8

A Reactive Anticipation for Autonomous Robot Navigation

Emna Ayari, Sameh El Hadouaj and Khaled Ghedira
High Institute of Management, Tunis
Tunisia

1. Introduction

Nowadays, mobile robots are expected to carry out various tasks in all kinds of application fields ranging from manufacturing plants, transportation, nursing service, resource or underwater exploration. In all these applications, robots should navigate autonomously in uncertain and dynamic environments in order to achieve their goals. So, the most current challenge in the development of autonomous robot control systems is making them respond intelligently to changing environments. Navigation in such environments involves many mechanisms such as: object detection, perception, internal building model, decision making, prediction of the future state of the environment and on-line navigation. To attend its goal safely, the robot should minimise interaction with other actors in order to avoid conflict situations. Generally, this problem comes up when many robots and/or actors would have access to the same space at the same time. In this case, the control of autonomous robotic navigation for conflict resolution has been widely studied. Some researchers have been focused on navigation in dynamic environments, where either reactive systems (producing real time behaviour), deliberative systems (introduce reasoning and need much more time to calculate a suitable decision) or hybrid systems (combine deliberative and reactive approaches) have been used in order to attend a known goal.
Typically, reactive systems are used to deal with simple problems (detect an obstacle, go away from an obstacle, follow a wall, etc.). Nevertheless, reactive systems are typically less affected by errors and do not require an explicit model of the environment in order to navigate inside an unknown space. Furthermore, they usually deal only with local information that may be captured at real time. However, in reactive systems, the robot can be derived to a conflict situation with other actors because they ignore prediction and reasoning in the decision process. To face this problem, global planning approaches are used. They consist of elaborating a global plan from beginning state to goal state. These approaches need prior complete information about the state of the environment, so they do not take into account the environment uncertainty. So, in more complex situations, hybrid approaches are used. They combine reactive approaches according to a higher level in order to include anticipation of the state of the environment in the decision process. In these cases, low level control operates in a reactive way (local navigation) whereas high level systems tend to be deliberative. It provides, at each step, a partial moving plan to the robot. But these systems are complex because they need much more time to calculate or to update a suitable trajectory toward a predefined goal. In this situation, it is interesting to introduce prediction

in reactive schemas, without using planning techniques, in order to consider the environment's evolution in the future.
To deal with uncertainty, autonomous navigation involves using systems for navigation control that must be not too computationally expensive, as this would result in a sluggish response. In this case, we choose the fuzzy logic technique because it is faster when the frequency of the environment's changes is high.
In the field of multi-agent systems, several works have encouraged researchers to develop models simulating robot's behaviours in order to achieve a known target (Posadas et al., 2008). They are more flexible and fault tolerant as several simple agents are easier to handle and cheaper to build. Indeed, the term "agent" has been defined as hardware or a software system with certain properties such as autonomy, social ability, reactivity and pro-activity (Ferber, 1995). In fact, there are similarities between robot and agent because they share the same characteristics. So, the mapping from a robot to an agent seems straightforward because each robot represents a physical entity, independent from other robots, with a specific task.
In this paper, we present a reactive anticipation model based on fuzzy logic that takes into account: (1) the reactive navigation in an environment composed of local minimum and moving obstacles, and (2) the anticipation of blocking situations and the prediction of the nature, the velocity and the position of obstacles in the future. This information is used to predict conflict situations without using a motion planning method. In order to validate our work, we evaluate our approach by simulating various scenarios. We also give a comparative study between results obtained by our model and those of some other approaches.
In the remainder of the article, we give, in section 2, an overview of the existing approaches for conflict resolution applied to autonomous robot navigation. In section 3, we present the most used techniques to deal with the uncertainty of perception and we justify our choice of the fuzzy logic method. In section 4, we describe our model that combines reactivity with anticipation in order to deal with the problem of conflict resolution without using a motion planning. In Section 5, we present our experimental results. Finally, we conclude in section 6 and we give perspectives for this work.

2. Existing approaches for conflict resolution

When navigation occurs in an environment that is totally or partially unknown or even dynamically changing, higher degree of autonomy for a mobile robot is required. Thus, a mobile robot should be able to take decisions on-line and to minimise conflict situations in such uncertain and dynamic environments using only sensors' limited information.
In this context, a wide range of approaches have been adopted to suggest solutions to the problem of space conflict when a robot shares the same environment with other actors.

2.1 Reactive approach

Many reactive approaches have been proposed to resolve the problem of autonomous robot navigation. In the following, we mention the most important among them.

- Potential Fields (Huang et al., 2006; Khatib, 1986): this approach relies on creating an artificial repulsion field around obstacles and an attraction field around the goal.
- Vector Field Histogram (Ulrich & Borenstein, 2000): it uses a heading dependent histogram to represent the obstacle's density. So, the robot can move in the direction where there are less obstacles in order to minimise its interaction with them.

- Dynamic Window (Fox et al., 1997) and Curvature Velocity (Simmons, 1996): they search a space of the robot's translational and rotational velocities. Obstacles near the robot are transformed into this space by eliminating all commanded velocities that would cause a collision within a certain time period. Both these methods take into account the kinematics and the dynamics of the robot. Commanded velocities are chosen based on an objective function that considers both progress towards the goal and robot safety.
- Nearness Diagram (Minguez & Montano, 2000): it consists in analysing a situation from two polar diagrams. One diagram is used to extract information of environmental characteristics and identify the immediate goal valley. The second diagram is used to define the safety level between the robot and the obstacles by identifying the closest one.
- Elastic Bands (Quinlan & Khatib, 1993): this method 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.
- Behaviour-based methods (Jing et al., 2003; Langer et al., 1994): they define fundamental behaviour sets and establish mappings between sensors' information under different situations and reactive behaviours of the mobile robot. The reactive behaviour of the robot can be regarded as a weighted sum of these fundamental behaviours or can be chosen from the behaviour set according to an evaluation function.
- The Velocity Obstacles method (Fiorini & Shiller, 1998): the moving obstacle is transformed into a velocity obstacle. This approach uses the distance between robot and obstacle to minimise conflict at real time. It is considered as a mapping between sensory data and control commands without introducing reasoning in the decision process.

The main drawback of these strategies is that the robot gets into an infinite loop or local minimum when it is moving among multiple obstacles or in conflict situations when it shares the same resource with moving obstacles. To overcome this problem, many methods are proposed. We mention:

- Memory state method that uses a state memory strategy (Zhu & Yang, 2004). The variables from which this method makes ultimate decisions are: "the distance between the robot and the target" and "the distance between the robot and the nearest obstacle".
- Minimum risk approach is based on trial-return behaviour phenomenon (Omid et al., 2009; Wang & Liu, 2005).
- Virtual wall method directs the robot away from the dead-end by placing a virtual wall that bars the limit cycle path (Ordonez et al., 2008). However, these strategies do not take into account the dynamic nature of the environment and the uncertainty of perception. So the robot can be driven into dangerous or conflict situations.

2.2 Global planning approach

Hence, other approaches applied in dynamic environments are proposed with the idea of combining the reactive techniques with global planning methods (Stachniss & Burgard, 2002). At the beginning, a complete plan from present state to goal state is computed. These approaches need prior complete information about the state of the environment, so they do not take into account the uncertainty.

