10

Industrial and Mobile Robot Collision–Free Motion Planning Using Fuzzy Logic Algorithms

Tzafestas S.G. and Zavlangas P.

1. Introduction

Motion planning is a primary task in robot operation, where the objective is to determine collision-free paths for a robot that works in an environment that contains some moving obstacles (Latombe, 1991; Fugimura, 1991; Tzafestas, 1999). A moving obstacle may be a rigid object, or an object with joints such as an industrial manipulator. In a persistently changing and partially unpredictable environment, robot motion planning must be on line. The planner receives continuous flow of information about occurring events and generates new commands while previous planned motions are being executed. Off – line robot motion planning is a one – shot computation prior to the execution of any motion, and requires all pertinent data to be available in advance. With an automatic motion planner and appropriate sensing devices, robots can adapt quickly to unexpected changes in the environment and be tolerant to modeling errors of the workspace. A basic feature of intelligent robotic systems is the ability to perform autonomously a multitude of tasks without complete a priori information, while adapting to continuous changes in the working environment.

Clearly, both robotic manipulators and mobile robots (as well their combination, i.e. mobile manipulators (Seraji, 1998; Tzafestas & Tzafestas, 2001)) need proper motion planning algorithms. For the robotic manipulators, motion planning is a critical aspect due to the fact that the end effector paths have always some form of task constraints. For example, in arc welding the torch may have to follow a complex 3-dimensional path during the welding process. Specifying manually such paths can be tedious and time consuming. For the mobile robots (indoor and outdoor robots) motion planning and autonomous navigation is also a critical issue, as evidenced by applications such as office cleaning, cargo delivery, autonomous wheel chairs for the disabled,etc.

Our purpose in this chapter is to present a solution of the motion planner design problem using fuzzy logic and fuzzy reasoning. Firstly, the case of industrial robotic manipulators is considered, and then the class of mobile robots is treated. The methodology adopted is primarily based on some recent results

Source: Industrial-Robotics-Theory-Modelling-Control, ISBN 3-86611-285-8, pp. 964, ARS/plV, Germany, December 2006, Edited by: Sam Cubero

derived by the authors (Moustris & Tzafestas, 2005; Zavlangas & Tzafestas, 2000; 2001; 2002). To help the reader appreciate the importance of the techniques presented in the chapter, a short review is first included concerning the general robot motion planning problem along with the basic concepts and some recent results. A good promising practical approach is to use fuzzy logic along the path of behavior–based system design which employs Brooks' subsumption architecture (Brooks, 1986; Izumi & Watanabe, 2000; Topalov & Tzafestas, 2001; Watanabe et al., 1996; Watanabe et al., 2005).
Section 2 provides the overview of robot motion planning for industrial and mobile robots. Section 3 presents the authors' technique for industrial manipulators' fuzzy path planning and navigation. Section 4 extends this technique to mobile robots and discusses the integration of global and local path planning and navigation. Global path planning uses topological maps for representing the robot's environment at the global level, in conjunction with the potential field method. Section 5 presents a representative set of experimental results for the SCARA Adept 1 robotic manipulator and the Robuter III mobile robot (which can be equipped with a robotic arm). Results for both local and global path planning / navigation are included for the Robuter III robot. Finally, a general discussion on the results of the present technique is provided, along with some indications for future directions of research.

2. Robot Motion Planning : An Overview

2.1 Review of Basic Motion Planning Concepts

Robot motion planning techniques have received a great deal of attention over the last twenty years. It can roughly be divided into two categories : *global and local*. Most of the research in global techniques has been focused *on off-line planning* in static environments. A plan is then computed as a geometric path. An important concept developed by this research is the *Configuration space* or *C-space* of a robot (Latombe, 1991; Lozano-Perez, 1983).
The global techniques, such as *road map* (Latombe, 1991), cell *decomposition* (Latombe, 1991) and *potential fields* methods (Khatib, 1986), generally assume that a complete model of the robot's environment is available.
The *roadmap approach* to path planning consists of capturing the connectivity of the robot's free space in a network of one-dimensional curves (called the roadmap), lying in the free space. Once the roadmap has been constructed, it is used as a set of standardized paths. Path planning is thus reduced to connecting the initial and goal configuration to points in the roadmap and searching it for a path between these points (Latombe, 1991).

Cell decomposition methods are perhaps the motion planning methods that have been most extensively studied so far (Latombe, 1991). They consist of decomposing the robot's free space into simple regions, called *cells*, such that a path between any two configurations in a cell can be easily generated. A nondirected graph representing the adjacency relation between the cells is then constructed and searched. This graph is called the *connectivity graph*. Its nodes are the cells extracted from the free space and two nodes are connected by a link if only the corresponding cells are adjacent. The outcome of the search is a sequence of cells called a *channel*. A continuous free path can be computed from this sequence (Latombe, 1991). A straightforward approach to motion planning is to discretize the *configuration space* into a fine regular grid of configurations and to search this grid for a free space. This approach requires powerful heuristics to guide the search. Several types of heuristics have been proposed. The most successful ones take the form of functions that are interpreted as *potential fields* (Latombe, 1991). The robot is represented as a point in configuration space, moving under the influence of an artificial potential produced by the goal configuration and the C-obstacles. Typically, the goal configuration generates an *"attractive potential"* which pulls the robot towards the goal, and the C-obstacles produce a *"repulsive potential"* which pushes the robot away from them. The generated gradient of the total potential is treated as an artificial force applied to the robot. At every configuration, the direction of this force is considered to be the most promising direction of motion.

The advantage of global approaches lies in the fact that a complete trajectory from the starting point to the target point can be computed *off-line*. However, global approaches are not appropriate for fast obstacle avoidance. Their strength is *global path planning*. Additionally, these methods were proven problematic when the global world model is inaccurate, or simply not available, as it is typically the case in the most populated environments. Some researchers have shown how to update global world models based on sensory inputs, using probabilistic representations. A second disadvantage of global methods is their low speed due to the inherent complexity of robot motion planning. This is particularly the case if the underlying world model changes with time, because of the resulting requirement for repeated adjustments of the global plan. In such cases, planning using a global model is usually too expensive to be done repeatedly.

