CHAPTER 7

Traversability and *Tenacity*: two new concepts improving the navigation capabilities of reactive control systems

J. ANTICH and A. ORTIZ

Department of Mathematics and Computer Science
University of the Balearic Islands
Cra. Valldemossa, km 7.5 - 07122 Palma de Mallorca (Spain)
{javi.antich,alberto.ortiz}@uib.es

Reactive mobile robot navigation based on potential field methods has shown to be a good solution for dealing with unknown and dynamic scenarios, where timely responses are required. Unfortunately, the complexity of the tasks which can successfully be carried out is limited by the inherent shortcomings of the approach such as trapping situations due to local minima, difficulties passing among closely spaced obstacles, oscillations in narrow corridors, etc... This chapter outlines a set of strategies which overcome the first of the aforementioned limitations by computing an adaptive navigation function on the basis of such artificial potential fields. As a result, navigation can be achieved in very difficult underwater (and land) scenarios where obstacles adopt maze-like configurations. A comparative study on the path length performance of our proposal with regard to other algorithms from the related literature is also presented.

1 Introduction

In robotic navigation, potential field methods (PFM) (Khatib, 1986) are a well-known solution for dealing with unknown and dynamic scenarios such as the submarine by taking into account the reality of the environment during the robot motion. The characteristic elegance and simplicity of the approach when representing and successfully solving a path-planning

problem in real-time explains its extensive application in this field. However, substantial shortcomings have been identified as problems inherent to this principle (Koren and Borenstein, 1991). *Getting stuck in local minima* is the best-known and most often-cited problem with PFMs. As a result, several obstacle configurations such as the typical U-shaped canyon may lead to undesirable trapping situations. Another drawback associated with this kind of systems is related to the lack of an oscillation-free motion when the robot navigates among very close obstacles at a high speed. Finally, the impossibility to go through small openings constitutes the last significant problem of PFMs.

In this context, the contribution of this work is twofold:

- On the one hand, a solution to the local minima problem is given according to the so-called concepts of *Traversability* and *Tenacity* —or T^2, in brief. As a result, navigation is achieved in very difficult scenarios, even including maze-like environments. Unfortunately, the completion of the mission cannot be always guaranteed due to the appearance of cyclic-oscillatory behaviours.
- On the other hand, three different algorithms, based on the T^2 principles, are also put forward to ensure, whenever possible, the attainment of any navigation task.

The rest of the chapter is organized as follows: section 2 introduces the basic framework under which the new family of navigation strategies, generically called T^2, will be defined, while sections 3 and 4 provide a detailed description of it; a comparative study on the path length performance of the proposal is presented in section 5; and, finally, some conclusions are given in section 6.

2 Framework

The classic potential field approach proposed in (Khatib, 1986) constitutes the basic framework for the application of the novel T^2 family of navigation strategies. It computes the motion of the robot on the basis of two simple behaviours: *GoTo* and *AvoidObstacles*. More precisely, the former generates an attractive force in direction to the goal, while the latter considers obstacles as repulsive surfaces. The robot follows the negative gradient of the resulting potential field towards its minimum, whose position is expected to coincide with the goal point.

3 Fundamentals of the T^2 Navigational Approach

The inability to move the robot away from the goal direction in a non-momentary and strategic way is the main cause of the undesirable trapping situations suffered by the reactive control paradigm and, in particular, by the artificial potential fields. Fig. 1(a) shows an example where a robot adopting this last approach is unable to escape from a U-shaped obstacle. In the following, a solution will be given by applying two concepts, *Traversability* and *Tenacity* —or T^2, in brief—, in the context of the so-called navigation filter. As a result, those trapping situations linked to the local minima problem will be successfully avoided. However, it will be also seen that this module does not always ensure the completion of the mission. In the next section, several changes will be carried out in order to guarantee the convergence. Finally, the reader should notice that the navigation filter is not an isolated control module but a new component of a generic behaviour-based control system. Fig. 1(b) illustrates its integration into the classic potential field approach. The resulting control diagram will be taken as a reference from now on.

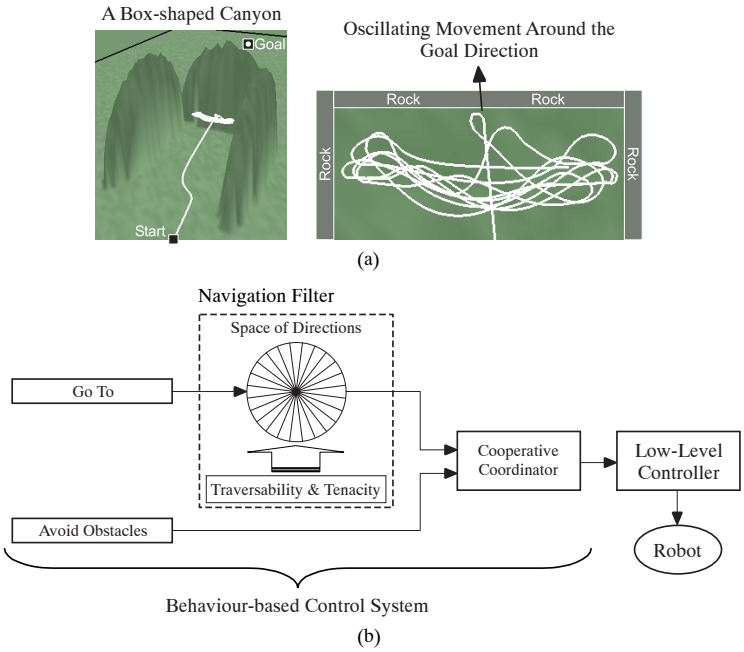

Fig. 1. (a) A typical trapping situation for reactive control systems; (b) integration of the navigation filter into the classic potential field approach.

3.1 The Navigation Filter

The main task of the navigation filter is the appropriate alteration of the direction of the motion vector generated by the *GoTo* behaviour in order to rapidly overcome any obstacle irrespective of its size and shape (see fig. 1(b) again). Such change is carried out according to the *Traversability* and *Tenacity* principles. In short, the first one suggests, on the one hand, banning those directions where an obstacle has been detected and, secondly, choosing an obstacle-free direction close to the desired one — *GoTo*'s response— when it has been banned. In this last case, the *tenacity* principle determines, finally, what obstacle-free direction will be selected among all the available alternatives. Next, both principles will be treated in depth.