To face this limit, deliberative approaches are appeared. They use a global world model provided by user input or sensory information to generate appropriate actions for the mobile robot to reach the target (Pruski & Rohmer, 1997). This kind of approach enables prediction and reasoning in the decision process by considering the current information and the past information (Nilsson, 1980; Sahota, 1994). The deliberative control architecture comprises three modules: sensing, planning and action modules. First, robot sense it's surrounding and creates a world model of static environment by combining sensory information. Then, it employs planning module to search an optimal path toward the goal and generate a plan for robot to follow. Finally, robot executes the desired actions to reach the target. After a successful action, robot stops and updates information to perform the next motion. Then, it repeats the process until it reaches the goal. It can coordinate multiple goals and constraints within a complex environment (Huq et al., 2008; Yang et al., 2006).
However, in deliberative navigation, accurate model of environment is needed to plan a globally feasible path. The computational complexity of such systems is generally too great to attain the cycle rates needed for the resulting action to keep pace with the changing environments (it is difficult to obtain a completely known map). To perform necessary calculations, enormous processing capabilities and memory is needed.
Therefore, these approaches are not proper in the presence of uncertainty in dynamic or real world.

2.3 Hybrid approach

In more complex situations, other control architectures appeared (Oreback & Christensen, 2003), including the anticipation in the decision process. This combination is known as hybrid architecture (Arkin, 1990) which has two levels: reactive and deliberative. The deliberative level has to determine and offer the reactive level those behavioural patterns that are necessary for the robot to achieve its objectives. The reactive level has to execute these behavioural patterns by guaranteeing real-time restrictions (Fulgenzi et al., 2008; Fulgenzi, 2009).
Practically, these methods are more complex and require more time calculations. They usually use local methods of obstacle avoidance in order to expand the tree and cover the search space. However, local methods are less suited to dynamic environments (Minguez et al., 2002).
In (Schwartz & Sharir, 1983), a general algorithm was developed. It is doubly exponential time, this limit has been lowered by the Canny algorithm described in (Canny et al., 1990) whose running time is exponential in dimension.
The algorithm D* proposed by Stentz (stentz, 1995) is a generalisation of the A* search algorithm in the case of partially known environments. In a finite discretized configuration space, the initial cost of shipping is to move from one configuration to another. A path is first found, and then the robot begins to execute it. If a change in the cost of a path is detected, only the relevant configurations are considered and the optimal path is updated in less time.
Partial motion planning (PMP) (Petti & Fraichard, 2005), the algorithm explicitly takes into account the computation time, constraints and problems safety of navigation in dynamic environment. A tree is grown using a conventional algorithm based on sampling. A node is added to the tree if there is not an inevitable consequence collision. This enables PMP to

provide a safe partial path at any time. During the execution of this partial path chosen, another partial path is developed from the end of the previous path developed. This method is applied at real time in dynamic environments, but its major drawback is that it does not consider the uncertainty of the information collected. So it can produce non-executable partial paths.

2.4 Multi-agent systems and robotic simulation

In the field of multi-agent systems, several works have encouraged researchers to develop models simulating robot's behaviours in order to achieve a known target. In fact, there are similarities between robot and agent. So, the mapping from a robot to an agent seems straightforward because each robot represents a physical entity, independent from other robots, with a specific task. Hence, multi-agent simulations models are widely used to solve problems in mobile robotics under dynamic environments. In this way, many approaches are using multi-agent systems to simulate and control autonomous mobile robot in order to resolve the problem of space's conflict by minimising the interaction with agents. For example, (Ros, 2005) proposes a multi-agent system for autonomous robot navigation in unknown environments. In this work, the authors use the case-based reasoning (CBR) techniques in order to solve the problem of local minimum and avoid only static obstacles. In (Ono, 2004), a multi-agent architecture is also proposed to control an intelligent wheelchair. It takes into account only three behaviours: obstacle avoidance (using distance), door passage and wall following. However, it cannot combine behaviours together and it is applied only in static environment. In (Innocenti, 2007), a multi-agent system is proposed as the robot control architecture divided into four sub-systems of agents: perception, behaviour, deliberative and actuator. This architecture is applied in static and dynamic environments and uses the distance between the robot and the moving obstacle in order to predict conflict situations. But this information gives a limited knowledge about the state of the environment and does not allow the robot to take an intelligent decision while navigating in a dynamic and unknown environment. In (Posadas et al., 2008) a multi-agent system is proposed to control the navigation of robot by using a hybrid approach (it uses a motion planning). The architecture is composed of two levels; reactive and deliberative. The communications between these levels are assured by the agents.

2.5 Discussion

Reactive systems are applied in local navigation and used to deal with simple problems. Nevertheless, reactive systems are typically less affected by errors and do not require an explicit models of the environment in order to navigate inside an unknown space. Furthermore, they usually deal only with local information that may be captured at real time. However, reactive systems may fall into local traps or blocking situations because they ignore prediction and reasoning in the decision process. For this reason, the robot can be derived to a conflict situation with other actors sharing the same space. In more complex situations, they are combined according to a higher level in order to include anticipation in the decision process. In these cases, low level control operates in a reactive way (local navigation) whereas high level systems tend to be deliberative and use motion planning method to update the trajectory when there are modifications in the state of the environment. These approaches are called hybrid approaches and are well applied in uncertain and dynamic environments by producing, at each step, a motion planning method

that minimise the conflict (Petti & Fraichard, 2005). Nevertheless, building a movement planning requires an important computation time in order to find the most appropriate way. However, if the frequency of environment's changes is high, the use of a planning method becomes not only inefficient, but also useless (if the robot generates plans without using them).
So, in this situation it is interesting to introduce prediction in reactive schemas in order to consider the environment's evolution in the future. Hence, the main concern of this paper is to propose a model that combines reactivity and anticipation in order to resolve the problem of conflict without using a motion planning. Thus, the robot should anticipate the nature and velocity of the obstacles in order to minimise interactions. We use the multi-agent system to simulate the robot's behaviour because there is a mapping between robots, while navigating in dynamic environments, and multi-agent systems.

3. Autonomous navigation in uncertain environment

Autonomous navigation in an uncertain environment involves using systems for navigation control that must not be too computationally expensive, as this would result in a sluggish response. In this case, soft computing methods have played important roles in the design of the robot controllers. The commonly used soft computing methods are: Bayesian Networks and Fuzzy Logic.

3.1 Bayesian networks

Bayesian Networks are models which capture uncertainties in terms of probabilities that can be used to perform reasoning under uncertainty. It epitomises probabilistic dependency models that represent random stochastic uncertainty via its nodes (Darwiche, 2009). The Bayesian inference has been widely used especially in the localisation problem (Thrun, 1998), in the mapping and in the learning mechanisms in mobile robotics. But this technique is very slow and complex (NP-Complex) because it uses a probabilistic reasoning that require much more time to choose a suitable decision. Also, this technique requires a causal model of reality that is not always given. So, its application in autonomous navigation in the case of dynamic environments can be unprofitable.

3.2 Fuzzy logic technique for autonomous navigation

Fuzzy Logic (FL) has been investigated by several researchers to treat the problem of uncertainty in perception. This technique represents uncertainty via fuzzy sets and membership functions. It gives robustness to the system with respect to inaccuracy in data acquisition and uncertainty, and makes definition behaviour and their interactions quite easy by using simple linguistic rules (Klir et al., 1997; Sajotti et al., 1995). It also allows implementing of human knowledge and experience without requiring a precise analytical model of the environment. Probably, the greatest strength of behaviour-based fuzzy approaches is that they operate on/and reason with uncertain perception-based information, which makes them suitable even for difficult environments.
We present in table 1 comparison results between the characteristics of the Bayesian Networks and the Fuzzy Logic. According to this comparison study, we can conclude that the fuzzy logic technique is faster, easier to implement and well applied to autonomous robot navigation.

	Fuzzy Logic	Bayesian Network
Complexity	Simple rules	NP-complex
Reasoning	Fuzzy reasoning	Probabilistic reasoning
Application	Avoidance collision	Localization and mapping
Controller	Fuzzy inference	Bayesian inference
Rapidity	Fast (simple rules)	Slow

Table 1. Comparison between fuzzy logic and Bayesian network.

