“Queen Bee” genetic optimization of an heuristic based fuzzy control scheme for a mobile robot[1]

Rodrigo A. Carrasco Schmidt
Pontificia Universidad Católica de Chile

***Abstract*—This work presents both a novel control scheme for a mobile robot and an optimization method for improving its performance. The analyzed control problem will be to move a two wheeled robot from an initial posture to a final destination using the minimum amount of time and arriving at a low speed to be able to stop. First the control strategy, based on a fuzzy logic controller for the robot kinematics and a PID controller for the robot dynamics, is presented. The fuzzy controller is then optimized using a new type of genetic algorithm that replies the reproduction method of bees. The optimized fuzzy controller presents an important improvement on its performance. Finally, several optimal controllers are combined together to create an adaptive controller that can handle general cases in an efficient way.**

***Index Terms*—Fuzzy control, Genetic algorithms, Mobile robots, Optimization methods.**

I. Introduction

The use of robotics and mobile automation systems is increasing every year, and with it, the necessity of more robust and flexible products that solve problems efficiently. One of the most difficult robotic systems to create is autonomous vehicles, due to the fact that they have to deal with dynamic and changing environments which make the task very challenging [1].

With the aid of robotic competitions like RoboCup [1], and the acceptance, by the consumers, of new robotic products such as vacuum cleaners or robotic pets, the interest in mobile robotics has increased. This has led to a great number of research studies aiming to improve autonomous vehicles, making them capable of dealing with the surrounding environment. Several solutions have emerged from these studies, from robust guidance mechanisms, to simple robots in colonies, able to help each other to complete a certain task.

One of the most important components of a mobile robot is the control loop, which enables the robot to follow a certain trajectory determined by higher level decision system. This work presents a novel control scheme, consisting of two layers of control systems that are able to work efficiently with the nonlinearities inherent to the mobile robot, but without adding too much extra computational cost. A simple fuzzy logic controller, based on heuristic rules, is presented as a way of dealing with the nonlinear elements of the system, which are optimized afterwards using new genetic techniques.

Evolutionary computational systems are one of the tools that have shown excellent results when used to optimize complex systems [2]-[4]. In this work, a new genetic algorithm that emulates the evolution principles of bee colonies is used as a way of optimizing the position of each membership function, improving through this method, the overall performance of the robot controller [2]. The results of the optimization are analyzed and tested, simulating the system in a Simulink model and showing that the performance of the resulting controller is better than the one of the original fuzzy controller. The optimization is done several times using different destination points to check if the solutions are equivalent. Based on these optimized solutions for specific cases, a new adaptive fuzzy controller is then designed, which generates the best solution for all general cases, but based on the optimized controllers obtained for specific destination points.

II. Mobile Robot Model

Figure 1 shows the mobile robot model with the basic parameters used in the system. The body of the robot is considered to be circular disc of radius b and mass M, with two wheels of radius r and mass m each. The right wheel rotates at an angular speed of $w_1 = \dot{q}_1$, and the left at $w_2 = \dot{q}_2$. Each wheel is connected to an independent DC motor using a gear system of ratio G:1.

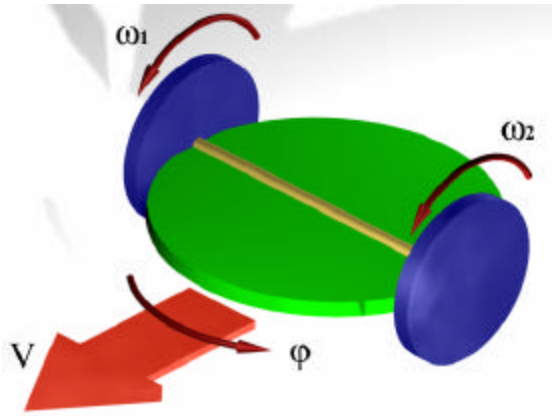

Fig. 1. Robot model showing the main dynamic parameters

[1] This work was presented at the IEEE First Latin American Conference on Robotics and Automation (November 2003)

A. Robot Kinematics

The kinematics equations for the robot relate the state or posture of the robot, with the angular velocities of each wheel. The posture of the robot is defined as the vector $\mathbf{X}=[x\ y\ \varphi]^T$, where x and y are the coordinates of the center of mass of the robot on a reference plane, whereas φ is the angle of the direction of motion of the robot, with respect to the X axis.