Local approaches, on the other hand, use only a small fraction of the world model to generate robot control. This comes at the obvious disadvantage that they cannot produce optimal solutions. Local approaches are easily trapped at local minima. However, the key advantage of local techniques over global ones lies in their low computational complexity, which is particularly important when the world model is updated frequently based on sensor information. For example, potential field methods determine the next step by assuming that obstacles assert negative forces on the robot, and that the target location asserts a

positive force. These methods are extremely fast, and they typically consider only a small subset of obstacles close to the robot. However, such methods have often failed to find trajectories between closely spaced obstacles; they also can produce oscillatory behaviour in narrow spaces.

2.2 Motion Planning of Mobile Robots

To be useful in the real world, mobile robots need to move safely in unstructured environments and achieve their given goals despite unexpected changes in their surroundings. The environments of real robots are rarely predictable or perfectly known so it does not make sense to make precise plans before moving. The robot navigation problem can be decomposed into the following two problems (Ratering & Gini, 1995) :

- *Getting to the goal.* This is a global problem because short paths to the goal generally cannot be found using only local information. The topology of the space is important in finding good routes to the goal.
- *Avoiding obstacles.* This can often be solved using only local information, but for an unpredictable environment it cannot be solved in advance because the robot needs to sense the obstacles before it can be expected to avoid them.

Over the years, robot collision avoidance has been a component of high-level controls in hierarchical robot systems. Collision avoidance has been treated as a planning problem, and research in this area was focused on the development of collision-free path planning algorithms. These algorithms aim at providing the low-level control with a path that will enable the robot to accomplish its assigned task free from any risk of collision. However, this places limits on the robot's real-time capabilities for precise, fast, and highly interactive operations in a cluttered and evolving environment. Collision avoidance at the low-level control is not intended to replace high-level functions or to solve planning problems. The purpose is to make better use of low-level control capabilities in performing real-time operations. A number of different architectures for autonomous robot navigation have been proposed in the last twenty years (Latombe, 1991; Fugimura, 1991; Tzafestas, 1999). These include hierarchical architectures that partition the robot's functionalities into high-level (model and plan) and low-level (sense and execute) layers; behaviour – based architectures that achieve complex behaviour by combining several simple behaviour-producing units; and hybrid architectures that combine a layered organization with a behaviour-based decomposition of the execution layer (see e.g., (Izumi & Watanabe, 2000; Watanabe et al., 1996; Topalov & Tzafestas, 2001; Watanabe et al., 2005; Lozano-Perez, 1983; Khatib, 1986; Ratering & Gini, 1995; Erdmann & Lozano-Perez, 1987; Griswold & Elan, 1990; Gil de Lamadrid & Gini, 1990;

Fibry, 1987; Gat, 1991; Sugeno & Nishida, 1985; Yen & Pflunger, 1992)). While the use of hybrid architectures is gaining increasing consensus in the field, a number of technological gaps still remain.

As mentioned in Section 2.1, the classical approaches can be mainly divided into two categories : *global path planning* and *local navigation* (Latombe, 1991). The global approaches preassume that a complete representation of the configuration space has been computed before looking for a path. They are complete in the sense that if a path exists it will be found. Unfortunately, computing the complete configuration space is very time consuming, worst, the complexity of this task grows exponentially as the number of degrees of freedom increases. Consequently, today most of the robot path planners are used off-line. The planner is equipped with a model of the environment and produces a path which is passed to the robot controller for execution. In general, the time necessary to achieve this, is not short enough to allow the robot to move in dynamic environments. The local approaches need only partial knowledge of the robot's workspace. The decisions to move the robot are taken using local criteria and heuristics to choose the most promising direction. Consequently, the local methods are much faster. Unfortunately, they are not complete, it may happen that a solution exists and cannot be found. The local approaches consider planning as an optimization problem, where finding a path to the goal configuration corresponds to the optimization of some given function. As an optimization technique, the local approaches are subject to get trapped in some local minima, where a path to the goal has not been found and from which it is impossible or, at least, very difficult to escape.

From the above, it is very clear, that both global and local techniques have many advantages, as well as important disadvantages. The output of a global path planner is a continuous path along which the robot will not collide with obstacles. However, any model of the real world will be incomplete and inaccurate, thus collisions may still occur if the robot moves blindly along such a path. One conventional application is for the robot to track the global path. More recently, work has been done on increasing the level of competence, by including real-time collision avoidance capabilities. Such local or reactive behaviours operate in real time but cannot solve the global problem of moving to an arbitrary goal. It is very clear that to built a complete system, the above approaches must be combined. A path planner must provide the robot with a global path to the goal. A local controller then moves the robot along the global path while handling small changes in the environment and unexpected or moving obstacles.

Some researchers have solved the navigation problem by solving these two sub-problems one after the other. A path is first found from the robot's initial position to the goal and then the robot approximates this path as it avoids obstacles. This method is restrictive in that the robot is required to stay fairly close to or perhaps on a given path. This would not work well if the path goes

through a passageway which turns out to be blocked by an unforeseen obstacle. Solutions that are only local or reactive (Brooks, 1986) can lead the robot into local minima traps. Solutions that assume a priori knowledge of the position of the obstacles (e.g. (Fugimura, 1991; Erdmann & Lozano-Perez, 1987)), or select a path using only information on stationary obstacles, and determine the speed of the robot while following the path (e.g. (Griswold & Elan, 1990)), or solutions that require the robot to stay within some distance from its assigned path, while avoiding unknown moving obstacles (e.g., (Gil de Lamadrid & Gini, 1990)), are not always sufficiently flexible to deal with situations in which an obstacle blocks a path to the goal.