However, the majority of existing fuzzy logic methods deal only with static environments, and only use the distance between the robot and the obstacle to avoid collision and to minimise conflict with other agents sharing the same space (Bonarini et al, 2003; Selekwa et al., 2008). This kind of information gives a limited knowledge about the state of the environment, so the robot cannot take intelligent decisions. For this reason, in our work, we adopt a fuzzy logic technique to deal with uncertainty. We propose a fuzzy model for autonomous navigation in a dynamic and uncertain environment based on the nature and the velocity of obstacles.

4. Proposed approach

According to Ferber (Ferber, 1995), a multi-agent system is composed of:

1. An environment 'E',
2. A set of objects 'O' in 'E',
3. A set of agents 'A' in 'E',
4. A set of relationships 'R' between agents,
5. A set of operation 'op' enabling agents 'A' to collect, produce, consume and manipulate objects of 'O'.

Basing on this definition, we define our multi-agent system as a set of objects which represent the static objects (have a fixed position) and a set of agents (have a moving position) that represent dynamic objects and the robots. In our model, the principal agent is the "robot". It has its own physical parts including sensors, processors, actuators and communication devices. So, in order to guarantee the safety of the mobile robot, it should be able to navigate by minimising interactions with the other agents navigating in the same space in order to minimise conflict and to achieve a known goal.

This section is organised as follows. We present in subsection 4.1 the perception model of the robot. It allows the robot to perceive its environment in order to take the suitable decision autonomously. In subsection 4.2, we present the kinematic model of the robot. It allows the robot to localise in its environment. In subsection 4.3, we describe the principle of the fuzzy controller technique. In subsection 4.4, we present our fuzzy controllers model. The first controller is described in subsection 4.4.1. It allows the robot to determine its angular velocity. The second controller is presented in subsection 4.4.2. It allows the robot to calculate its linear velocity. We present in subsection 4.4.3 the system rules applied in our fuzzy controllers used to avoid conflict situations.

4.1 The perception model

One of the ultimate goals of robotics is to realise autonomous robots that are able to organise their own internal structure and to achieve their goals through interactions with dynamically changing environments. The robot should take decisions on-line using only sensors providing limited information (without prior information about the position of obstacles and their trajectories) when it navigates autonomously in dynamic spaces.

Our main objective is to make the robot able to predict the nature and the velocity of obstacles (static or moving) in order to minimise conflict situations, to avoid collision and to achieve a specific goal. At this stage, we use a perception model composed of eight ultrasonic sensors. We choose this type of sensors because it has some attractive properties, e.g. cheapness, reliability and soon, which makes it widely used in mobile robots. This type of sensor calculates the distance between the robots.

According to this model of perception, we divide the robot's space into three subspaces controlled by three groups of sensors (see figure 1). The subspaces are: (1) the front area "F" controlled by the first group composed of sensors number 1,2,7,8, (2) the left area "L" controlled by the second group composed of sensors number 3, 4, and (3) the right area "R" controlled by the third group composed of sensors number 5, 6.

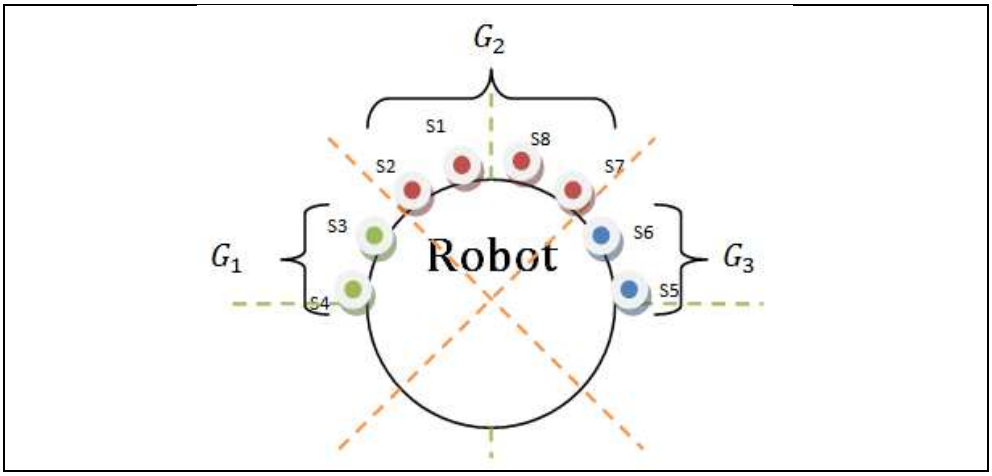


Fig. 1. The Robot's perception areas.

4.2 The kinematic model

In this study, the mobile robot is a system, which is subject to non holonomic constraints. Its position in the environment is represented by P=(x,y,θ), where (x, y) is the position of the robot in the reference coordinate system XOY, and the heading direction θ is taken counter clockwise from the positive direction of X-axis (see figure 2). X'O'Y' is the reference coordinate according to the robot system. It allows the robot to locate the objects in the environment.

The robot is composed of two wheels (left and right). The velocity, acceleration and orientation angle of the robot is assured by the fuzzy controllers.

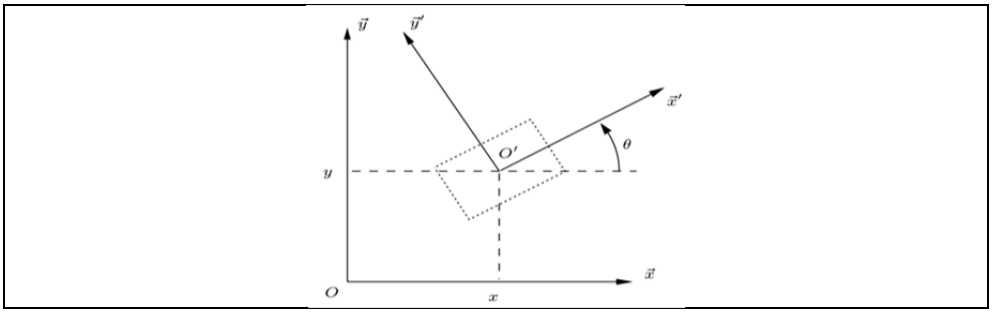


Fig. 2. Robot's kinematic model.

4.3 The fuzzy controller

A Fuzzy Controller (FC) is a control system, whose input is the output of the process to be controlled (sensory data, internal state). Its outputs are commands for the actuators of the process. The ranges of the input and output parameters/variables are represented by membership functions and fuzzy sets. In addition, the interactions between input and output variables/parameters are represented by fuzzy rules. In a nutshell, system input parameters and variables are encoded into fuzzy representations using well defined "If/Then" rules which are converted into their mathematical equivalents. These rules would then determine actions to be taken based on Implication Operators such as Zadeh Min/Max (Zadeh, 1973), or Mamdani Min (Mamdani, 1974). The fuzzified data is then put through a defuzzification process via Center of Gravity, Center of Sum or Mean of Maxima methods to obtain actual commands for the actuators of the process.

In our work, we use the Mamdani fuzzy inference methodology in order to provide the adequate way of modelling the relevance of the controller. For the defuzzification process we choose the Center of Gravity method by applying the formula 1. The crisp value U* is the geometrical center of the output fuzzy value μ(y), where μ(y) is the union of all the contributions of rules whose Degree Of Freedom (DOF) is more than zero, y is the universe of discourse and b is the number of samples.

$$U^* = \frac{\sum_{i=a}^{b} \mu(y) \times y}{\sum_{i=a}^{b} \mu(y)} \tag{1}$$

4.4 The principle of the behaviour controller based on Fuzzy Logic

The use of fuzzy logic in the design of autonomous navigation behaviours is nowadays quite popular in robotic. The set of behaviours that are being implemented can include, for example, the following of walls, corridors or the avoidance of obstacles. However, usually the existing fuzzy logic methods use only the distance between the robot and the obstacle in order to avoid conflicts and so deal with static environments.

