training with more obstacles to test how flexible the system is against unexpected situations in real environments. It is important to note that the robot is not working with the *casebase* developed in simulated environments. However, during all these tests the robot is allowed to learn by own experience. Figures 12(b) and (d) show different stages of this unsupervised learning process. Immediately after the supervised learning stage, the robot is sent to a goal so that, in order to reach it, it needs to avoid an obstacle on the left and move closer to the wall than before (Fig. 12(b)). Naturally, the robot needs to adapt to the new situation. After that, the environment gets more complex by including a new obstacle (Fig. 12(c)). This corridor-like situation may seem to be similar to the previous one, but it is not because cardboard boxes behave like refractors for sonar signals and, consequently, sensor readings are noisier in this second test. Thus, more *cases* are added to the library.

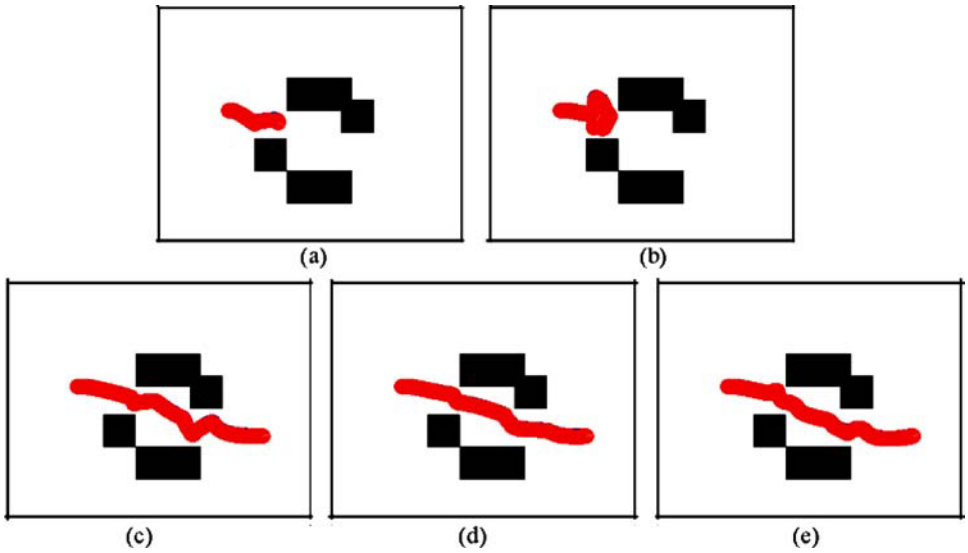

Fig. 11 Trajectories in a difficult situation using: (a) Potential Fields; (b) GBR with no knowledge; (c) GBR with non-supervised learning; (d) GBR with Potential Fields learning; (e) GBR with human learning