3.1.1 The Traversability Principle

The application of the *traversability* principle requires the division of the space of directions around the robot into K identical angular regions as it is shown in fig. 2(a). These regions can be additionally classified as *banned* or *allowed*. Specifically, a region will belong to the former group when at least one obstacle is known to be in the range of directions which consists of. The non-existence of obstacles, on the other hand, characterises the latter.

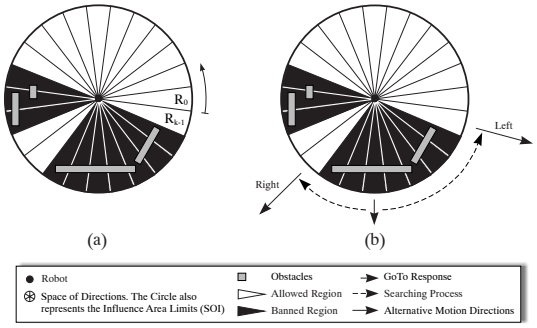

Fig. 2. Exemplifying the implementation of the *traversability* principle: (a) division of the space of directions into K regions, labeling them as *allowed* or *banned*; (b) selection of two obstacle-free motion directions.

Based on the previous information, this principle is intended to forbid the robot's movement in directions where the presence of obstacles has

recently been determined, avoiding thus unnecessary and unsuccessful displacements in the task of looking for a path towards the goal point. With this purpose, after receiving the response of the *GoTo* behaviour, its viability is studied according to the above-mentioned premise. Changes are required only if the direction of such motion vector lies in a banned region. Otherwise, the navigation filter will not act, generating as output the same input vector. In such a case, note that the classic potential field approach is really used to produce the response of the control system. Assuming the first situation which has been described, two alternative directions, generically labeled as left and right, will be obtained as a result of a double searching process, clockwise and counterclockwise, for the first allowed region starting from the desired direction of motion (see fig. 2(b) for an example). The final decision about choosing one direction or another depends on the *tenacity* principle.

3.1.2 The Tenacity Principle

Taking fig. 1(a) as an example, the trajectory of a robot which has been trapped can be seen, in a simplified way, as the result of two simple movements: forward and backward. These movements alternate the control of the robot each time the vehicle moves excessively away from the goal direction. The *tenacity* principle precisely tries to give a solution to that oscillating/hesitant behaviour by preventing the robot from drastically changing its direction of motion. In this way, progress will be always ensured removing thus the main cause of any trapping situation. Regarding the implementation details, remember that two alternative motion directions labeled as left and right are obtained when the response of the *GoTo* behaviour lies in a banned region. Under these circumstances, one of such directions has to be selected as output of the navigation filter. With this aim, the *tenacity* principle is applied by choosing left or right in coincidence with the last decision made. Finally, note that, despite the obvious simplicity of the concept, this principle has proven fully effective.

3.1.3 Need for Remembering the Obstacles

Purely reactive systems such as the classic potential field approach react directly to the world as it is sensed, avoiding the need for intervening any kind of abstract representational knowledge. The sentence "what you see is what you get" faithfully summarises this idea. However, the local information provided by the robot's sensory equipment may not be enough to

solve a given navigation task. Our navigation filter, against this problem, keeps and uses information regarding the obstacles beyond the robot's immediate sensory range. The approach is, nevertheless, still reactive since no path planning is ever conducted on such information. By way of example, fig. 3 shows how the navigation filter is able to remember the presence of obstacles in directions where, currently, an obstacle-free space is being locally detected. To this end, the approximate location of the obstacles in the environment is memorised. It is important to note, however, that the character of these data is temporary, being removed when the robot is sure that the corresponding obstacle has been successfully overcome. Further details about this subject will be given later.

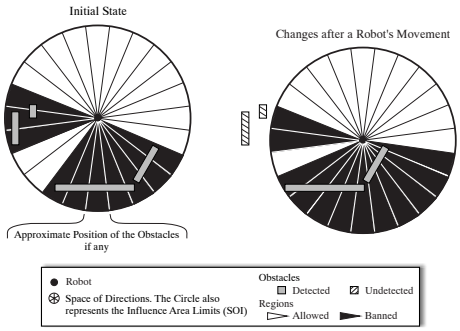

Fig. 3. Memorising the obstacle locations while navigating.

3.2 The Emergent Global Behaviour

The combined application of both the *traversability* and *tenacity* principles results in an emergent global behaviour which can be summarised in the next three points (see fig. 4 for a simple example):

1. When the robot is navigating far from obstacles, it heads for the goal following a straight path. During this period of time, the navigation filter remains inactive generating as output the same input motion vector.
2. After the detection of an obstacle, the robot follows its contour in a certain direction. To this end, changes are required on the response of the *GoTo* behaviour based on T^2. Of special interest is, however, the first time that a change has to be performed. In such a case, the *tenacity* principle cannot be applied due to the lack of a previous

decision, so that an additional selection criterion is needed for these particular situations. This first decision will determine the direction taken by the robot when following the obstacle boundary. In short, the decision is made on the basis of how the previous obstacle was followed, or lacking this information —first obstacle—, according to a minimum turn criterion.

3. Finally, the robot knows that the obstacle has been overcome when the direction to the goal becomes free of obstacles, that is, not banned. At that moment, the filter is reset losing thus all the previously kept information —obstacle positions—, which is no longer necessary for the navigation task.

These stages will be sequentially executed in the order specified so many times as obstacles the robot finds on its way towards the goal point. To finish, note that the *AvoidObstacles* behaviour also plays an important role on the final robot's behaviour. More precisely, it helps to maintain a reasonable distance between the robot and the contour of an obstacle when it is being followed.

3.3 Major Shortcoming

The successful mission completion cannot be always guaranteed by means of, only, the navigation filter —that is, the T^2 principles. Fig. 5 shows, precisely, an example confirming this fact. As can be observed, the robot was not able to reach the goal point by generating a cyclic behaviour around a G-shaped obstacle. In the next section, three different solutions to this problem will be given. They slightly modify the way how the robot decides to leave and follow, in one direction or another, the boundary of an obstacle.