We propose a fuzzy model for the navigation in dynamic and uncertain environments based on"the nature" and "the velocity" of obstacles. For example, if there is an obstacle in front of the robot, it is more logical to reason about its velocity. In fact, the distance between the robot and the obstacle allows it to immediately change the trajectory without taking into account the obstacle nature (mobile or static) and whether the obstacle is going in the same

direction of the robot or not. In contrast, the velocity allows the robot to predict if there is conflict in the future. In the reminder of this section, we present robot behaviour modules implemented as fuzzy logic controllers. Each behaviour module receives data input that describes the situation and produces an output to be addressed to the actuators. Our fuzzy architecture is composed of two fuzzy controllers. The first controller is in charge of determining the angular velocity of the two wheels. This controller allows the robot to change its angular orientation based on the prediction of the nature of the obstacle, in order to avoid conflict and to move closer to its target. The second controller uses the velocity of the obstacle. It is in charge of determining whether the robot's speed should be increased (accelerate its speed if there is a free space) or decreased (minimise its speed if there is an obstacle in its trajectory).

4.4.1 Obstacle nature prediction: Time collision concept

Typically ultrasonic sensors calculate the distance between the robot and the obstacle and therefore they do not provide information about the obstacle's nature (mobile or static) and its position in the future state. However, this kind of information is necessary to predict conflict situations in the future. In order to overcome this problem, in our work, the robot uses this distance to calculate new information called "Time Collision" between the robot and the obstacle.

The Time Collision (TC) represents the time left for the robot before a collision occurs with an obstacle. In order to predict the nature and the position of the obstacle in the future, the robot operates as follows. At time Ti, the robot should observe its environment (using the perception model). If it detects an obstacle, it calculates a Time Collision Needed (TCN) representing the time required to collide with this obstacle in the future while keeping the same velocity (see figure 3). At time Ti+1, it repeats the same procedure to recalculate a Time Collision Remaining (TCR) representing the time required to collide the same obstacle in the future while keeping the same velocity (see figure 3). The TC can be obtained by applying the formula 2. Di represents the distance between the robot and the obstacle at time Ti, and VRi represents the velocity of the robot.

$$TC=D_i/VR_i \quad (2)$$

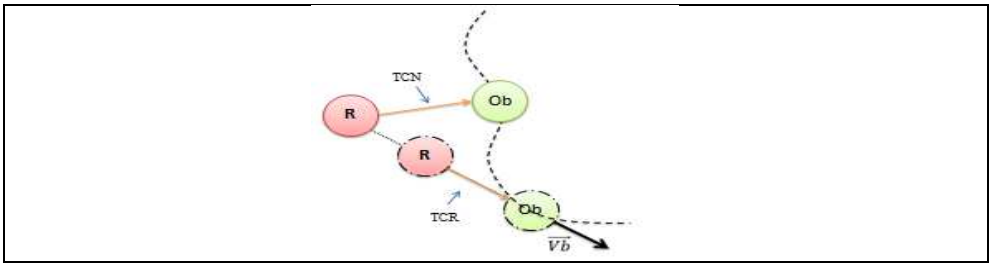


Fig. 3. Illustration of "the Time Collision".

From these two values, the robot can calculate a Difference Time Collision (DTC) with this obstacle by applying formula 3. If the DTC is equal to zero, it means that the obstacle in the trajectory of the robot is static. In this situation, the robot should immediately change its direction in order to avoid local minimum. If the DTC is positive, then the obstacle is

considered as a mobile one. In this situation, the robot can expect that the trajectory will be free after a period of time and then it does not need to change its direction. If the DTC is negative, then obstacle is mobile and it is going towards the robot. In this case, the robot should minimise its velocity in order to avoid conflict with this obstacle, and then decide to change the direction completely if it is possible (moves to another subspace, which is safe), or wait until this mobile object moves away (the current subspace will be free), if there is no other free subspace.

$$DTC=TCN-TCR \tag{3}$$

The advantage of this information is that it allows the robot not only to avoid collision and local minimum at real time with obstacles around it, but also to anticipate their natures and their positions in the future so as to predict conflict situations.
We present in figure 4 the FLC of angular velocity. We describe four linguistic variables that are: the DTCL, DTCF, DTCR, and TO. These variables represent respectively: the difference time collision on the left, the difference time collision in the front, the difference time collision on the right and the current target orientation between the robot and the goal. Each

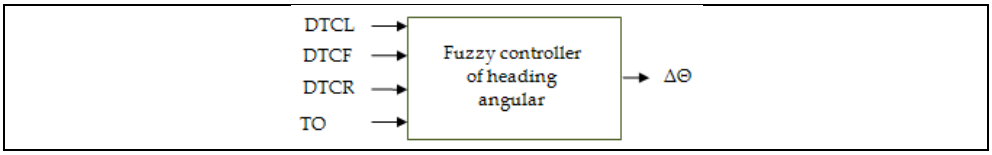


Fig. 4. Fuzzy controller of heading angular

DTC have two membership functions: Fixed Obstacle and Moving Obstacle. The DTCL is used to control the left area. We present in algorithm 1 the reasoning process of the robot that is used to predict the nature of the nearest obstacle in this area. Thus, if the DTCL is

```
1: if (DTCi) ≤ the time interval Δ then
2:       The obstacle is static
3: else
4:      The obstacle is mobile
5:       Free space
6: end if
```

Algorithm 1. The robot's behaviour is based on the prediction of the nature of the obstacles in left and right areas

smaller or equal to Δ (the time interval between two successive perceptions) it means that the obstacle is static; else it is considered as mobile. We can obtain the value of this variable from the formula 4. The same reasoning process is applied to the right area based on the DTCR that can be obtained from the formula 5. The value of DTCF variable can be obtained from the formula 6. Algorithm 2 describes the reasoning mechanism of the robot that is used to predict the nature of the nearest obstacle in the front area. For example, if the DTCF is equal to zero, this means that the obstacle is static. Therefore, the robot should change its trajectory immediately in order to avoid collision. Otherwise, if the DTCF is equal to infinity, this means that the space is free. In order to move toward a specific goal, the robot

```
1:  if (DTCF) =0 then
2:       The obstacle is static. The robot should change its trajectory
4:  else
5:      if ((DTCF)>0) and ((DTCF) < + ∞) then
6: The obstacle is mobile and it is moving away from the robot.
7:       else
8: if ((DTCF) <0) and ((DTCF)> −∞) then
9: The obstacle is mobile and it is going toward the robot.
10: else
11: if (DTCF) =- ∞ then
12: The space is considered as free at t_i
13: else
14: if (INF (DTCF) =+ ∞ then
15: No obstacle in front of the robot. Free space
17: end if
18: end if
19:              end if
20:       end if
21: end if
```

Algorithm 2. The robot's behaviour is based on the prediction of the nature of the obstacles in front area

should have an idea about the area in which the target exists. For this purpose, the TO which gives an idea about this position is combined with the DTC for each area in order to reach a compromise between "avoid conflict" and "reach goal". The TO has three membership functions that are: L: Left, F: Front, R: Right. The output of this controller is the angular velocity, it has five membership functions: LLT: Large Left Turn, SLT: Small Left Turn, NT: No Turn, SRT: Small Right Turn, LRT: Large Right Turn. Note that figure 5 describes the input of the fuzzy sets involved in the definition of the heading angular. Figure 6 describes the output variable for the angular velocity.