Equations 1 and 2 show the relation between the angular speed of each wheel, and the rotational and tangential speed of the robot, as obtained from [5]:

$$\dot{j} = \frac{r\left(\dot{q_1} - \dot{q}_2\right)}{2b} \tag{1}$$

$$V = \frac{r\left(\dot{q_1} + \dot{q}_2\right)}{2} \tag{2}$$

The posture elements x and y are obtained projecting the velocity of the robot on the X and Y axes. Equations 3, 4, and 5 give the position of the center of mass and the angle of direction of the robot due to the speed of the wheels:

$$x(t) = x(0) + \int_o^t \frac{r\left(\dot{q_1}(t) + \dot{q}_2(t)\right)}{2}\cos\left(j\ (t)\right)dt \tag{3}$$

$$y(t) = y(0) + \int_o^t \frac{r\left(\dot{q_1}(t) + \dot{q}_2(t)\right)}{2}\sin\left(j\ (t)\right)dt \tag{4}$$

$$j\ (t) = j\ (0) + \int_o^t \frac{r\left(\dot{q_1}(t) - \dot{q}_2(t)\right)}{2b}dt \tag{5}$$

These equations also make the system non-linear, due to the trigonometric equations needed for the projection of the velocity over each axis.

B. Robot Dynamics

The dynamic equations of the robot relate the torque applied to the wheels, with the angular acceleration they acquire, considering the mass inertia of the different elements in the model. These equations can be deduced using the Lagrangian formulation, which is based on the calculation of the energy of the system [6]. The total energy of the robot can be calculated as the sum of the kinetic energy of the body and the kinetic energy of each wheel, shown on equation 6, whereas the potential energy is not used, as the robot is considered to move on a single level plane.

$$\mathsf{L} = K_B + K_{w_1} + K_{w_2} \tag{6}$$

Each of these terms will consist on a term due to the linear movement and one due to the rotation:

$$K_B = \tfrac{1}{2}MV^2 + \tfrac{1}{2}I_B\dot{j}^2 \tag{7}$$

$$K_{w_i} = \tfrac{1}{2}mv_i^2 + \tfrac{1}{2}I_w\dot{q_i}^2, i = 1,2 \tag{8}$$

In equation 7, I_B represents the moment of inertia of the robot whereas in equation 8, I_w represents the moment of inertia of each wheel. As both body and wheels are considered solid discs:

$$I_B = \tfrac{1}{2}Mb^2 \text{ and } I_w = \tfrac{1}{2}mr^2 \tag{9}$$

Replacing these values for the inertia, and using equations 1 and 2 on equation 6, the Lagrangian expression is obtained:

$$\mathsf{L} = \left[\frac{3r^2}{16}(M + 4m)\right]\left(\dot{q_1}^2 + \dot{q}_2^2\right) + \left[\frac{Mr^2}{8}\right]\dot{q_1}\dot{q}_2 \tag{10}$$

The relation between the angular acceleration of each wheel and the torques applied is obtained from equation 10, using the following relation:

$$t_i = \frac{d}{dt}\left(\frac{\partial}{\partial\dot{q_i}}\mathsf{L}\right) - \frac{\partial}{\partial q_i}\mathsf{L} \tag{11}$$

In equation 12 $\ddot{q_i}$ represents the acceleration of wheel i, and τ_i the applied torque.

$$\begin{bmatrix}\ddot{q_1}\\ \ddot{q}_2\end{bmatrix} = \begin{bmatrix}\frac{3r^2}{8}(M+4m) & \frac{Mr^2}{8}\\ \frac{Mr^2}{8} & \frac{3r^2}{8}(M+4m)\end{bmatrix}^{-1}\begin{bmatrix}t_1\\ t_2\end{bmatrix} \tag{12}$$

C. DC Motor Model

To complete the model of the robot, the DC motors attached to each wheel must be also added. These motors will apply the needed torque to achieve the desired acceleration. The simplified equations that relate the voltage applied to each motor, V_i, with the applied torque are as follows:

$$L\frac{di_i}{dt} + Ri_i = V_i - K_m G\dot{q_i}, i = 1,2 \tag{13}$$

$$t_i = GK_a i_i, i = 1,2 \tag{14}$$

L represents the electric inductance of the motor, R the electric resistance, K_m is the motor constant and K_a is the armature constant. G represents the mechanical gear reduction that connects each wheel to its motor.

III. CONTROL STRATEGY

A. Control Problem

The objective of the control strategy is to generate the necessary voltages on each DC motor, to move the robot from a starting posture $\mathbf{X_0}=[x_0\ y_0\ \varphi_0]^T$, to a final goal (x_f, y_f), without constrains on the final angle φ_f.