4 Three T^2-based Algorithms with Provable Guarantees

This section constitutes a continuation of the previous one, where a new component named navigation filter was incorporated into a control system based on the potential fields approach to avoid the known and, up till then, unsolved local minima problem. In this sense, at present, it is intended to step forward by ensuring, whenever possible, the achievement of the goal point for any mission. To this end, three different algorithms will be proposed which alter the way how such navigation filter takes two kinds of decisions related to: on the one hand, the direction for following

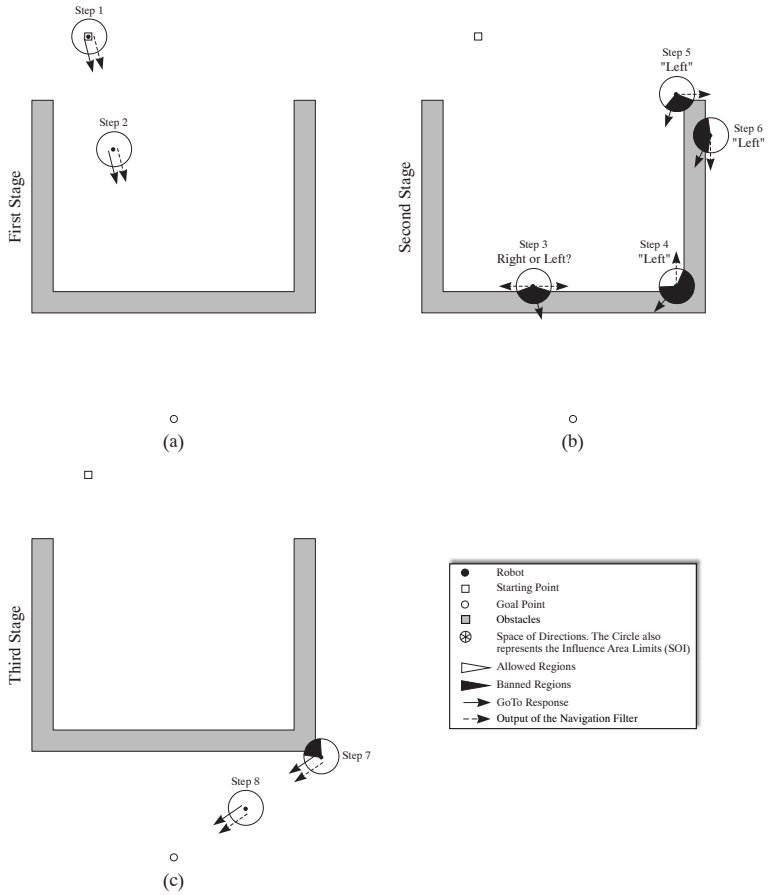

Fig. 4. Escaping from a U-shaped obstacle: (a) direct path to the goal point; (b) following the contour of the obstacle to the robot's left (note that this direction is defined in step 3 by selecting the option labeled as left based on a minimum turn criterion); (c) reset of the navigation filter returning, afterwards, to the same situation as in (a) from which the goal is finally reached.

the contour of an obstacle and, on the other hand, the leaving of such contour for trying, afterwards, to definitely reach the goal through a straight path (see fig. 6 for a graphic illustration of both types of decisions). These algorithms are specifically called *Random* T^2, *Connectivity* T^2 and *Bug-based* T^2, representing all of them the new family of algorithms T^2. Next, each of the members of such family will be briefly described (the reader is referred to (Antich, 2006) for further details).

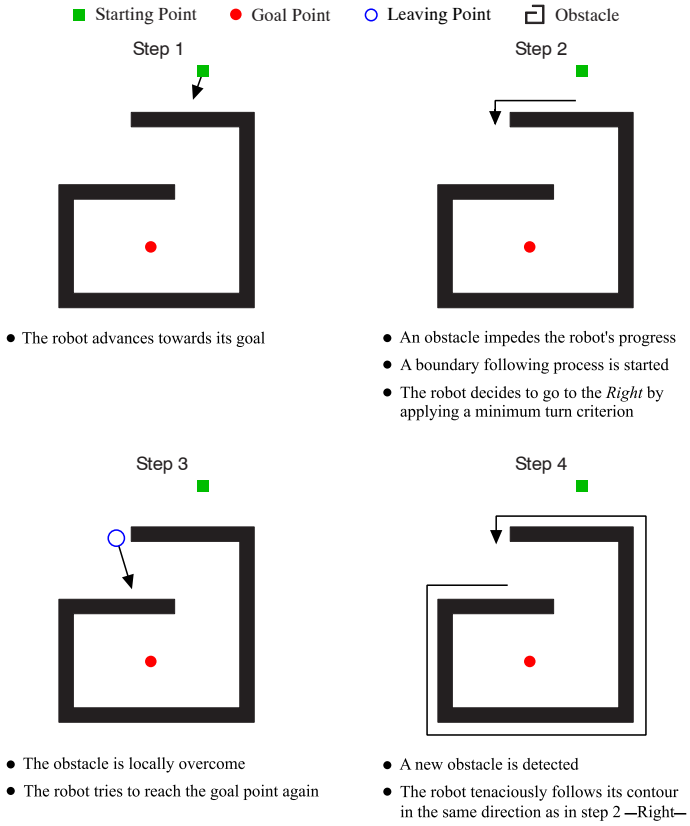

Fig. 5. A mission not solved by the navigation filter. Note that after step 4, the third step will be executed again, generating thus a cyclic behaviour. The robot's trajectory has been computed by hand according to the known three stages of the strategy (refer to section 3.2).

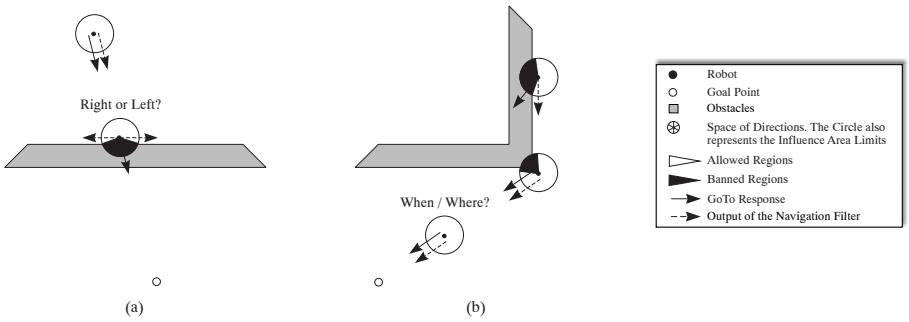

Fig. 6. Decisions having influence on the convergence —goal achievement guarantee— of an algorithm based on the T^2 principles: (a) selection of the contour following direction of the obstacles; (b) leaving the obstacle boundaries.