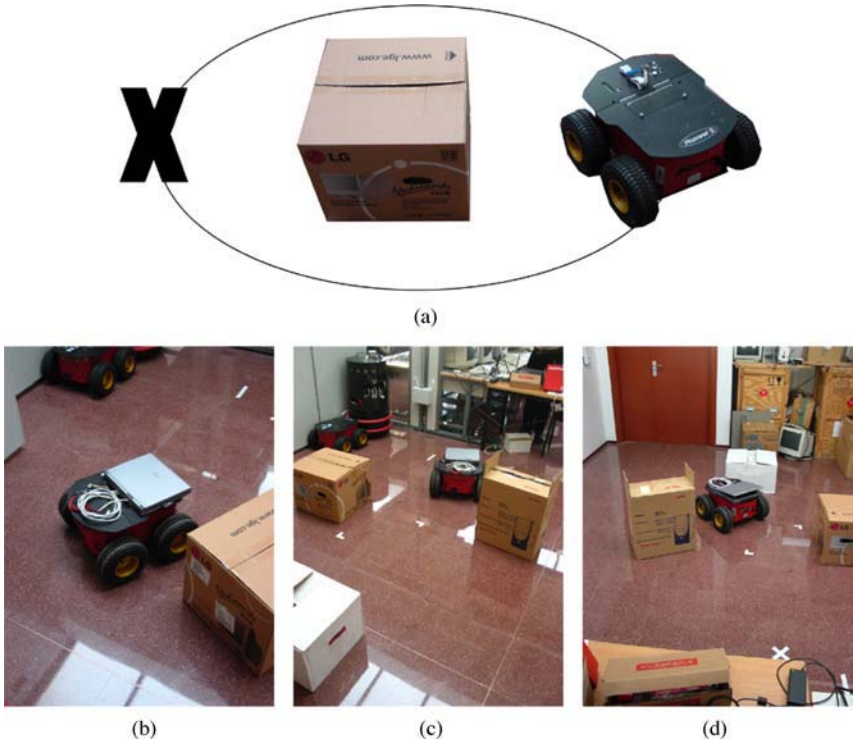

Fig. 12 Training and learning: (a) supervised training plan; learning by experience when (b) a single obstacle is closer to a wall; (c) there are two obstacles; (d) there are three obstacles

Figure 12(d) shows a third path where, in order to reach the goal, the robot must move between three boxes. It is important to note that in all cases the only directive for the robot was to move ahead to the goal.

Figure 13 presents the odometric data of two consecutive runs of the reactive navigation system for two obstacles, positioned at different locations and separations. Obstacles have been manually overimposed to the Figures for a better understanding of the tests. Figure 13(a) shows the first test. We observed that the emergent behaviour of the robot was mainly based on ($Driv_2(1)$), where the robot left a single obstacle on the right. During this experiment, the robot learnt 40 more *cases* related to situations with two obstacles nearby. In order to test this better trained robot, we moved obstacles closer to achieve a more critical situation. In fact, since narrow corridors are a typical situation where pure PFMs may fail, we narrowed the corridor until our PFM either refused to move between the obstacles or collided depending on its configuration parameters. It must be noted that, as long as no *case* adaptation is required, curvature tends to be preserved. Nevertheless, it can be observed that the CBR trajectory is not a straight line either.

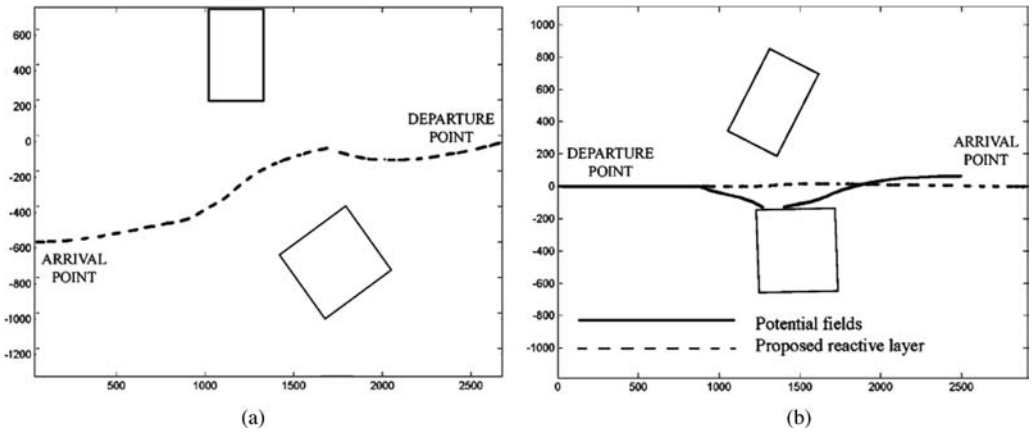

Fig. 13 Navigating and learning with 2 obstacles: (a) first run with 2 obstacles; (b) second run with 2 obstacles