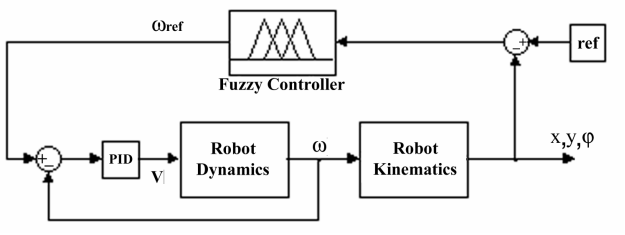

Fig. 2. Cascade control scheme

The main difficulty of this control strategy is that the kinematic equations of the robot are non-linear and there is no unique operating point, which could help the design by using a linearization [12]. Another problem is that the posture equations (3 and 4) are coupled, as they both depend on ω_1 and ω_2, or τ_1 and τ_2 which are the actual manipulated variables. On the other hand, the dynamic and DC motor equations are linear, and although they are also coupled, the use of a classical controller, such as a PID controller, to control the velocity of each wheel could return good results. However, the use of a PID controller for solving the whole control problem is very inefficient, especially because there are no general methods to tune the gain parameters in the case of non-linear plants such as this one.

A strategy that has shown to be very efficient to control non-linear plants is fuzzy logic [7],[8]. The problem with this method is that the amount of input variables needed in this case is high, due to the fact that the manipulated variables are acceleration related, whereas the control is done over position related variables. This means that the controller needs not only the distance and relative angle to the final destination, but also the approaching velocity and angular speed of the robot. As a way to reduce the amount of input variables on the fuzzy controller and simplify the computational requirements, a cascade control scheme is used. First, a tuned PID controller is implemented to control the velocity of each wheel by modifying the voltage applied to the motors. On top of this controller, a fuzzy logic controller is used to generate the needed angular velocities so the robot moves to the desired reference. Figure 2 shows the proposed control scheme.

B. PID Controller Design

Although the dynamic equations of the robot are coupled, the implemented PID stage is based on two independent controllers, one for each wheel. As figure 2 shows, each PID controller senses the angular speed of the corresponding wheel and uses the detected error to increase or reduce the voltage applied to the motor. The reference for this loop is given by the fuzzy logic controller. As in real life robots have a limited voltage range to apply to the motors and the motors have a maximum input voltage, the PID output is limited to ±5 [V]. This also ensures that the torques applied by the motors to the robot wheels stay in a limited range.

The gains for each PID controller are tuned, having as a goal a quick settling time and no more than 1% overshoot [9].

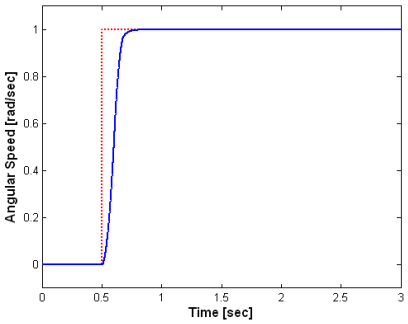

Fig. 3. Angular speed control

As the simulation on figure 3 shows, the PID controller is able to meet the requirements using the following gains: K_p=450, K_i=1, and K_d=20.

The PID control stage was tested in several conditions, showing in all the tests that the design constrains were respected, even in the worst scenario: when one wheel is set to move in one direction while the other is set to another. The simulations also showed that changes in one of the references made no significant disturbances on the velocity of the other wheel.

C. Fuzzy Controller

The objective of this controller is to create the necessary references for the angular velocities of each wheel, in order to move the robot from its starting posture to the final destination.

Considering the problem from a qualitative point of view, it is possible to create a set of rules that takes into account the distance to the objective (named D) and the relative angle, between the direction of the robot and the final destination (named Δφ), to determine the velocity of each wheel, which will be the manipulated variable. The rules will be of the form:

$$If\ D\ is\ LD\ and\ D\,j\ is\ LD\,j\ then\ w_1\ is\ Lw_1\ and\ w_2\ is\ Lw_2 \qquad (15)$$

In equation 15, LD is one of the distance related membership functions, LΔφ is related to the relative angle, and $L\omega_1$ and $L\omega_2$ are the membership functions for the speed of each wheel. Figures 4, 5, and 6 show the membership functions used for the fuzzy controller.