4.1 The Strategy *Random* T^2

Random T^2 is the simplest strategy of the family of algorithms being proposed. In this sense, non-extra data and reasoning are needed apart from the ones associated with the navigation filter to make the decisions pointed out in fig. 6. More precisely, these are the specific criteria applied by the strategy regarding such decisions:

- In the context of T^2, the completion of a mission may be fully conditional on the following of the contour of some obstacles in certain directions —left or right— depending on the particular features of the navigation environment. These features are unknown by the robot in accordance with its reactive nature, concluding thus that it will never be possible to properly define a fixed beforehand criterion for choosing the contour following direction which is valid for any mission without exception. Note that it was precisely the mistake made by the basic T^2 approach presented in section 3 where the same direction was always taken. *Random* T^2, keeping the previous considerations in mind, adopts the simplest —and also less efficient— solution to the problem by selecting that direction in a random way. In this manner, if an obstacle is followed in the wrong direction, the robot will be able to unconsciously/probabilistically rectify its decision in subsequent occasions reaching, finally, the goal point.
- On the other hand, as for the leaving of the contour of the obstacles, no changes have been performed with regard to the original T^2 approach. In this way, the boundary of an obstacle will be essentially left when the direction to the goal point becomes *allowed* —free of obstacles. Note that, from a geometric point of view, such circumstance exclusively arises in situations where the robot faces the goal while following the obstacle contour.

Finally, by way of example, fig. 7 shows how the strategy would suitably accomplish the mission presented in fig. 5 which was not solved by the navigation filter alone.

4.2 The Strategy *Connectivity* T^2

The strategy *Connectivity* T^2 is intended to improve the efficiency of the previous algorithm in the search of a path towards the goal point avoiding that, fundamentally, parts of the environment —obstacle boundaries— which have already been explored by the robot in an unsuccessful way

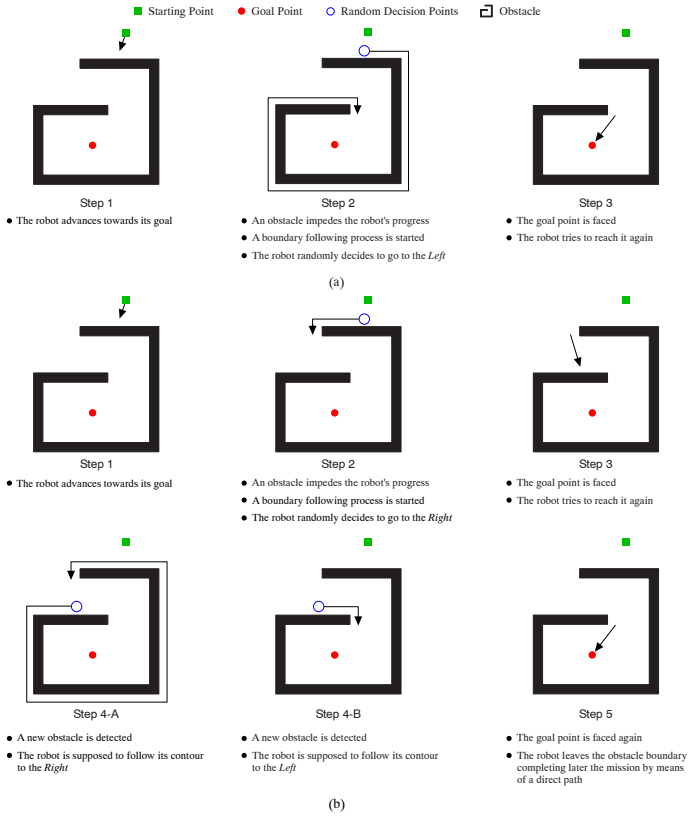

Fig. 7. Exemplifying the behaviour of the strategy *Random* T^2 in a scenario with a G-shaped obstacle. (a) and (b) are the two paths which result from considering all the possible decisions of the algorithm with regard to the selection of the obstacle contour following directions. The situation shown in (b) requires, however, an additional comment. Either step 4-A or step 4-B will be randomly chosen to be carried out by the robot. In the former case, step 3 will be afterwards executed again. In the latter, on the contrary, the control flow will go to step 5 where the goal point will be finally achieved.

are covered again (remember fig. 7 where step 4-A might be executed more than once). To this end, the concept of *key area* is introduced into the original T^2 approach. Specifically, it is essentially an artificial landmark virtually located next to an obstacle that indicates in what directions —left and/or right— the contour of such obstacle has been followed starting from that precise position. Fig. 8 illustrates the mentioned concept in a simple mission. As can be observed, each time the robot detects a new obstacle and, in consequence, has to decide to follow its contour in

one direction or another, a key area is created for registering the decision taken. Note that this information, in more complex scenarios such as that of fig. 7, will allow the robot not to repeat decisions on the same obstacle favouring thus an extensive and fast exploration of the navigation environment. On the other hand, as for the leaving of the obstacle boundaries, just like it happened in the algorithm *Random* T^2, no changes have been carried out on the proposal of section 3.

Finally, fig. 9 shows an example of the application of the strategy.

4.3 The Strategy *Bug-based* T^2

Based on the T^2 principles, the last strategy which is proposed is called *Bug-based* T^2. Specifically, this strategy ensures convergence —that is, the mission completion whenever possible— by slightly modifying the way how the navigation filter decided to leave the contour of the obstacles.

As was already explained in section 3.1, the navigation filter makes the robot abandon the boundary of an obstacle when, essentially, the direction to the goal becomes *allowed* —free of obstacles. It is important to note that, from a geometric point of view, such circumstance exclusively arises in situations where the robot faces the goal point while following the obstacle contour. These points of leaving, taking into account the local information which is really managed by the strategy, correspond to the most promising and closest places from where the robot would be supposedly capable of completing the mission through a direct path. This advantage, however, contrasts with the lack of convergence which is derived from such decisions (remember fig. 5).