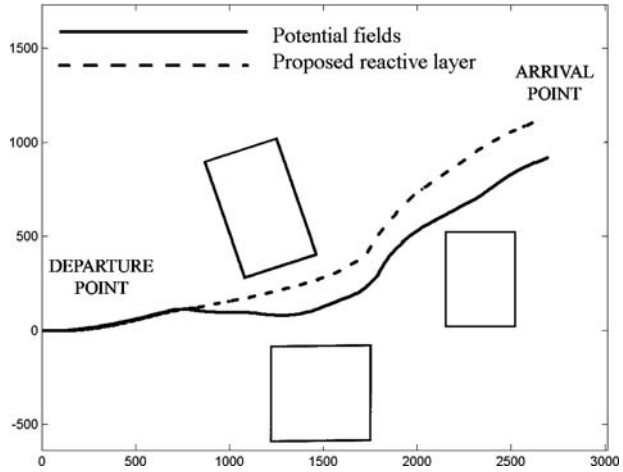

Fig. 14 Navigating with 3 obstacles

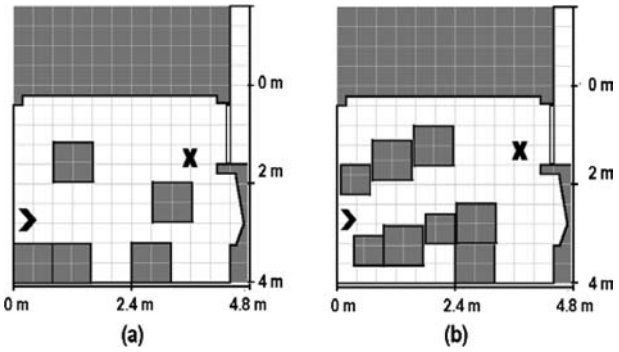

Fig. 15 Real environment configurations: (a) 2 obstacles; (b) corridor obstacles

Figure 14 shows another test, this time in an environment presenting three obstacles. The robot already knows how to handle two obstacles—from previous tests- because this test is performed after the previously described ones. However, it has never met three obstacles in its way. As in the previous test, we have compared the results (dash line) with a pure PFM (continuous line). These obstacles are close enough to provoke oscillations in the PFM trajectory. In our approach, though, the robot does not mind about close obstacles if the CBR has learnt from experience that such trajectory is safe.[1] Thus, it follows a smooth trajectory until it detects an obstacle ahead. At this point (obstacles at both sides and ahead), the retrieved *case* (obstacles at both sides) must be adapted and a sharp trajectory change can be observed. From this point on, the situation is, once more, similar to known ones. It can be observed that the CBR trajectory does not oscillate and is shorter and smoother than the one returned by the PFM.

4.3. Real tests after simulation training

A final set of experiments focused on testing how the robot moved in real environments after only virtual training was provided. During these tests, learning by own experience is not allowed. In the real world sonars are affected by noise and errors and, hence, spurious readings may appear and sharp changes in sensors readings are likely to be detected.

Figure 15 presents a map of the real environment in Fig. 12 and different obstacle layouts. Figure 15(a) shows a first layout where two obstacles are separated approximately 1.20 m. The robot is roughly heading towards the one on the left and the arrival point in these experiments is set so that the robot needs to move between both obstacles or, else, the length of its trajectory would be too long. In fact, we checked that if no training at all was provided, the robot rather tried to leave both obstacles on the left despite the longer path. However, if we include in the *casebase* all *cases* learnt from PFM in Fig. 9, the robot moves between both obstacles as expected (Fig. 16(a)). It can be noted that, while in the simulator tests

[1] In previous experiments, the robot learnt that it could move close to obstacles as long as they were on its sides and it was not moving towards them.