$$DTCL = Sup\ (DTCL_i) \text{ where } i=3, 4 \tag{4}$$

$$DTCR = Sup\ (DTCR_i) \text{ where } i=5, 6 \tag{5}$$

$$DTCF = Inf\ (DTCF_i) \text{ where } i=1, 2, 7, 8 \tag{6}$$

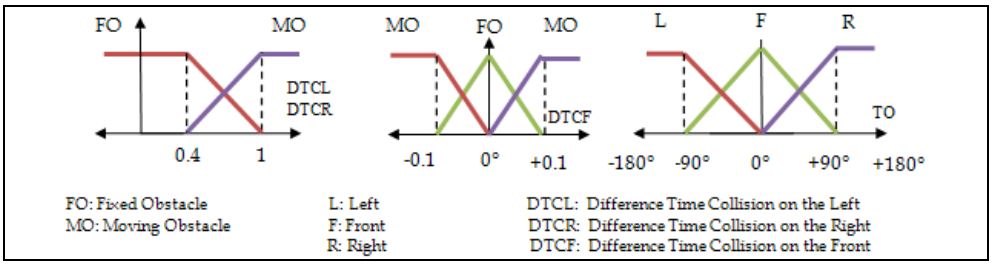


Fig. 5. The definition of angular velocity input membership functions.

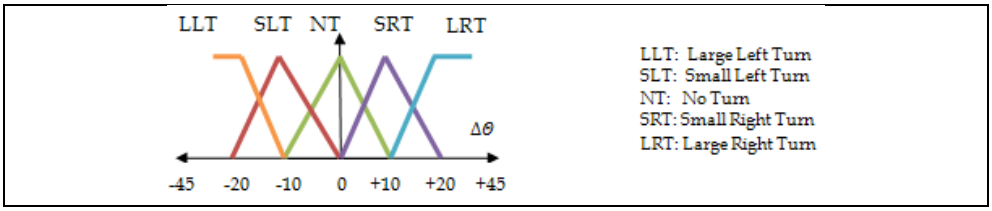


Fig. 6. The definition of angular velocity output membership functions.

4.4.2 Linear velocity prediction: Obstacle velocity concept

The fuzzy controller for angular velocity uses the DTC that gives information about the nature of an obstacle and its trajectory. So, the robot is able to predict the future situation of the environment if it keeps moving in the same direction. While the second fuzzy controller for linear velocity uses the obstacle's velocity in the front area. This information determines whether the robot's speed should be increased or decreased to respond to the nearest situation detected.

Hence, the obstacle's velocity can be calculated according to formula 7 where D represents the distance travelled by the obstacle during the time interval Δ.

$$\vartheta = D/\Delta \tag{7}$$

The distance D is calculated according to formula 8 where X_c and Y_c represent the coordinates of the obstacle at time t_i and X'_c and Y'_c represent the coordinates of the obstacle at $t_i+\Delta$ (see figure 7). The coordinates are obtained according to formula 9.

$$D=\sqrt{(X_c-Y_c)^2+(X'_c-Y'_c)^2} \tag{8}$$

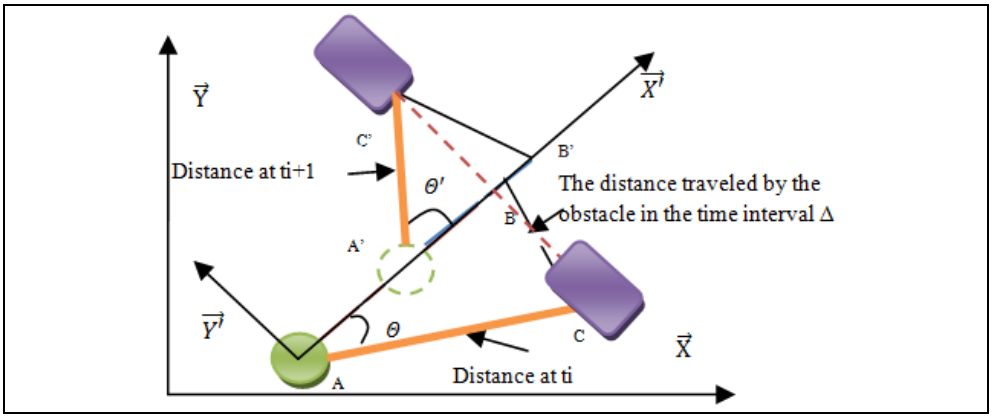


Fig. 7. Illustration of distance traversed by the obstacle.

$$\text{Cord}= \begin{cases} X_c=X_a+(X_b-X_a)\cos\theta-(Y_b-Y_a)\sin\theta \\ Y_c=Y_a+(Y_b-Y_a)\cos\theta+(X_b-X_a)\sin\theta \end{cases} \tag{9}$$

We present in figure 8 the fuzzy logic controller for the linear velocity. This controller determines whether the robot's speed should be increased or decreased. It is used to determine the speed change Δϑ (the linear velocity).

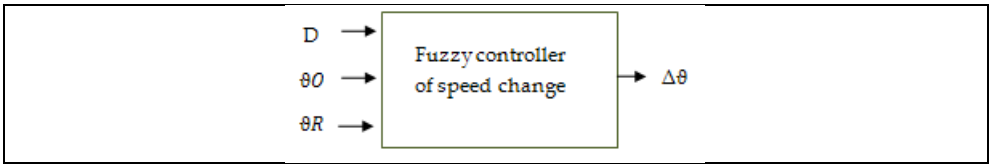


Fig. 8. Fuzzy controller of speed change

The D, ϑO, and the ϑR represent respectively the distance between the robot and the nearest obstacle in the front area, the velocity of the nearest obstacle in the front area and the current velocity of the robot. The input and output membership functions of the fuzzy controller for linear velocity are shown respectively in figure 9 and figure 10.

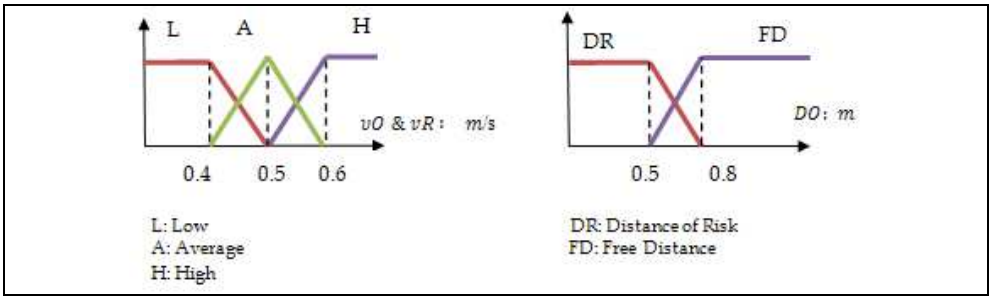


Fig. 9. Input membership functions for the linear velocity

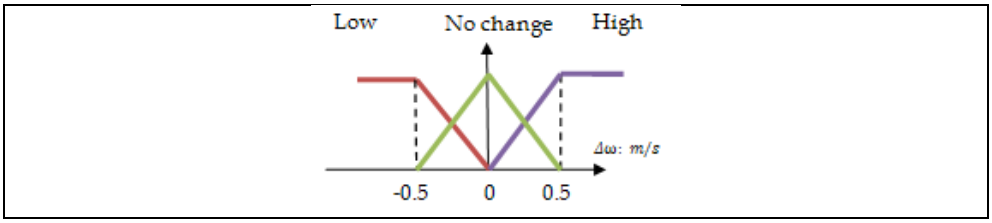


Fig. 10. Output membership functions. Velocity up/down control.