In the general case, knowledge of the environment is partial and approximate; sensing is noisy; the dynamics of the environment can only be partially predicted; and robot's hardware execution is not completely reliable. Though, the robot needs to make decisions and execute actions at the time-scale of the environment. Classical planning approaches have been criticized for not being able to adequately cope with this situation, and a number of *reactive approaches* to robot control have been proposed (e.g. (Fibry, 1987; Gat, 1991)), including the use of fuzzy control techniques (e.g., (Martinez et al., 1994; Seraji & Howard, 2002; Sugeno & Nishida, 1985; Yen & Pflunger, 1992)). Reactivity provides immediate response to unpredicted environmental situations by giving up the idea of reasoning about future consequences of actions. Reasoning about future consequences (sometimes called *"strategic planning"*), however, is still needed in order to intelligently solve complex tasks.

Some recent developments in mobile robot navigation using fuzzy logic algorithms include (Parhi, 2005) and (El Hajjaji, 2004). In (Parhi, 2005) the fuzzy controller enables the robot to avoid obstacles that are not mobile robots. The fuzzy rules steer the robot according to whether there are obstacles or targets around it and how far they are from it. Fuzzy logic is suitable for this problem because this information is usually not precisely known. In (El Hajjaji, 2004) the case of 4-wheel automotive vehicles is considered which are modeled by a Takagi-Sugeno type of model. The fuzzy controller is then designed to improve the stability of the vehicle. A comprehensive study of the kinematics of nonholonomic mobile manipulators composed by an n_a-joint robotic arm and a nonholonomic mobile platform having two independently driven wheels is provided in (Bayle et al., 2003). Finally, a new approach to the navigation of mobile robots for dynamic obstacle avoidance is proposed in (Belkhous et al., 2005). This approach merges the static and dynamic modes of path planning to provide an algorithm giving fast optimal solutions for static environments, and produces a new path whenever an unanticipated situation occurs.

3. Fuzzy Path Planning and Navigation of Industrial Manipulators

3.1 Fuzzy Obstacle Avoidance

Here, we will outline a technique developed by the authors (Zavlangas & Tzafestas, 2000), which has been primarily influenced by Khatib's (1986) artificial potential field method and the subsumption architecture developed by Brooks (1986). The local navigation approach is chosen because our main goal is to develop an on-line planner for fast collision – free trajectory generation. The proposed local navigator was implemented and applied to several practical scenarios. Our experimental results, some of which will be presented in Section 5, are very satisfactory. The technique is based on separate fuzzy logic-based obstacle avoidance units, each controlling one individual link l_j, $j = 1,\ldots,n$ Each unit has two principal inputs :

1. the distance between the link and the nearest obstacle d_j, and
2. the difference between the current link configuration and the target configuration,

$\theta_j - \theta_{j,target} = \Delta\theta_j$.

The output variable of each unit is the motor command τ_j. All these variables can be positive or negative, i.e., they inform both about the magnitude and the sign of displacement relative to the link – left or right. The motor command which can be interpreted as an actuation for the link motor is fed to the manipulator at each iteration (Figure 1). For the calculation of the distance, the only obstacles considered are those which fall into a bounded area surrounding each link and move along with it. In this implementation, each such area is chosen to be a cylindrical volume around each link. The area is as long as the link and reaches up to a predefined horizon. This area can be seen as a simplified model for the space scanned by ranging sensors (for example, ultra-sonic sensors) attached to the sides of a link (Pedrycz, 1995). Of course, other shapes to describe the 3-dimensional scan areas are conceivable. It is, for example, advisable to deal with the blind zones near the joints when the magnitude of the angles is large so as to assure that small moving obstacles are not missed by the algorithm (Pedrycz, 1995).

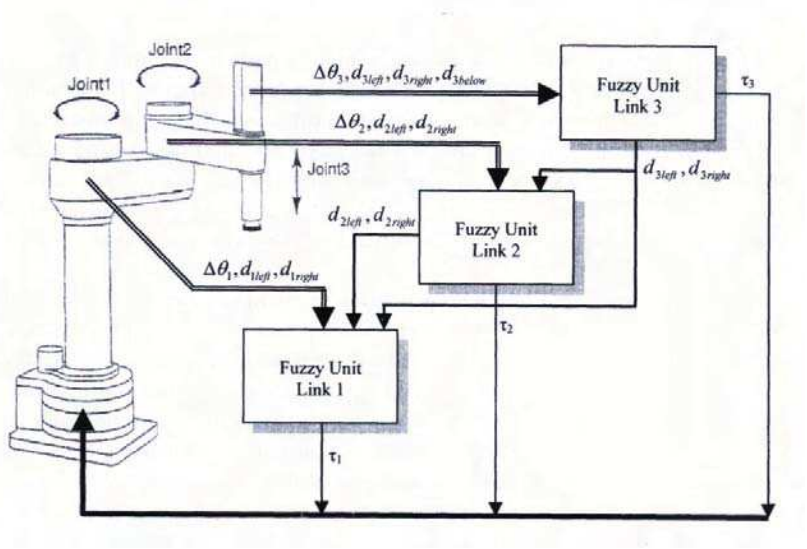


Figure 1. The Adept 1 industrial robotic manipulator connected to the corresponding fuzzy units.

Each fuzzy unit receives via an input the difference between target and actual configuration, and, via a second input, two values in a sequential way representing the distance between the corresponding link and the nearest obstacle on the left and on the right of this link. If no obstacle is detected inside the scan area, the fuzzy unit is informed of an obstacle in the far distance. Additionally, proximal units are informed about obstacles in the vicinity of more distal links. Besides an input from ultrasonic sensors, a camera can be used to acquire the environment. Either a stationary camera or a set of cameras which oversee the entire workspace can be utilised (Pedrycz, 1995; Jaitly & Fraser, 1996). The task of each fuzzy unit is to provide a control function which produces an appropriate motor command from the given inputs. In broad lines, the control function can be described as follows : on the one hand the function has to lead the corresponding link to its attracting end – position; on the other hand, it has to force the link to back up when approaching an obstacle which conveys a repelling influence. The fuzzy-rule-base (which represents the control function in each fuzzy unit) is built up by using common sense rules.