Fig. 16 Tests in the environment in Fig. 15(a) after training: (a) all potential field *cases* learnt in Fig. 9; (b) all human *cases* learnt in Fig. 7; (c) all *cases* learnt during training in simulation

Fig. 17 Tests in the environment in Fig. 15(b) using all *cases* learnt during training in simulation: (a) for a wide corridor; (b) for a narrow corridor; (c) for the corridor in (b) after the *cases* in (b) are added to the *casebase*

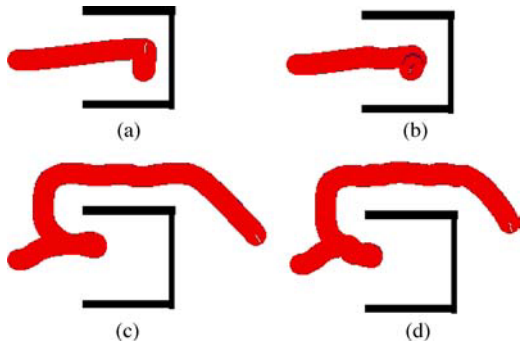

Fig. 18 Reactive tests: (a) PFM; (b) CBR; Hybrid tests: (c) PFM; (d) CBR

obstacles were clearly defined in the evidence grids, they can hardly be discerned now in the real tests, due to sonar errors.

It can be observed in Fig. 16(a) that the robot mostly relies on its training, even though it was performed under simulation, until a new situation is found. The obstacle layout in this experiment is similar to all three obstacles experiments under simulation minus the obstacle in the bottom of the image. Thus, some new *cases* need to be acquired. Besides, whenever the robot moves between close obstacles in real environments, sonar errors appear and, sometimes, *cases* are mutated because of these errors even though similar situations were learnt under simulation. Mutations occur when the robot is moving between both obstacles. We checked afterwards the sonar readings in that segment of the path and, indeed, sometimes obstacles were not detected. However, since the reactive scheme is continuosly retriggered in such areas, errors were implicitly filtered out of the emergent behavior of the robot because of inertia. Figure 16(a) also shows that when the robot learns from PFM, trajectories present oscillations when obstacles are very close. Figure 16(b) shows the same experiment using all *cases* learnt from a human in Fig. 7. As observed under simulations as well, when the robot learns from a human, resulting trajectories are globally not as smooth as when it learns from PFM. Figure 16(c) shows the same experiment in the real environment in Fig. 15(a). It needs to be noted that the robot is not allowed to learn by own experience in these experiments, so it still cannot recognize the obstacle configuration in Fig. 15(a) and some *cases* need to be adapted. However, it can be clearly appreciated how the resulting trajectory is smooth and presents no sharp oscillations.

A second set of real experiments after training in simulation consisted of sending the robot across the corridor in Fig. 15(b), as during simulations that was the worst test situation. In fact, neither a pure PFM nor the proposed scheme without training were able to move through such a corridor. However, after any training the proposed scheme provided a trajectory through the corridor. Initially, the walls of the corridor were separated approximately 1 m. In this case, the corridor was recalled by the robot as a combination of situations with two and three obstacles during training in simulations. It can be observed in Fig. 17(a) that the robot moves through the corridor relying uniquely on learnt *cases*. However, after we moved the obstacles closer to achieve a narrower corridor, it can be observed in Fig. 17(b) that now all *cases* need to be mutated to suit the new situation. Furthermore, if *cases* learnt during the experiment in Fig. 17(b) are added to the *casebase*, no further mutations are needed

for the same corridor (Fig. 17(c)). Performing the same experiment with a pure PFM or the proposed scheme without training resulted in the robot not entering the corridor at all.

These tests are interesting for several important reasons: (i) they prove that spurious readings and noise are filtered out of the input data by the proposed purely reactive scheme as commented in previous sections; (ii) they also prove that training can be adapted to the real physical situation of the robot, which is not fully covered by a simulator; and (iii) if the robot can be trained under simulation and then operate in real environments, it is possible to provide as much training as necessary to deal efficiently with many different situations with minor efforts.