So, the speed of the robot can be influenced by the velocity of the obstacle in the front area. We present in algorithm 3 the reasoning process of the robot used to predict the velocity of the nearest obstacle in this area in order to determine its velocity. For example, if the velocity of the obstacle is smaller than the velocity of the robot, then the robot decreases its speed in order to avoid collision in the near future. If the velocity of the obstacle is larger than the velocity of the robot, the robot keeps the same speed if its velocity and the velocity of the obstacle are close. However, if there is a difference between the speed of the robot and the obstacle, it accelerates.

```
1:  if (VO) =0 then
2:      The obstacle is static
3:      The robot must minimise its acceleration.
4:  else
5:     if (VO < VA) then
6:           Reduce the velocity of the robot in order to avoid conflict in the near future.
7:      else
8: if (VO > VA) then
9:             Keep the same speed if the velocity of the robot and the velocity of the
obstacle     10:            are close or accelerate if they are different.
11: else
12:                  Keep the same speed.
13:            end if
14:       end if
15: end if
```

Algorithm 3. The velocity of the robot according to the concept of the velocity of the obstacle

4.4.3 Fuzzy system rules

Robot navigation is actually governed by rules in the fuzzy controller which represent qualitatively the human driving heuristics. The fuzzy logic controllers are applied to coordinate and to combine the various behaviours of the robot in order to choose a suitable decision. The fuzzification converts the input variables into input grades by means of the membership functions shown in figure 5 and figure 9. The inference and aggregation generate a resultant output membership function with respect to fuzzy rules. The defuzzification gives the output membership function shown in figure 6 and in figure 10. Therefore, the total number of rules to determine the angular velocity is N= 24. We can reduce this number into 9 if-else rules by eliminating redundancies (rules that have different values of input variables and the same value of the output variable can be replaced by one rule). We present in table 2 the fuzzy rules for the angular velocity. For example, if DTCF=M (the obstacle in the front area is moving) and TO=F (the target is in front area) then $\Delta\theta$=NT (the robot do not change its trajectory) whatever the value of DTCL and DTCR. Likewise, for the linear velocity there are N=18 rules, but we can reduce them into 9 if-else rules.

The use of a small number of rules is required in dynamic environments, this is especially important when a fast response to the changes is required (short execution and high frequency of sensor readings). We use the method of Center of Gravity Defuzzification to

DTCF		F				M			
DTCL		F		M		F		M	
DTCR		F	M	F	M	F	M	F	M
TO	F	LRT	LRT	SLT	SLT	NT	NT	NT	NT
	L	LRT	LRT	LLT	LLT	NT	NT	LLT	LLT
	R	LRT	LRT	LRT	LRT	NT	LRT	NT	LRT

Table 2. The fuzzy rules for the angular velocity.

calculate the output for the two controllers. The constants of output membership function for the first controller (angular velocity) are shown in figure 6 and the values of the constants of output membership function for the second controller (linear velocity) are shown in figure 10. During simulation work, the robot's normal velocity VN is set to 0.5m/s.

5. Experimentation results

The robot calculates a "TC" with each obstacle around it. The value of "TC" allows it to predict the nature of each obstacle (static or dynamic). Then, the robot can calculate the velocity of each obstacle in order firstly to anticipate the state of the environment in the future and therefore to adjust its speed in order to avoid collisions.
In order to prove the efficiency of the proposed model, we proceed as follows. In a first time, we test our method in static environments (composed of U-shape and T-shape obstacles). Having obtained valid results, we achieve, in a second time, tests in dynamic environments (including moving robots). We have used the Simbad simulation platform.

5.1 The parameters of the robot and the controller in simulation

In simulation, the parameters of the robot are: the maximum linear velocity ϑmax = 1m/s and the maximum angular velocity θmax = 45/s. The height of the mobile robot is 0.78 m, the diameter is 0.4 m, the weight is 50 kg and the radius of the wheel is 0.05m. The Δ time interval between two successive perceptions is 0.4s. This interval time is used to calculate the TC.
In addition, we have defined a variety of basic scenarios for the experimentation stage. In the reminder of this section we detail some of them. For each scenario, the robot is initially positioned at the centre of the environment and starts to work without having any preliminary information. The simulation environment is a square area composed of multiple traps like U-shape and T-shape forming the static obstacle, and moving agents (moving objects and other robots).

5.2 Experimentation results in static environments

Test results in static environments are presented in figure 11. Each scenario shows the robot's environment with the robot's path covered and the detected obstacles.
Scenario 1 The robot is initially positioned in an environment composed of multiple "dead cycle" as shown in figure 11.a. The robot calculates for each trap a DTC in order to predict the nature of the obstacle. The robot's decision depends on this information (e.g. if the DTC is equal to zero then, the obstacle is static and the robot should change its direction). In region 'a' it detects that the DTCF is equal to zero, which means that the robot should change immediately its direction to find a free path towards its target. In region 'b', it changes its behaviour and Turns Right. In region 'c', the robot detects a new local minimum (DTCF=0, DTCR<ΔT and DTCL<ΔT). In this situation, the robot should find a free trajectory in order to move away from this obstacle. When it leaves this local minimum, the robot adjusts its trajectory to minimise the distance between the robot and the target. For each step, the robot reasons about the nature, velocity and the orientation angle in order to choose a direction that will be free in the future (predict the future state using the nature and the velocity of obstacle in order to find a suitable path with fewer constraints). However, in (Wang & Liu, 2005) based on trial-return behaviour phenomenon, the robot searches the nearest exit. Therefore, when the nearest exit is blocked by along wall, the

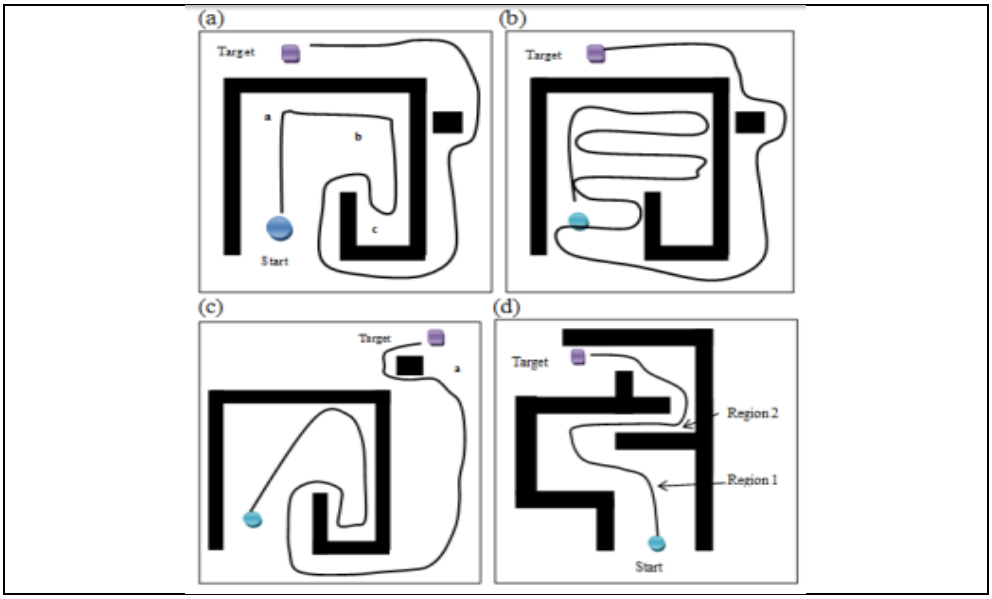


Fig. 11. Experimental results in static environments

nearest exit will be the opening at the right hand side where the wall ends (see figure 11.b). This method takes a large time to find the free end of the wall. However, the problem with trial-return motion is high power consumption and the time spent from start point to target is long when the obstacle is composed of complex traps like U-shape and T-shape. In (Zhu & Yang, 2004), the dead cycle problem is resolved by using a state memory strategy (see figure 11.c). Therefore, there is a situation where the robot cannot go straight toward the target. It has to keep turning around the obstacle (point "a") if the distance between the robot and the obstacle is shorter than the distance between the robot and the target.