In our particular implementation, at each iteration the distances of the nearest obstacle on the left $\left(d_{jleft}\right)$ and on the right $\left(d_{jright}\right)$ of link l_j are fed sequentially into the fuzzy unit. This process could also be carried out in a parallel fashion where two equivalent fuzzy controllers compute the response for the left and the right obstacle separately. The resulting two motor commands are superimposed, hence, both obstacles influence the final motor command which is applied to the link. The use of this method guarantees that the repulsion caused by one obstacle on one side of the link does not result in a collision with a nearby obstacle on the opposite side. Only those obstacles are considered which are the nearest on the left and right.

In addition, fuzzy units of distal links communicate information about the distance to their nearest obstacles on the left and right to units of more proximal links. Once sent to fuzzy units of more proximal links, this information can be used by the decision process of those units to slow down or even reserve the motion of the more proximal links. Without this propagation of information the control strategy might fail in situations where one obstacle is "known" only to a fuzzy unit of a distal link, while proximal links continue their motion based on their local environment dictating an adverse motion for the rest of the arm. This is especially important, since the same change of angle occurring at a proximal link and at a distal link produces a different velocity at the manipulator's tip. Thus, the motion of a proximal link might not be sufficiently compensated by an adverse motion at a more distal link. Fuzzy units are only fed with the distance values of those obstacles which are inside the scan range. If no obstacle is detected inside a scan range, the fuzzy unit is informed of an obstacle which is *far left* or *far right,* respectively.

3.2 The Fuzzy Navigation Algorithm

The first input of each fuzzy unit is the difference between the actual angle and the target angle, $\theta_j - \theta_{j,target} = \Delta\theta_j \in \Theta_j$, $j = 1,\ldots,n$ (n is the number of links). The value $\Delta\theta_j$ is positive if the target is on the right, and negative if the target is on the left. The second input receives values describing the distance between link l_j and the nearest obstacles on the left and right in the "scanned" region, $d_j \in D_j$. An obstacle on the left produces a negative distance value, while an obstacle on the right produces a positive one. The single output is the motor command $\tau_j \in T_j$. A positive motor command moves the link to the left and a negative one to the right (Althoefer, 1996; Althoefer & Fraser, 1996).
Each universe of discourse D_j can be partitioned by fuzzy sets $\mu_1^{(j)},\ldots,\mu_{pj}^{(j)}$. Each of the sets $\mu_{\tilde{p}j}^{(j)}$, $\tilde{p}_j = 1,\ldots,p_j$, represents a mapping $\mu_{\tilde{p}j}^{(j)}\left(d_j\right): D_j \to [0,1]$ by which d_j is associated with a number in the interval $[0,1]$ indicating to what degree d_j is a member of the fuzzy set. Since d_j is a signed value, "*close_left*", for example, may be considered as a particular fuzzy value of the variable distance and each d_j is assigned a number $\mu_{close\_left}\left(d_j\right) \in [0,1]$ which indicates the extent to which that d_j is considered to be close_left (Mamdani & Assilian, 1981). In an equivalent way, fuzzy sets $\nu_1^{(j)},\ldots,\nu_{qj}^{(j)}$ can be defined over the universe of discourse Θ_j. The Fuzzy Navigator is based on the Mamdani fuzzy model (Mamdani, 1974) and has an output set which is partitioned into fuzzy sets $\tau_{\tilde{r}j}$.
There is a variety of functions that can be employed to represent fuzzy sets (Mamdani & Assilian, 1981). In the proposed controller asymmetrical trape-

zoidal functions were employed to represent the fuzzy sets. The parameters, $ml^{(j)}$, $mr^{(j)}$, which are the x-coordinates of the left and right zero crossing, respectively, and $mcl^{(j)}$, $mcr^{(j)}$, which describe the x-coordinate (left and right) where the fuzzy set becomes 1, define the following trapezoidal functions :

$$\mu_{p_j}^{(j)}\left(d_j\right)=\begin{cases}\min\left(\dfrac{\left(d_j-ml_{p_j}^{(j)}\right)}{\left(mcl_{p_j}^{(j)}-ml_{p_j}^{(j)}\right)},0\right) & if\ ml_{p_j}^{(j)}<d_j\le mcl_{p_j}^{(j)},\\ 1 & if\ mcl_{p_j}^{(j)}<d_j\le mcr_{p_j}^{(j)},\\ \min\left(\dfrac{\left(d_j-mr_{p_j}^{(j)}\right)}{mcr_{p_j}^{(j)}-mr_{p_j}^{(j)}},0\right) & if\ mcr_{p_j}^{(j)}<d_j\le mr_{p_j}^{(j)}.\end{cases} \tag{1}$$

As commonly done, the trapezoidal functions are continued as constant values of magnitude 1 at the left and right side of the interval (Equations (2) and (3)) :

$$\mu_{p_j}^{(j)}\left(d_j\right)=\begin{cases}1 & if\ ml_1^{(j)}<d_j\le mcl_1^{(j)},\\ 1 & if\ mcl_1^{(j)}<d_j\le mcr_1^{(j)},\\ \min\left(\dfrac{\left(d_j-mr_1^{(j)}\right)}{\left(mcr_1^{(j)}-mr_1^{(j)}\right)},0\right) & if\ mcr_1^{(j)}<d_j\le mr_1^{(j)}.\end{cases} \tag{2}$$

and

$$\mu_{p_j}^{(j)}\left(d_j\right)=\begin{cases}\min\left(\dfrac{\left(d_j-ml_{p_j}^{(j)}\right)}{\left(mcl_{p_j}^{(j)}-ml_{p_j}^{(j)}\right)},0\right) & if\ ml_{p_j}^{(j)}<d_j\le mcl_{p_j}^{(j)},\\ 1 & if\ mcl_{p_j}^{(j)}<d_j\le mcr_{p_j}^{(j)},\\ 1 & if\ mcr_{p_j}^{(j)}<d_j\le mr_{p_j}^{(j)}.\end{cases} \tag{3}$$