4.4. Hybridization

The main drawback of the proposed system is that, as all purely reactive schemes, it is prone to fall in local traps. However, this problem can be easily overcome by adding a high level planner to build a hybrid system. A final test consisted of replacing the PFM based reactive layer in the hybrid architecture proposed by the authors in Urdiales et al. (2003) with the new CBR based one. Figures 18(a) and (c) show respectively how PFM fall into a classic "U" trap and how the problem is solved by the hybrid system, whereas Figs. 18(b) and (d) show the same experiment using the proposed CBR based system. It can be observed that both approaches behave similarly, even though PFM tend to behave in a more conservative way when the robot is close to obstacles as reported. This experiment proves that the proposed scheme can be easily integrated in a hybrid architecture, keeping all aforementioned advantages.

5. Conclusions

This paper has presented a new CBR based purely reactive system for an autonomous mobile platform. The proposed system only uses the on-board sonar sensors readings, the robot heading and the goal direction to choose particular actions which, in combination, return an emergent reactive behaviour. The main advantages of this scheme are that it does not depend on any model of the environment. It only acquires local information to store sense/act patterns, taking implicitly into account the sensor nature and robot structure. Hence, it is valid for any place, and it can adapt to any robotic architecture, sensor layout or environment conditions through learning.

The proposed system supports both learning by observation and own experience. Supervised training provides faster convergence to efficient trajectories, but is not required. *Cases* can be learnt from a human driver or from any available reactive algorithm and also from the experiences of the platform while following a path. All *cases* are ranked to provide smooth, short and safe trajectories so that the *casebase* can be easily reorganized to return the best possible *case* and to keep a bounded size. Learning through CBR is in this *case* better than other approaches like Artificial Neural Networks because CBR provides lazy generalization from a potentially small number of samples.

The main advantage of learning is that the algorithm easily adapts itself to changes. In fact, unlike in most popular reactive approaches, no kinematics or dynamics need to be taken into account to develop the algorithm. Kinematics are simply absorbed in the *casebase* and dynamics are taken into account by the repetitive nature of reactive control: the more complex the environment, the slower the robot moves, as retrieved cases change rapidly. It has been tested that the longer the robot operates, the better it behaves because it is more and more adapted to its environment. Nevertheless, if the platform or environment is changed, the reactive algorithm adapts itself again to the new situation: this has been tested by training the algorithm under simulation and testing it in a real environment.

The proposed system has been successfully tested using a Pioneer robot equipped with 8 Polaroid sonar sensors. The main advantages of the system are: (i) it is fast enough to operate in a continuous way; (ii) it converges to non-oscillating trajectories; (iii) it provides short and smooth paths to the goal; (iv) it behaves correctly even in narrow corridors; and (v) it can adapt itself to new, unexpected situations. It has also been tested that the system acquires knowledge in an efficient way both in a supervised and non-supervised way and efficiently combines this knowledge to reach its goal.

The proposed system only depends on two thresholds: U_{learn} and CS_{max}. It is important to observe that, unlike in other reactive approaches, these parameters do not depend on the robot nature or environment features. U_{learn} determines how different a situation must be from learnt cases to consider it a new one and, hence, how many cases are acquired through learning. In order to reorganize those cases, CS_{max} fixes how many cases remain in the *casebase* and how significant they are. The system performance is not too dependent on those thresholds: if the *casebase* has many cases, the robot clones better the learnt trajectories whereas if there are few cases stored, the system generalizes more and more adaptation is required. Nevertheless, we have experimentally checked that the robot works correctly for a wide range of parameter values.

It can be noted that in all presented tests only a few obstacles were placed in the environment. However, since the proposed CBR scheme is purely reactive, it would deal with more obstacles by simply decomposing the problem into a set of simpler ones and adapting itself to any local change. The main drawback of the proposed system is that, being

purely reactive, it may fall into local traps, but it can be solved by adding a high level planner to the system.

Acknowledgments This work has been partially supported by the Spanish Ministerio de Ciencia y Tecnologia (MCYT) and FEDER funds, project No. TIN2004-07741.

References

Aamodt, A. 1994. Explanation-driven case-based reasoning. In S. Wess, K. Althoff, and M. Richter (eds.), *Topics in Case-Based Reasoning*, Springer Verlag, pp. 274–288.