Regarding the leaving of the contour following process, other interesting criteria can be found in the related literature. More precisely, the strategy *Bug2* (Lumelsky and Stepanov, 1987) suggests that the robot stops following the boundary of an obstacle when its trajectory cuts the straight segment joining the starting and the target points, also named Main Line —or M-line, in brief. As a result, this strategy guarantees the goal achievement at the expense of, generally, longer paths. Fig. 10(a) and (b) compare the conditions for leaving applied by respectively the T^2 and *Bug2* approaches, from the viewpoint of the resultant robot's trajectory in a simple scenario.

Both advantages, a good path length performance as well as convergence, can be really attained by combining and slightly altering the previous conditions in a proper way. On the one hand, the leaving conditions for T^2 (C1) and *Bug2* (C2) are defined as follows:

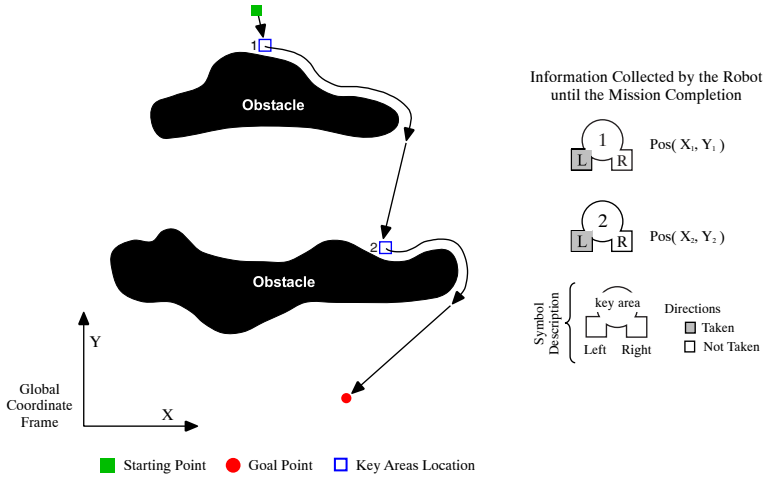

Fig. 8. The *key area* concept.

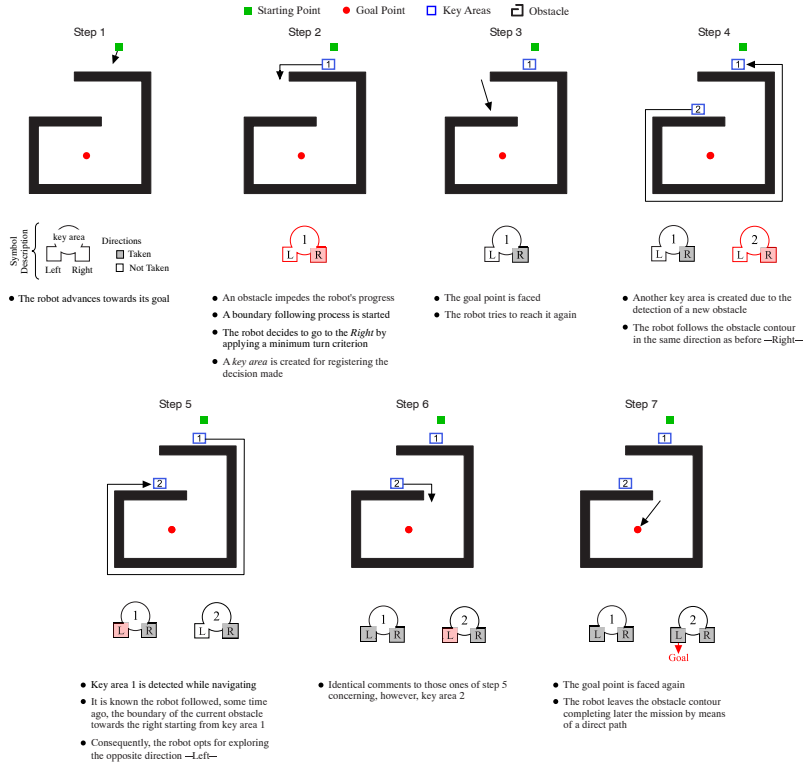

Fig. 9. The strategy *Connectivity* T^2 solving a mission with a G-shaped obstacle. Note that the key area information kept by the robot is also represented in the figure for each step being highlighted in red the latest changes.

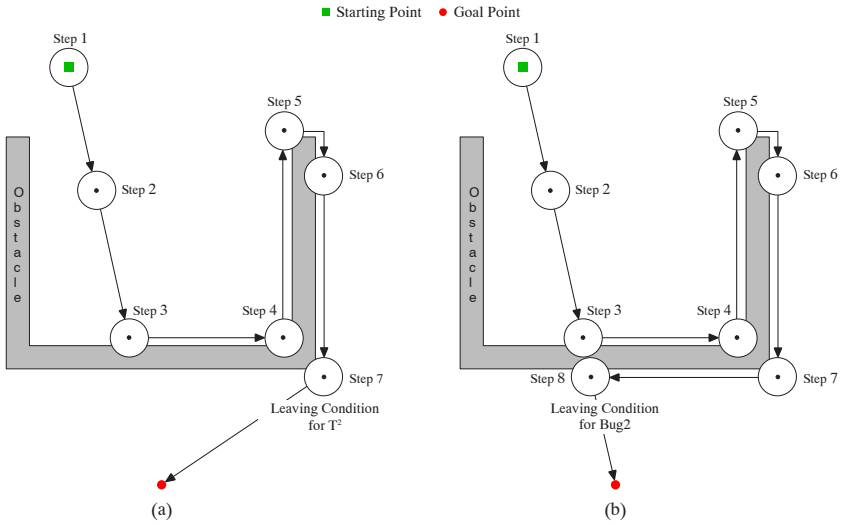

Fig. 10. Comparing the leaving conditions linked to the T^2 (a) and *Bug2* (b) strategies.

C1) The navigation filter indicates there is a free-obstacle path towards the goal point. Moreover, this must be the first time the robot leaves the obstacle at approximately that same position.