Note, that for the two fuzzy sets of Equations (2) and (3) :

$$mcr_{p_j}^{(j)}=mr_{p_j}^{(j)},\ mcl_{p_j}^{(j)}=ml_{p_j}^{(j)} \tag{4}$$

The fuzzy sets for $\Delta\theta_j$ can be defined in the same way. The fuzzy sets $m_1^{(j)},\ldots,m_p^{(j)}$ and $n_1^{(j)},\ldots,n_q^{(j)}$ for link 2 as functions of the two inputs d_j and $\Delta\theta_j$ are shown in Figure 2, together with the output fuzzy set.

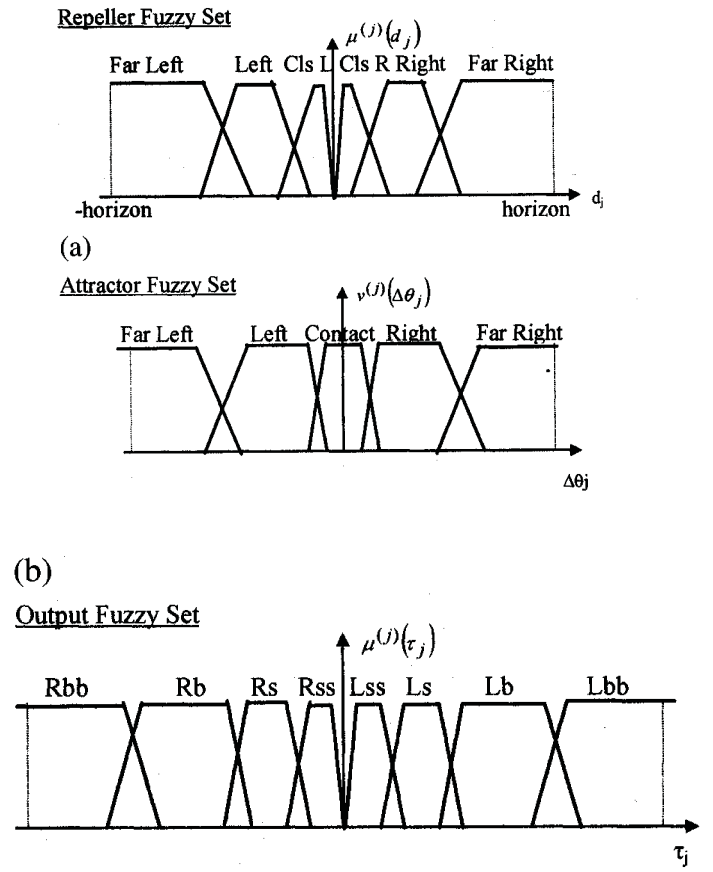


Figure 2. Repeller, attractor and output fuzzy sets for link 2.

The first diagram (a) is the fuzzy set of the distance between the link and the obstacle, and the second one (b) is the fuzzy set of the difference between the actual and target configuration. The output fuzzy set is shown in the third diagram (c).

Additionally to the two inputs d_j and $\Delta\theta_j$, each fuzzy unit (apart from the most distal one) uses the distance fuzzy sets of more distal links $\mu_{\tilde{p}_{j+1}}^{(j+1)},\ldots,\mu_{\tilde{p}_n}^{(n)}$ for decision making to assure that proximal links are slowed down, in case a more distal link is about to collide with an obstacle. Thus, each unit uses the following fuzzy sets : $\mu_{\tilde{p}_k}^{(k)}$, $k=j+1,\ldots,n$, and $\nu_{\tilde{q}_j}^{(j)}$. Each of the fuzzy sets $\mu_{\tilde{p}_k}^{(k)}$ and $\nu_{\tilde{q}_j}^{(j)}$ are associated with linguistic terms $A_{\tilde{p}_j}^{(j)}$ and $B_{\tilde{q}_j}^{(j)}$, respectively. Thus, for link l_j the linguistic control rules $R_1^{(j)},\ldots,R_{r_j}^{(j)}$, which constitute the rule base, can be defined as :

$$R_{\tilde{r}_j}^{(j)} : IF\ d_j\ is\ A_{\tilde{p}_j}^{(j)}\ AND \ldots AND\ d_n\ is\ A_{\tilde{p}_n}^{(n)}\ AND\ \Delta\theta_j\ is\ B_{\tilde{q}_j}^{(j)}\ THEN\ \tau_{\tilde{r}_j}$$

where $\tilde{r}_j = 1,\ldots,r_j$, r_j, is the number of rules for the fuzzy unit of link l_j, and $\tau_{\tilde{r}_j}$ is a numerical entry in the rule base used in the defuzzification process (Equation (5)). The most popular methods to calculate the fuzzy intersection (fuzzy – AND) are the *minimum* and *product* operators (Tzafestas & Venetsanopoulos, 1994; Zadeh, 1965; 1973). If the *minimum* operator is used, the minimum of the inputs is chosen. If the *product* operator is chosen, the inputs are multiplied with each other. While the result of the first approach contains only one piece of information, the second approach produces results which are influenced by all inputs (Brown, 1994).