Scenario 2 We test our method in an environment composed of corridors (wide and narrow). As shown in figure 11.d the robot moves in a corridor in order to reach its target. The problem is how to ensure the motion's stability without oscillation. In our architecture, there is a compromise between "reach target" and "avoid obstacles". In region 1, the robot crosses a large corridor; it changes its direction when it detects a static obstacle. In region 2, the robot crosses a narrow corridor; in this situation the DTCR<ΔT and the DTCL<ΔT. Thus, the robot cannot change its trajectory, but it should only adjust its velocity with the velocity of the nearest obstacle in the front area. This reasoning allows the robot to reach its target easily without oscillation.

5.3 Experimentation results in dynamic environments

Scenario 1 Generally it is difficult to deal with narrow passage cases when there are moving actors because there may be oscillation in the trajectory between multiple obstacles. However, we can show, in figure 12, the effectiveness of our method when dealing with this case. The robot navigates in an environment composed of corridors with a moving obstacle. At the first time, the corridor is narrow. In this setting the robot adjusts its velocity to avoid collision with the obstacle without changing the direction because the goal is in the front

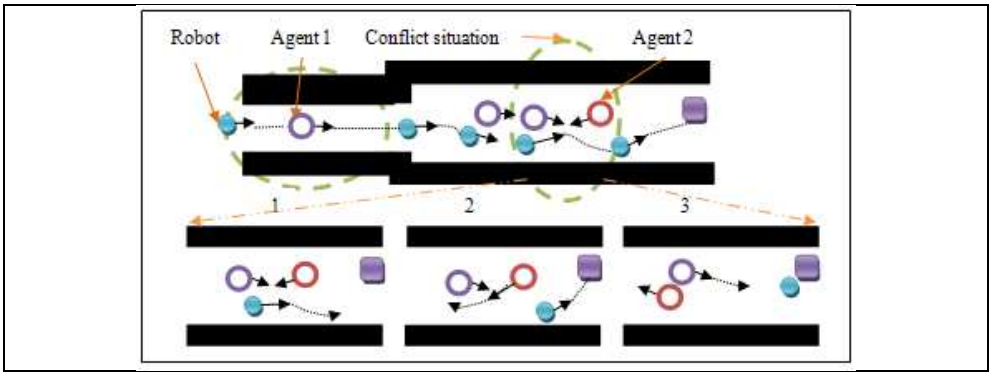


Fig. 12. Experimental results in a dynamic environment: case of conflict situations in narrow and large corridors.

area and there is no free area neither on the left nor on the right (DTC<ΔT on the Left and the Right). When the corridor becomes large, the robot changes its direction because there is a free area on the right. In this trajectory, the robot detects that there is a conflict space (DTCF is negative). With the concept of DTC, the robot can find the trajectory that will be free in the future. Thus, the robot should choose the most adequate direction that minimises states of conflict. So, in our model the robot can go through narrow passages successfully. However, in (Lazkano et al., 2007) a reactive navigation method based on Bayesian Networks is proposed. This method only takes into account the door-crossing behaviour. Its drawback is that it becomes slow when the environment becomes more dynamic because it requires a long computing time.

Scenario 2 We test our method in a dynamic environment composed of multiple traps as shown in figure 13. The robot adjusts its velocity in order to avoid collision with moving

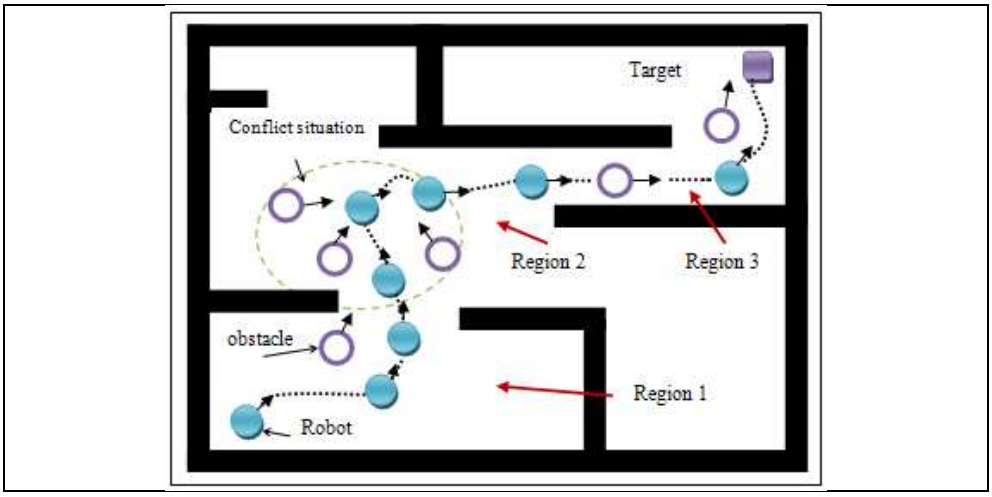


Fig. 13. Experimental results in a dynamic environment: case of conflict situations in a dynamic environment composed of local traps.

obstacles and changes its orientation if it detects a static obstacle. In region 1, it moves away from a U-shape obstacle because it detects that the DTC is equal to zero. In region 2, the robot is in a large corridor with moving obstacles, it calculates the DTC relative to every moving obstacle in order to predict conflict situations. In this setting, the DTC of every obstacle around the robot decreases, which means that the obstacles are going toward the robot. At the first step, the robot calculates the velocity of the nearest moving obstacle in order to adjust its velocity. At the second step, it reasons about the trajectory that it has to choose (minimise interactions with other agents).

According to a large number of simulation experiments, we can conclude that:

1. The robot uses a simple model of perception composed of ultrasonic sensors that provides information at real time.
2. The robot can provide a suitable trajectory in dynamic environments and there are no local minimum encountered.
3. The results provided by our method are satisfactory with the robot's dynamic constraints.
4. The robot can easily predict the nature of obstacles around it, and it uses this kind of information to predict conflict situations with other agents sharing the same resources.
5. The robot can also predict the velocity of each obstacle, using its simple model of perception, to adjust its linear velocity in order to avoid collision with other agents.
6. Our method can be adapted to different cases in dynamic environments.

6. Conclusion and perspectives

In this paper, a new method based on the sensors' information to combine reactivity and anticipation in order to predict conflict situations between robots has been proposed. We have used fuzzy logic to develop a control system that enables the robot to navigate at real time and to minimise its interaction with other agents. Basing on the simulation experiments, we can conclude that our method operates in different cases of dynamic environments. The robot can easily predict the nature of obstacles in order to anticipate conflict situations. It can also predict the velocity of each obstacle and adjust its linear velocity in order to avoid collision. As perspectives for this work, we attend to improve the robot's behaviour by introducing both the ability to learn and the concept of communication between robots in order to explore the environment and to share knowledge.

7. References

Arkin, R. (1990). Integrating behavioural, perceptual and world knowledge in reactive navigation. *Robots and Autonomous Systems*, Vol.6, pp. 105-122

Bonarini, A., Invernizzi, G. & Labella, T. H. (2003). An architecture to coordinate fuzzy behaviours to control an autonomous robot. *Proc. Fuzzy sets Systems*, pp.101-115

Canny, J., Rege, A. & Reif, J. (1990). An exact algorithm for kino dynamic planning in the plane. *In ACM Symp.On Computational Geometry,*pp. 271-280.

Darwiche, A. (2009). Modelling and Reasoning with Bayesian Networks. *New York, NY: Cambridge University Press*

Ferber, J. (1995). Les systèmes multi-agents vers une intelligence collective.*InterEditions*, Paris.