C2) The robot's trajectory cuts the M-line. Additionally, the distance from the robot to the goal has to be shorter than the one associated with the last time this condition was satisfied. Initially, this distance is initialised to the M-line's length.

On the other hand, either C1 or C2 must be met for the leaving to occur. At this point, notice that, contrary to the *Bug2* algorithm, the M-line concept is not static for *Bug-based* T^2. More precisely, each time the leaving condition C1 is satisfied, the M-line is modified by considering as new starting point the current robot position[1]. This fact together with the limited number of times that such condition can be fulfilled[2] allow proving the convergence of *Bug-based* T^2. The formal proof can be found in (Antich, 2006) together with some simulated and real experiments.

[1] In such a case, the distance used by condition C2 is also reinitialised

[2] This assertion is based on the assumption that there is a finite number of obstacles in the navigation environment

5 A Comparative Study

In the following, a comparative study on the performance of the new set of navigation strategies —*Random* T^2, *Connectivity* T^2 and *Bug-based* T^2 to be precise—, generically called T^2, will be presented and discussed in detail. Four algorithms from the related literature have been considered. Among them, there are some popular approaches while the others have been recently published. On the other hand, the comparison is carried out from the point of view of a single criterion which is the length of the path generated between the starting and the goal points.

5.1 Strategies Considered

Several strategies have been selected to take part in the comparative study against the three proposed navigation algorithms. Their main features are summarised next:

- *Avoiding the Past* (Balch and Arkin, 1993). The robot moves to the user-specified goal point while being repelled from locations which were already visited. With this purpose, a local map of the environment implemented as a two-dimensional grid is stored in memory, where a different value is assigned to visited and non-visited locations. As the robot visits an area more times, the values of the corresponding cells in the grid increase and, consequently, the resultant repulsive force exerted by such cells increases as well. In this way, it is intended to favour the continuous exploration of new regions of the navigation environment avoiding thus, at least apparently, the robot gets stuck into a local minimum.
- *Learning Momentum* (LM) (Lee and Arkin, 2001). This strategy adjusts the behavioural parameters of a particular reactive control system at runtime instead of using static values. A module called *Adjuster* is responsible for this task. This module, based on recent experience and a set of heuristic rules, identifies when good progress to the goal is being made. According to this, the gains as well as other parameters of the three behaviours making up the control system —GoTo, AvoidObstacles and Noise— are properly altered.
- *Micronavigation* (μNAV) (Scalzo *et al.*, 2003). This approach tries to solve the problem of mobile robot navigation from a minimalist point of view by only using, as its author says, a handful of bytes. Specifically, the robot is provided with a hierarchy of simple behaviours

designed for smooth obstacle avoidance through the *equipotential line* concept and for escaping from concavities.

- *Bug2* (Lumelsky and Stepanov, 1987). This is a representative member of one of the most popular families of algorithms for path planning with incomplete information named *Bug*. The goal achievement guarantee whenever possible can be found among its main features. As for the strategy, two basic behaviours, GoTo and ContourFollowing, alternate the control of the robot. Initially, the GoTo behaviour is active. The detection of an obstacle, on the other hand, starts the contour following process. Such process is left by the robot when it cuts the virtual line connecting the starting and the goal points, also called *Main Line*. Finally, notice that some of the concepts introduced by this algorithm were suitably incorporated into our approach *Bug-based* T^2 to ensure its convergence (see section 4.3). This fact explains the inclusion of this strategy in the comparative study.

5.2 Results for a Representative Set of Missions

A 3D simulation environment named $NEMO_{CAT}$ (Antich and Ortiz, 2004) was used to measure the path length of the two first navigational approaches as well as T^2. This simulator incorporates the dynamic model of a real underwater vehicle called GARBI, designed and built by the Computer Vision and Robotics research group of the University of Girona (Spain), making thus the simulations more realistic.

Initially, three environments were defined for testing purposes (see fig. 11). In the first one, walls/rocks of different length impede the progress of the vehicle towards its goal. The second environment, on the other hand, corresponds to a very deep box-shaped canyon. Finally, the third one appeared in (Ranganathan and Koenig, 2003), where a control system with deliberative capabilities was employed to solve it. *Avoiding the Past* and *LM* strategies were not able to successfully carry out any of the previously described missions, which shows their poor effectiveness to escape from large trapping areas. In both cases, the simulation was stopped after a travel time twice the longest of T^2.

In order to continue with the comparative study, a robot programming environment based on the AuRA (Arkin and Balch, 1997) architecture called *MissionLab* (Mackenzie *et al.*, 1997) was also used. The latest release of this software (version 6.0) integrates the μNAV algorithm implemented by one of its authors (Sgorbissa, 2000). Different tests with increasing complexity were performed in *MissionLab*, simulating a