Arkin, R.C. 1995. Reactive robotic systems. In M. Arbib (ed.), *Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 793–796.

Arkin, R.C. 1998. *Behaviour based robotics*. MIT Press: Cambridge.

Branting, L.K. and Aha, D.W. 1995. Stratified case-based reasoning: Reusing hierarchical problem solving episodes. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence*, Montreal, Canada, pp. 384–390.

Brock, O. and Khatib, O. 1999. High-speed navigation using the global dynamic window approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 341–346.

Brock, O. and Khatib, O. 2002. Elastic strips: A framework for motion generation in human environments. *International Journal of Robotics Research*, 21(12):1031–1052.

Deichsel, G. and Trampisch, H.J. 1985. *Clusteranalyse und Diskriminanzanal-yse*, Gustav Fischer, Verlag, Stuttgart.

Dougherty, J., Kohavi, R., and Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In *Proc. of the 12th Int. Conf. on Machine Learning*, pp. 194–202.

Fox, S. and Leake, D.B. 1995. Combining case-based planning and introspective reasoning. In *Proc. of the 6th Midwest Artificial Intelligence and Cognitive Science Society Conference*, Carbondale, IL, pp. 32–36.

Fox, D., Burgard, W., and Thrun, S. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33.

Fox, D., Burgard, W., Thrun, S., and Cremers, A.B. 1998. A hybrid collision avoidance method for mobile robots. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1238–1243.

Green, D.N., Sasiadek, J.Z., and Vukovich, G.S. 1994. Path tracking, obstacle avoidance, and position estimation by an autonomous, wheeled planetary rover. In *Proc. of the IEEE International Conference on Robotics and Automation*, San Diego, California (USA), pp. 1300–1305.

Haigh, K.Z. and Veloso, M. 1995. Route planning by analogy. In *Proc. of Int. Conf. on Case-Based Reasoning*, Springer Verlag, Berlin, pp. 160–180.

Hammond, K.J. 1989. *Case-Based Planning: Viewing Planning as a Memory Task*, Academic Press: Boston, MA.

Khatib, O. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98.

Kruusmaa, M. 2003. Global navigation in dynamic environments using case-based reasoning. *Autonomous Robots*, 14:71–91.

Likhachev, M. and Arkini, R.C. 2001. Spatio-temporal case-based reasoning for behavioral selection. In *Proc. of the IEEE Int. Conf. on Robotics and Automation* (ICRA), Seoul, Korea, pp. 1627–1634.

Minguez, J. and Montano, L. 2003. The ego-kinoDynamic space: Collision avoidance for any shape mobile robot with kinematic and dynamic constraints. In *Proc. of 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'03)*, Las Vegas, Nevada, USA, pp. 637–643.

Moravec, H.P. 1998. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9:61–74.

Quinlan, S. and Khatib, O. 1993. Elastic Bands: Connecting path planning and control. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, Atlanta, USA, pp. 802–807.

Ram, A. and Santamaria, J.C. 1993. A multistrategy case-based and reinforcement learning approach to self-improving reactive control systems for autonomous robotic navigation. In *Proc. of the 2nd Int. Workshop on Multistrategy Learning*, Harpers Ferry, West Virginia, pp. 259–275.

Sànchez-Marrè, M., Cortés, U., Béjar, J., Roda, I.R., and Poch, M. 1999. Reflective reasoning in a case-based reasoning agent. Lecture Notes in Artifical Intelligence 1624, Springer-Verlag, pp. 143–158.

Schank, R. 1982. *Dynamic Memory*. Cambridge University Press: New York.

Schank, R. and Abelson, R. 1977. *Scripts, Plans, Goals and Understanding*: Lawrence Erlbaum.

Sekhavat, S. and Chyba, M. 1999. Nonholonomic deformation of a potential field for motion planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI (US), pp. 817–823.

Simmons, R. 1996. The curvature-velocity method for local obstacle avoidance. In *Proc. of Int. Conf. on Robotics and Automation*, 3:2737–2742.