Here, the fuzzy intersection is calculated by using the product operator "*" :

$$\sigma_{\tilde{r}_j} = \mu_{\tilde{p}_j,\tilde{r}_j}^{(j)}\left(d_j\right) \bigcap \cdots \bigcap \mu_{\tilde{p}_n,\tilde{r}_n}^{(n)}\left(d_n\right) \bigcap \nu_{\tilde{q}_j,\tilde{r}_j}^{(j)}\left(\Delta\theta_j\right) = \mu_{\tilde{p}_j,\tilde{r}_j}^{(j)}\left(d_j\right) * \cdots * \mu_{\tilde{p}_n,\tilde{r}_n}^{(n)}\left(d_n\right) * \nu_{\tilde{q}_j,\tilde{r}_j}^{(j)}\left(\Delta\theta_j\right)$$

The output of the unit is given by the centroid defuzzification over all rules (Kosko, 1992) :

$$\tau_j = \frac{\sum_{r_j=1}^{r_j} \tau_{r_j} * \mu_{p_j,r_j}^{(j)}\left(t_{r_j}\right)}{\sum_{r_j=1}^{r_j} \mu_{p_j,r_j}^{(j)}\left(t_{r_j}\right)} \tag{5}$$

The fuzzy rule base for link 2 is displayed in Table I.

	Far left	Left	Close left	Close right	Right	Far right
Far left	Ls	Rs	Rbb	Lb	Ls	Ls
Left	Lss	Rs	Rbb	Ls	Lss	Lss
Contact	nil	nil	nil	nil	nil	nil
Right	Rss	Rss	Rs	Lbb	Ls	Rss
Far right	Rs	Rs	Rb	Lbb	Ls	Rs
Lss, Rss : very small to the left / right Ls, Rs : small to the left / right Lb, Rb : big to the left / right Lbb, Rbb : very big to the left / right						

Table 1. FAM matrix (Fuzzy Associative Memory) for link 2

4. Fuzzy Path Planning and Navigation of Mobile Robots

4.1 General Description

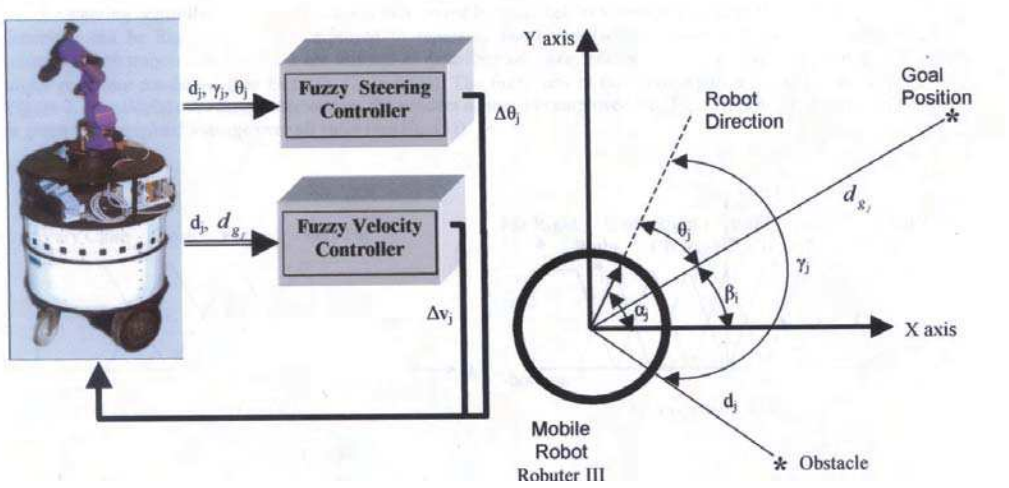
Basically, the technique to be described for mobile fuzzy path planning and navigation is the same with that described in Section 3 for the case of industrial manipulators, i.e. it is based on Khatib's potential field method (Khatib, 1986) and on Brooks subsumption structure (Brooks, 1986).
Khatib computes an artificial potential field that has a strong repelling force in the vicinity of obstacles and an attracting force produced by the target location. The superposition of the two forces creates a potential field, which incorporates information about the environment. Following the steepest gradient from a start position, a path can be found that guides the robot to the target position avoiding obstacles. In our approach the amount of computation, that is required, is reduced by using only the nearest obstacles to determine the direction of motion. The main idea of Brooks is a collection of modules, which are interconnected on different layers with different hierarchies. These modules are for example *wall following, obstacle avoidance, goal reaching,* etc. Depending on sensory input, a module becomes active and generates a command for the robot. While Brooks' system resembles an expert system where for any input signal one specific reaction module or a specific combination of modules is active, our fuzzy approach is a parallel processing strategy where each input contributes to the final decision (see Section 3.2).
The technique is based on two fuzzy – based controllers, one for **steering control,** and the other for **velocity control**. The steering controller has three principal inputs :

1) the distance between the robot and the nearest obstacle d_j,
2) the angle between the robot and the nearest obstacle γ_j, and
3) the angle between the robot's direction and the straight line connecting the current position of the robot and the goal configuration $\theta_j = \alpha_j - \beta_j$, where β_j is the angular difference between the straight line connecting the robot's current position and the goal configuration, and α_j is the current direction of the robot (see Fig. 3).

The output variable of this unit is the required change of angle $\Delta\theta_j$, and can be considered as a command for the robot's steering actuators. The velocity controller has two principal inputs :

1) the distance between the robot and the nearest obstacle d_j,
2) the distance between the robot and the goal configuration d_{g_j}.

The output variable of this unit is an acceleration command Δv_j, and can be considered as a command for the robot's drive actuators. All these variables can be positive or negative, i.e. they do not only inform about the magnitude, but also about the sign of displacement relative to the robot – left or right. The motor commands are fed to the mobile platform at each iteration.



(a) (b)

Figure3. (a) The Robosoft Robuter III mobile robot of IRAL / NTUA connected to the corresponding fuzzy-based obstacle avoidance unit.