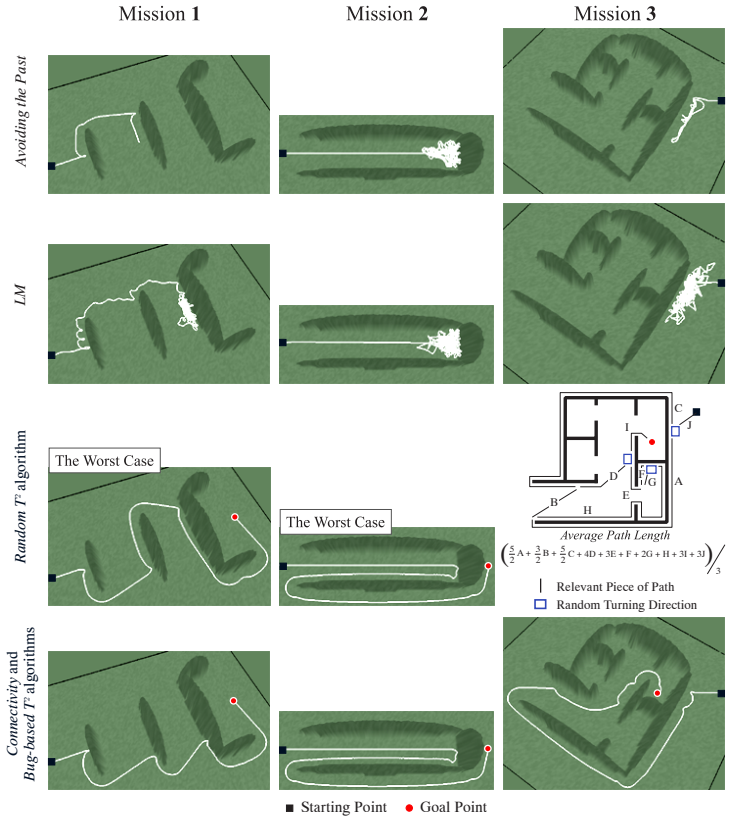

Fig. 11. From top to bottom, paths generated by the *Avoiding the Past*, *LM*, and T^2 approaches in $NEMO_{CAT}$. Note that due to the non-deterministic behavior of the Random T^2 algorithm, different results can be obtained in different runs of the simulator. Only one, the worst, is shown in the figure whenever possible. A stochastic analysis about the average length of the robot's path is presented when such worst case cannot be computed. For more details, the reader is referred to (Antich, 2006).

holonomic robot equipped with several range finders, and wheel encoders to compute its position by means of dead-reckoning. As can be verified in (Scalzo *et al.*, 2003), such experiments are a representative sample of the whole power of the μNAV strategy for a typical behavior hierarchy. Each environment was then accurately reproduced in $NEMO_{CAT}$ and successfully solved by T^2. The results from *MissionLab* are shown in fig. 12(a) while fig. 12(b) provides the ones from $NEMO_{CAT}$. Besides, tables 1 and 2 compare the performance of all these strategies from the viewpoint of the resultant path length. As can be observed, the T^2 algo-

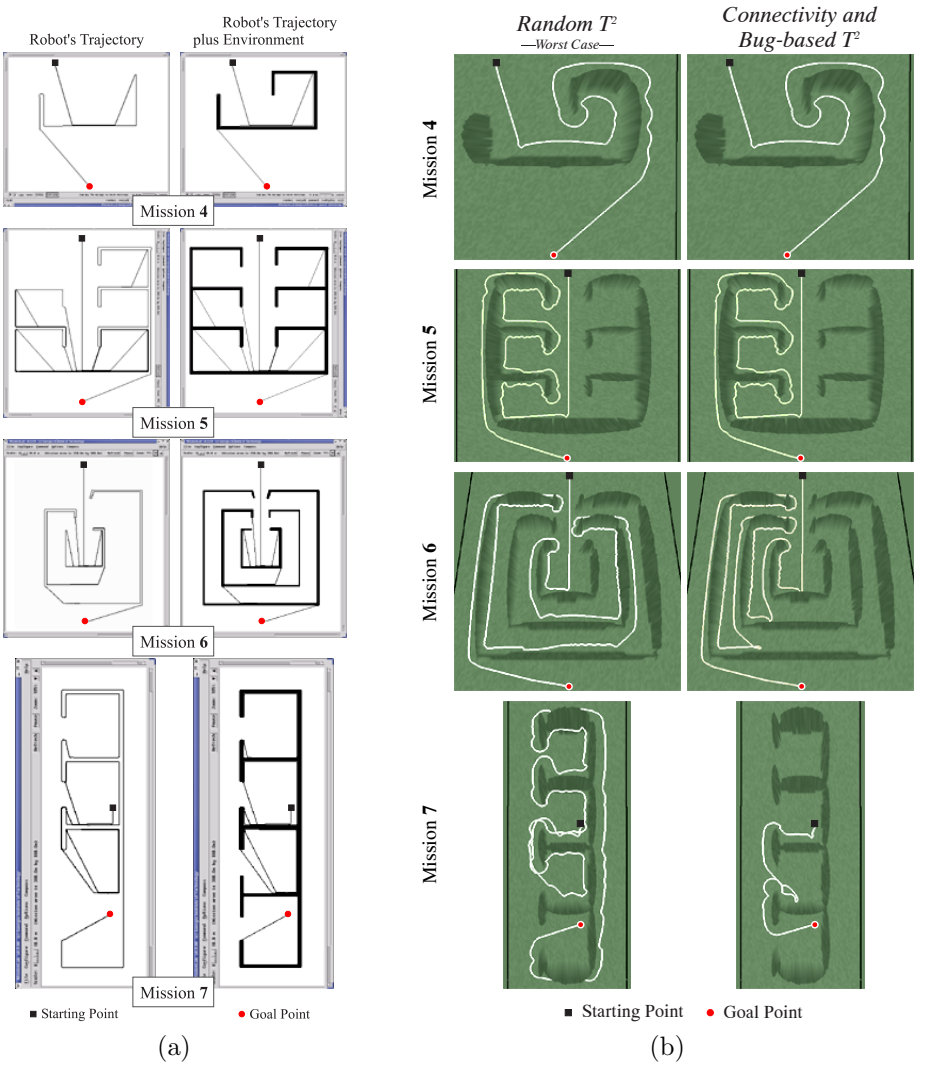

Fig. 12. Simulation results for the μNAV(a) and T^2(b) strategies in four different scenarios where concavities appear.

rithms produced, on average, trajectories between the starting and goal points 2.4 times shorter than μNAV at worst. The difference derives from the fact that μNAV allows the robot, in general, to head for the goal as soon as it is faced without any immediate obstacle on its way, while any T^2-based strategy limits the applicability of such rule to, at least, situations where a concavity is not detected.

Finally, the theoretical/ideal path of the *Bug2* algorithm for the missions previously outlined was drawn by hand according to the steps described in (Lumelsky and Stepanov, 1987) (see fig. 13). Tables 3 and 4 provide the comparative data with regard to our proposal. As can be seen, the *Bug2* algorithm generated, on average, trajectories between 1.14 and 1.48 times longer than T^2. On this occasion, the strict condition associated with the end of the contour following process is the key cause of the lower performance of such algorithm. Only when the main line is cut by the robot's path, the *GoTo* behavior becomes active to ensure convergence.