Fiorini, P. & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *International.Journal.Robot. Res*, Vol.17 (7), pp. 760-772

Fox, D., Burgard, W. & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, pp.23-33

Fulgenzi, C. (2009). Autonomous navigation in dynamic uncertain environment using probabilistic models of perception and collision risk prediction, PhD thesis, INRIA Rhône-Alpes, Grenoble France

Fulgenzi, C., Tay, C., Spalanzani, A. & Laugier, C. (2008). Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian Processes. *Proc.IEEE/RSJ International Conference on Intelligent Robotsand Systems*, pp.508-513

Huang, W.H., Fajen, R., Fink, J. & Warren, W.H. (2006). Visual navigation and obstacle avoidance using a steering potential function, *Proc. Robotics and Autonomous Systems*, pp. 288-299

Huq, R., Mann, G. K. I. & Gosine, R. G. (2008). Mobile robot navigation usingmotors schema and fuzzy context dependent behaviour modulation. *Appl. Soft. Comput.*, pp. 422-436

Innocenti, B. (2007). A multi-agent architecture with cooperative fuzzy control for a mobile robot. *Proc. Robotics and Autonomous Systems,* Vol. 55, PP. 881-891

Jing, X., Wang, Y. & Tan, D. (2003). Cooperative motion behaviours using biology-modelling behaviour decision-making rules. *Cont. Theory. Appl*, Vol.20 (3), pp. 407-410

Khatib, O. (1986). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *In International Journal of Robotics Research*, pp.90-98

Klir, G. J., Yuan, B. & Clair, U. S. (1997). Fuzzy Set Theory: Foundations and Applications. *Prentice-Hall, Upper Saddle River*

Langer, D., Rosenblatt, J. & Hebert, M. (1994). A behaviour-based system for off-road navigation. *Proc.IEEE Transaction. Robot. Automation*, Vol.10, pp. 776-783

Lazkano, E., Sierra, B., Astigarraga, A. & Martnez-Otzeta, J.M. (2007). On the use of Bayesian Networks to develop behaviours for mobile robots. *Robotics and Autonomous Systems*, Vol.55, pp. 253-265

Mamdani, E. H. (1974). Applications of fuzzy algorithms for simple dynamic plants.*Proc.IEEE, 121(12),* pp. 1585-1588

Minguez, J. & Montano, L. (2000). Nearness diagram navigation: a new real time collision avoidance approach. *Proc. IntelligentRobots and Systems*, pp.2094-2100

Minguez, J., Montano, L., Siméon, N. & Alami, R. (2002). Global nearness diagram navigation. *Proc. IEEE International Conference on Robotics and Automation.*

Nilsson, N. J. (1980). Principles of Artificial Intelligence. *Morgan Kauf-mann Ed, Los Altos, CA*

Omid, R.E.M., Tang, S.H. & Napsiah, I. (2009). Development of a new minimum avoidance system for a behaviour-based mobile robot. Proc.*Fuzzy Sets and Systems*

Ono, Y. (2004). A mobile robot for corridor navigation: A multi-agent approach. *Proc. ACM Southeast Regional Conference, ACM Press*, pp. 379-384

Ordonez, C., Collins, E. G., Selekwa, M.F. & Dunlap, D. D. (2008). The virtual wall approach to limit cycle avoidance for unmanned ground vehicles. *Proc. Robotic and Autonomous System*, vol. 56, pp.645-657

Oreback, A. & Christensen, H. I. (2003). Evaluation of architectures for mobile robotics. *Autonomous Robots*, Vol.14, pp. 33-49

Petti, S. & Fraichard, T. (2005). Safe motion planning in dynamic environments. *IEEE IROS*.

Posadas, J. L., Poza, J.L., Sim, J.E., Benet, G. & Blanes, F. (2008). Agent-based distributed architecture for mobile robot control. *Engineering Applications of Artificial Intelligence*, Vol.21, pp. 805-823

Pruski, A. & Rohmer, S. (1997). Robust path planning for non-holonomic robots.*J. Intell. Robot.Syst: Theory Appl*, 18(4), pp. 329-350

Quinlan, S., Khatib, O. (1993). Elastic bands: Connecting path planning and control. *Proc. IEEE International Conference on Robotics and Automation*, pp. 802-807

Ros, R. (2005). A CBR system for autonomous robot navigation. *Proc. Frontiers in Artificial Intelligence and Applications*, vol.131, pp. 299-306

Sahota, M. K. (1994). Reactive Deliberation: An Architecture for Real Time Intelligent Control in Dynamic Environments. *Proceedings of the AAAI*, pp. 1303–1308.

Sajotti, A., Konolige, K. & Ruspini, E. H. (1995). A multi valued-logic approach to integrating planning and control. *Artificial Intelligence,* pp. 481-526

Schwartz, J. T. & Sharir, M. (1983). General techniques for computing topological properties of algebraic manifolds. *Advances in Applied Mathematics*, 12, 1983.

Selekwa, M. F., Dunlap, D., Shi, D. & Collins, E. (2008). Robot navigation in very cluttered environments by preference-based fuzzy behaviours. *Proc.Robotics and Autonomous Systems*, pp.231-246

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. *Proc. International Conference onRobotics and Automation*, pp.2737-2742

Stachniss, C. & Burgard, W. (2002). An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. *Proc. IEEE International Conference on Intelligent Robots and Systems*, pp.508-513

Stentz, A. (1995). The focussed d* algorithm for real-time re-planning. *Proceedings of the International Joint Conference onArtificialIntelligence*.

Thrun, S. (1998). Bayesian landmark learning for mobile robot navigation. *Machine Learning,*Vol.33 (1), pp. 41-76

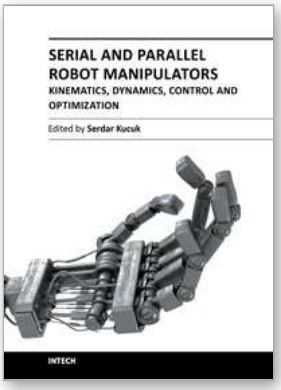
Ulrich, I. & Borenstein, J. (2000). VFH: Local obstacle avoidance with look-ahead verification. *Proc. IEEE International Conference on Robotics and Automation*, pp.2505-2511

Wang, M. & Liu, J. N. K. (2005). Fuzzy logic based robot path planning in unknown environments. *Proc. Internat. Conf.on Mach. Learn. And Cybernet*, pp.818-818

Yang, X., Moallem, M. & Patel, R. V. (2006). A layered goal-oriented fuzzymotion planning strategy for mobile robot navigation. *IEEE Trans.Syst, Man Cyber Part B: Cybernetics*, 35(6), pp. 1214-1224.

Zadeh, L. A. (1973). Outline of a New Approach to the Analysis of Complex Systems and Decision-Making Approach. *IEEE Trans. on Systems, Man, and Cybernetics SME-3(1),* pp. 28-45

Zhu, A. & Yang, S. X.(2004). A fuzzy logic approach to reactive navigation of behaviour-based mobile robots. *Proc. IEEE International Conference on Robotics and Automation*, pp.5045-5050



Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization

Edited by Dr. Serdar Kucuk

ISBN 978-953-51-0437-7
Hard cover, 458 pages
Publisher InTech
Published online 30, March, 2012
Published in print edition March, 2012

The robotics is an important part of modern engineering and is related to a group of branches such as electric & electronics, computer, mathematics and mechanism design. The interest in robotics has been steadily increasing during the last decades. This concern has directly impacted the development of the novel theoretical research areas and products. This new book provides information about fundamental topics of serial and parallel manipulators such as kinematics & dynamics modeling, optimization, control algorithms and design strategies. I would like to thank all authors who have contributed the book chapters with their valuable novel ideas and current developments.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821