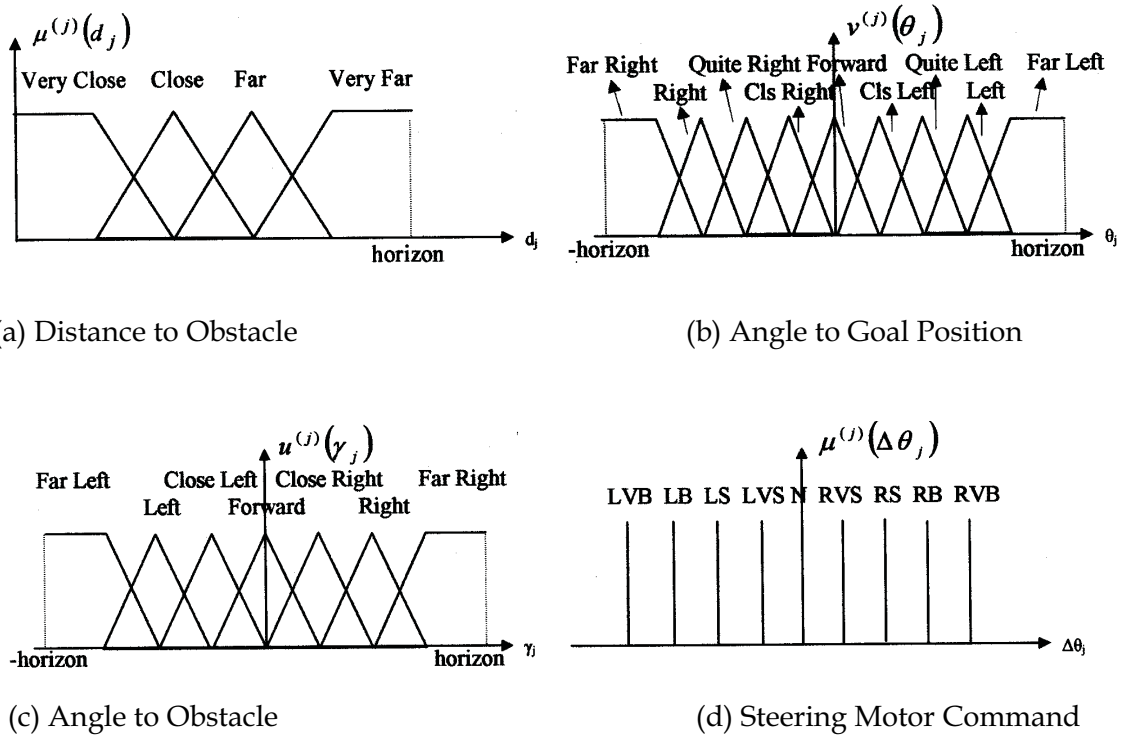
This steering control unit receives via an input the angle between the robot's direction and the straight line connecting the current position of the robot and the goal configuration $\theta_j = \alpha_j - \beta_j$, and the distance and the angle to the nearest obstacle $\left(d_j, \gamma_j\right)$ (see (b)). If no obstacle is detected inside the scan area, the fuzzy unit is informed of an obstacle in the far distance. The output variable of this unit is the required change of angle $\Delta\theta_j$, and can be considered as a command for the robot's steering actuators. The velocity control unit has two inputs : the distance between the robot and the nearest obstacle d_j, and the distance between the robot and the goal configuration d_{g_j}. The output variable of this unit is an acceleration command Δv_j and can be considered as a command for the robot's driving actuators.

The obstacles considered are those that fall into a confined area surrounding the robot and moving along with it. Here, this area is chosen to be a cylindrical volume around the mobile platform. This area is regarded as a simplified model for the space scanned by ranging sensors (for example ultrasonic sensors) attached to the sides of the robot (Pedrycz, 1995). Besides an input from ultrasonic sensors, a camera can also be used to acquire the environment. Mo-

bile robots are usually equipped with a *pan / tilt platform* where a camera is mounted. This camera can be utilized as shown in (Pedrycz, 1995; Jaitly et al., 1996; Kamon & Rivlin, 1997). If no obstacle is detected inside the scan area, the fuzzy units are informed of an obstacle in the far distance. The task of the fuzzy units is to provide a control function, that produces appropriate motor commands from the given inputs. This control function on the one hand has to lead the mobile robot to its attracting goal–position, and on the other hand it has to force the robot to back up when approaching an obstacle which conveys a repelling influence. The fuzzy–rule – base (which represents the control function in the fuzzy unit) is constructed using common sense rules or by a neural network training algorithm (see e.g., (Tzafestas & Zavlangas, 1999)). An alternative fuzzy motion control scheme of mobile robots which employs the sliding – mode control principle can be found in (Rigatos & Tzafestas, 2000).

4.2 Detailed Description

As mentioned in Section 4.1 the proposed methodology is based on two fuzzy – based controllers, one for **steering control,** and the other for **velocity control**. The fuzzy controllers here are based on the functional reasoning control principles developed by Sugeno (see, for example, (Sugeno, 1985; Sugeno & Murakami, 1984)). For the steering controller, each input space is partitioned by fuzzy sets as shown in Fig. 4. Here, asymmetrical triangular and trapezoidal functions are utilized to describe each fuzzy set, which allow a fast computation, essential under real – time conditions (see Eqs. (6) – (8) and (11)). The fuzzy sets of the three inputs d_j, γ_j and θ_j are depicted in Figure 4. To calculate the fuzzy intersection, the product operator is employed (see Eq (10)). The final output of the unit is given by a weighted average over all rules (see Eq. (11)). Intuitively, the rules for obstacle – avoiding navigation can be written as sentences with three antecedents and one conclusion. This structure lends itself to a tabular representation such as the one shown in Table 2. This table represents the prior knowledge of the problem domain. The tools of fuzzy logic allow us to translate this intuitive knowledge into a control system. To translate Table 2 into fuzzy logic, the universe of discourse D_j which describes the distance $d_j \in D_j$ to the obstacle is partitioned by fuzzy sets $\mu_1^{(j)},\ldots,\mu_{p_j}^{(j)}$, where p_j is the number of fuzzy sets. Each set $\mu_{\tilde{p}_j}^{(j)}$, $\tilde{p}_j = 1,\ldots,p_j$, represents a mapping $\mu_{\tilde{p}_j}^{(j)}(d_j): D_j \to [0,1]$ by which d_j is considered as a particular fuzzy value of the variable distance and each d_j is associated with, e.g., a number in the interval [0,1] indicating to what degree d_j is a member of the fuzzy set. Since d_j is a measure of distance, "very_close", it may be assigned a number : $\mu_{very\_close}(d_j) \in [0,1]$ which indicates the extent to which this particular d_j is considered to be "very_close" (Mamdani & Assilian, 1981).