6 Conclusions

In this chapter, based on both artificial potential fields and two new principles named *Traversability* and *Tenacity* (T^2), a novel family of geometric algorithms for sensor-based motion planning has been finally put forward trying to ensure, whenever possible, the global achievement of the target point for any mission. The members of this family are specifically called *Random* T^2, *Connectivity* T^2 and *Bug-based* T^2. These strategies have also been compared against other well-known algorithms —*Avoiding the Past*, *Learning Momentum*, *Micronavigation* and *Bug2*— sharing the same goal. The length of the resulting paths was used as the figure of merit. Our proposals generated, on average, trajectories between 1.14 and 3.20 times shorter for a representative set of missions.

Finally, it is important to note that any T^2-based strategy can be applied to both ground and underwater vehicles, although, in this last case, a generalization to three dimensions, which is being developed at the moment, is expected to yield still better performance.

Acknowledgments

This study has been partially supported by project CICYTDPI2005-09001-C03-02 and FEDER funds. Its authors wish to thank the Georgia Tech Research Corporation for the free distribution of its software *MissionLab*.

Table 1. Comparison of the Path Lengths of T^2 and the μNAV Strategy

Mission	Algorithm Type		
	T^2		
	Random (average)	*Connectivity* and *Bug-based*	μNAV
4	286.83	368.55	384.20
5	883.47	883.47	4180.41
6	1586.71	1484.91	2489.40
7	663.82	246.85	1316.40
Total (m)	3420.83	2983.78	8370.41

Table 2. Relative Performance of T^2 with regard to the μNAV Strategy

Mission	$\frac{\mu\text{NAV}}{\text{Random } T^2}$	$\frac{\mu\text{NAV}}{\text{Connectivity } T^2}$, $\frac{\mu\text{NAV}}{\text{Bug-based } T^2}$
4	1.34	1.04
5	4.73	4.73
6	1.57	1.68
7	1.98	5.33
Average	2.40	3.20

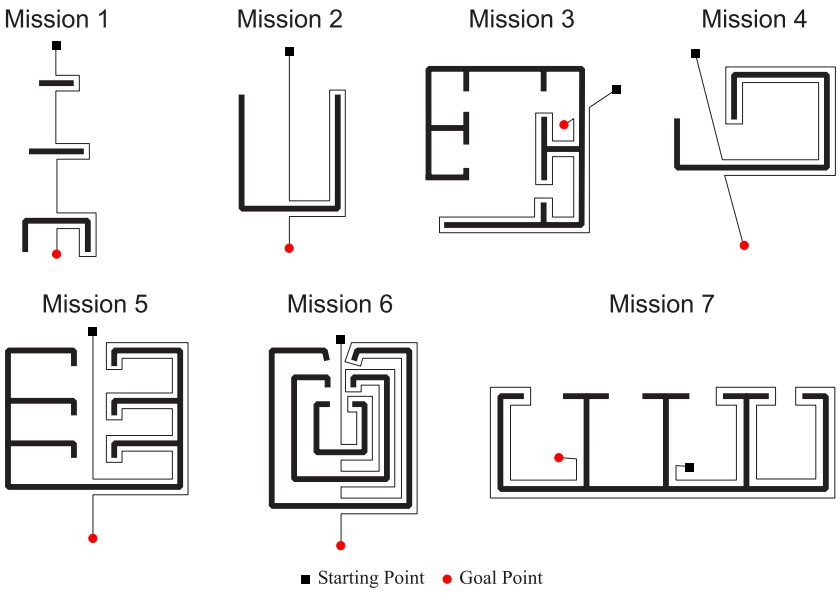

Fig. 13. Expected results for the *Bug2* algorithm from mission 1 to 7.

Table 3. Comparison of the Path Lengths of T^2 and the *Bug2* Strategy

Mission	Algorithm Type		
	T^2		
	Random (average)	*Connectivity* and *Bug-based*	*Bug2*
1	313.62	281.54	388.00
2	420.65	420.65	426.00
3	623.00	405.17	578.00
4	286.83	368.55	421.00
5	883.47	883.47	934.00
6	1586.71	1484.91	1672.00
7	663.82	246.85	797.00
Total (m)	4778.10	4091.14	5216.00

Table 4. Relative Performance of T^2 with regard to the *Bug2* Strategy

Mission	$\frac{\text{Bug2}}{\text{Random } T^2}$	$\frac{\text{Bug2}}{\text{Connectivity } T^2}$, $\frac{\text{Bug2}}{\text{Bug-based } T^2}$
1	1.24	1.38
2	1.01	1.01
3	0.93	1.43
4	1.47	1.14
5	1.06	1.06
6	1.05	1.13
7	1.20	3.23
Average	1.14	1.48

References

Antich, J. 2006. Reactive Robotics: A Paradigm Not Limited to Simple Tasks. Master's thesis, University of the Balearic Islands.

Antich, J. and Ortiz, A. 2004. An underwater simulation environment for testing autonomous robot control architectures. In Proceedings of the Conference on Control Applications in Marine Systems, pages 509–514.

Arkin, R. and Balch, T. 1997. AuRA: Principles and practice in review. Journal of Experimental and Theoretical Artificial Intelligence, **9**(2-3): 175–188.

Balch, T. and Arkin, R. 1993. Avoiding the past: a simple but effective strategy for reactive navigation. In Proceedings of the International Conference on Robotics and Automation, pages 678–685.

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

Koren, Y. and Borenstein, J. 1991. Potential field methods and their inherent limitations for mobile robot navigation. In Proceedings of the International Conference on Robotics and Automation, pages 1398–1404.

Lee, J. and Arkin, R. 2001. Learning momentum: integration and experimentation. In Proceedings of the International Conference on Robotics and Automation, pages 1975–1980.

Lumelsky, V. and Stepanov, A. 1987. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica, **2**: 403–430.

Mackenzie, D., Arkin, R., and Cameron, J. 1997. Multiagent mission specification and execution. Autonomous Robots, **4**(1): 29–52.

Ranganathan, A. and Koenig, S. 2003. A reactive robot architecture with planning on demand. In Proceedings of the International Conference on Intelligent Robots and Systems, pages 1462–1468.

Scalzo, A., Sgorbissa, A., and Zaccaria, R. 2003. μNAV: a minimalist approach to navigation. In Proceedings of the International Conference on Robotics and Automation, pages 2018–2023.

Sgorbissa, A. 2000. Toward a Multi-Ethnic Community of Humans, Mobile Robots, and Intelligent Devices. Ph.D. thesis, Università di Genova.