Ulrich, I. and Borenstein, J. 2000. VFH: Local obstacle avoidance with look-ahead verification. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, pp. 2505–2511.

Urdiales, C., Bandera, A., Pérez, E.J., Poncela, A., and Sandoval, F. 2003. Hierarchical planning in a mobile robot for map learning and navigation. In D. Maravall, D. Ruan, and C. Zhou (eds.), *Autonomous Robotic Systems—Soft Computing and Hard Computing Methodologies and Applications*, Physica Verlag, pp. 165–188.

van Lent, M. and Laird, J. 1998. Learning by observation in a complex domain. In *11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Banff, Alberta, Canada, pp. 18–23.

Veloso, M. and Carbonell, J.G. 1993. Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning*, 10(3):249–278.

Watson, I. 1996. An introduction to case-based reasoning. *Progress in Case-Based Reasoning*, Lecture Notes in Artificial Intelligence (LNAI), 1020, pp. 3–16.

Cristina Urdiales is a Lecturer at the Department of Tecnología Electrónica (DTE) of the University of Málaga (UMA). She received a MSc degree in Telecommunication Engineering at the Universidad Politécnica de Madrid (UPM) and her Ph.D. degree at University of Málaga (UMA). Her research is focused on robotics and computer vision.

E.J. Pérez was born in Barcelona, Spain, in 1974. He received his title of Telecommunication Engineering from the University of Málaga, Spain, in 1999. During 1999 he worked in a research project under a grant by the Spanish CYCIT. From 2000 to the present day he has worked as Assistant Professor in the Department of Tecnología Electrónica of the University of Málaga. His research is focused on robotics and artificial vision.

Javier Vázquez-Salceda is an Associate Researcher of the Artificial Intelligence Section of the Software Department (LSI), at the Technical University of Catalonia (UPC). Javier obtained an MSc degree in Computer Science at UPC. After his master studies he became research assistant in the KEMLg Group at UPC. In 2003 he presented his Ph.D. dissertation (with honours), which has been awarded with the 2003 ECCAI Artificial Intelligence Dissertation Award. The dissertation has been also recently published as a book by Birkhauser-Verlag. From 2003 to 2005 he was researcher in the Intelligent Systems Group at Utrecht University. Currently he is again member of the KEMLg Group at UPC. His research is focused on theoretical and applied issues of Normative Systems, software and physical agents' autonomy and social control, especially in distributed applications for complex domains such as eCommerce or Medicine.

Miquel Sànchez-Marrè (Barcelona, 1964) received a Ph.D. in Computer Science in 1996 from the Technical University of Catalonia (UPC). He is Associate Professor in the Computer Software Department (LSI) of the UPC since 1990 (tenure 1996). He was the head of the Artificial Intelligence section of LSI (1997–2000). He is a pioneer member of International Environmental Modelling and Software Society (IEMSS) and a board member of IEMSS also, since 2000. He is a member of the Editorial Board of International Journal of Applied Intelligence, since October 2001. Since October 2004 he is Associate Editor of Environmental Modelling and Software journal. His main research topics are case-based reasoning, machine learning, knowledge acquisition and data mining, knowledge engineering, intelligent decision-support systems, and integrated AI architectures. He has an special interest on the application of AI techniques to Environmental Decision Support Systems.

Francisco Sandoval was born in Spain in 1947. He received the title of Telecommunication Engineering and Ph.D. degree from the Technical University of Madrid, Spain, in 1972 and 1980, respectively. From 1972 to 1989 he was engaged in teaching and research in the fields of opto-electronics and integrated circuits in the Universidad Politécnica de Madrid (UPM) as an Assistant Professor and a Lecturer successively. In 1990 he joined the University of Málaga as Full Professor in the Department of Tecnología Electrónica. He is currently involved in autonomous systems and foveal vision, application of Artificial Neural Networks to Energy Management Systems, and in Broad Band and Multimedia Communication.