(a) Distance to Obstacle

(b) Angle to Goal Position

(c) Angle to Obstacle

(d) Steering Motor Command

Fig. 4. Fuzzy sets for the mobile robot : (a) distance to obstacle, (b) angle between robot and goal position, (c) angle between robot and obstacle, and (d) steering motor command. Note that the output is not partitioned into fuzzy sets, but consists of crisp values.

v_j/μ_j	very close	close	far	very far
far right	right big	right small	left very big	left very big
right	right big	right small	left big	left big
quite right	right big	right small	left small	left small
close right	right big	right small	left very small	left very small
forward	null	null	null	null
close left	right very small	right very small	right very small	right very small
quite left	right small	right small	right small	right small
left	right big	right big	right big	right big
far left	right very big	right very big	right very big	right very big
This rule-base is a translation of the common – sense knowledge of the problem domain into the language of fuzzy logic. Rows represent the fuzzy measures of the distance to an obstacle, while columns are fuzzy representations of the angle to the goal. Each element of the table can be interpreted as a particular motor actuation command.				

Table 2. A rule – base for the mobile robot when γ_j is "far_left".

Similarly, fuzzy sets $v_1^{(j)},\ldots v_{q_j}^{(j)}$ can be defined over the universe of discourse Θ_j which represents the angle between the robot's direction and the straight line connecting the current position of the robot and the goal configuration : $\theta_j \in \Theta_j$. Finally, fuzzy sets $u_1^{(j)},\ldots,u_{g_j}^{(j)}$ can be defined over the universe of discourse Γ_j that represents the angle to the nearest obstacle $\gamma_j \in \Gamma_j$. In contrast to the Mamdani's controller, Sugeno's controller (see (Sugeno, 1985; Sugeno & Murakami, 1984; Tzafestas & Zikidis, 2001)), of which ours is an example, has an output set which is not partitioned into fuzzy sets (see Fig. 2). Thus, the rule conclusions merely consist of scalars $\Delta\theta_{\tilde{r}_j}$, $\tilde{r}_j = 1,\ldots,r_j$.
The fuzzy sets $\mu_{\tilde{p}_j}^{(j)}$, $\tilde{p}_j = 1,\ldots,p_j$, are described by asymmetrical triangular and trapezoidal functions. Defining the parameters, $ml_{\tilde{p}_j}^{(j)}$ and $mr_{\tilde{p}_j}^{(j)}$ as the x-co-ordinates of the left and right zero crossing respectively, and $mcl_{\tilde{p}_j}^{(j)}$ and $mcr_{\tilde{p}_j}^{(j)}$ as the x-co-ordinates of the left and right side of the trapezoid's plateau, the trapezoidal functions can be written as :

$$\mu_{\tilde{p}_j}^{(j)}\left(d_j\right)=\begin{cases}\max\left(\left(d_j-ml_{\tilde{p}_j}^{(j)}\right)/\left(mcl_{\tilde{p}_j}^{(j)}-ml_{\tilde{p}_j}^{(j)}\right),0\right) & if\ \ d_j < mcl_{\tilde{p}_j}^{(j)}\\ 1 & if\ mcl_{\tilde{p}_j}^{(j)} \le d_j \le mcr_{\tilde{p}_j}^{(j)}\\ \max\left(\left(d_j-mr_{\tilde{p}_j}^{(j)}\right)/\left(mcr_{\tilde{p}_j}^{(j)}-mr_{\tilde{p}_j}^{(j)}\right),0\right) & if\ d_j > mcr_{\tilde{p}_j}^{(j)}\end{cases} \tag{6}$$

with $\tilde{p}_j = 1,\ldots,p_j$. Triangular functions can be achieved by setting $mcl_{\tilde{p}_j}^{(j)} = mcr_{\tilde{p}_j}^{(j)}$.

At the left and right sides of the interval, the functions are continued as constant values of magnitude one, i.e. :

$$\mu_1^{(j)}\left(d_j\right)=\begin{cases}1 & if\ d_j \le mcr_1^{(j)}\\ \max\left(\left(d_j-mr_1^{(j)}\right)/\left(mcr_1^{(j)}-mr_1^{(j)}\right),0\right) & if\ d_j > mcr_1^{(j)}\end{cases} \tag{7}$$

and

$$\mu_{p_j}^{(j)}\left(d_j\right)=\begin{cases}\max\left(\left(d_j-ml_{p_j}^{(j)}\right)/\left(mcl_{p_j}^{(j)}-ml_{p_j}^{(j)}\right),0\right) & if\ d_j \le mcl_{p_j}^{(j)}\\ 1 & if\ d_j > mcl_{p_j}^{(j)}\end{cases} \tag{8}$$

The fuzzy sets for θ_j and γ_j are defined analogously. Figure 4 shows the fuzzy sets $\mu_1^{(j)},\ldots,\mu_{p_j}^{(j)}$, $v_1^{(j)},\ldots,v_{q_j}^{(j)}$ and $u_1^{(j)},\ldots,u_{g_j}^{(j)}$.