The distance between the center of mass of the robot and the objective is used as a way of controlling the arrival speed. This is done later, in the creation of the rule base, by relating membership functions (MFs) associated with smaller distances to MFs associated with slower speeds for each wheel. Three MFs were created for the distance: Close (C), Far (F), and Very

Far (VF), as shown in figure 4.

The other input of the fuzzy controller is the relative angle $\Delta\varphi$, which was divided into five MFs, covering from $-\pi$ to π. The used MFs relate the position of the objective with respect to the angle of the robot: Back Right (BR), Front Right (FR), Center (C), Front Left (FL), and Back Left (BL).

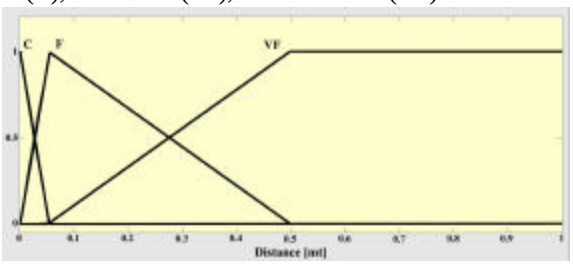

Fig. 4. Distance membership functions.

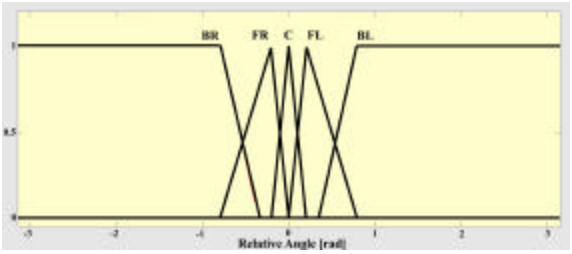

Fig. 5. Relative angle membership functions.

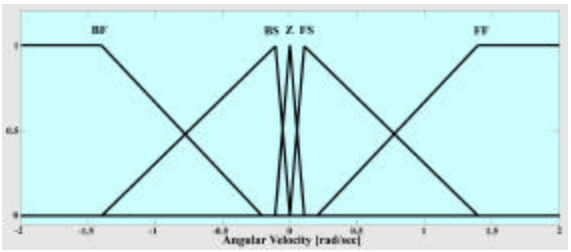

Fig. 6. Membership functions for the angular velocity of the wheels.

Figure 5 shows the different MFs created for the relative angle. This variable is used to control the rotation speed of the robot, making it turn quickly when the relative angle is high, whereas it moves in a straight line when the relative angle is close to zero. The width of the center membership function, C, is responsible of deciding when the robot is going to start to move forward. If C is too narrow, the robot starts moving towards its goal only when it is in front, loosing time and energy in a rotation without advancing. On the other hand, if C is too wide, the robot starts moving before it is facing the objective, doing long and curved trajectories that are not efficient.

Finally, five MFs are implemented for the speed of each wheel: Back Fast (BF), Back Slow (BS), Zero (Z), Front Slow (FS), and Front Fast (FF). These are presented on figure 6.

The rule base for the fuzzy logic controller is shown on tables 1 and 2, one for each wheel. These rules associate the state of the robot with respect to the objective (distance and relative angle), with the needed velocity for each wheel. The rule base is designed to make the robot turn quickly when it is far away from the goal, and then continue on a straight line. In this way the trajectory followed by the robot is minimal and no energy is wasted in log turns. The rule base must also make the robot move fast when it is far away, and slow down at the time of arrival.

TABLE I
RULE BASE FOR ω_1

D						
	C	BF	BS	Z	FS	FF
	F	BF	Z	FS	FS	FF
	VF	BF	FS	FF	FF	FF
		BR	**FR**	**C**	**FL**	**BL**
				D j		

TABLE II
RULE BASE FOR ω_2

D						
	C	FF	FS	Z	BS	BF
	F	FF	FS	FS	Z	BF
	VF	FF	FF	FF	FS	BF
		BR	**FR**	**C**	**FL**	**BL**
				D j		

The rules of the fuzzy controller are inspired on heuristic knowledge of the behavior the robot must have in order to accomplish the task. The behavior is similar to what humans do in order to go from one point to another. For example, if the objective is at the back and to the left, then the right wheel must go forward, while the left one must go backwards, making the robot turn till the objective is almost in front. Then the robot must start moving forward towards the goal, correcting slightly the direction of movement if the relative angle increases while moving. Depending on how far is the objective, the velocity of the wheels will increase to move faster (or turn quicker), and when the goal is near the speed is reduced so the robot can stop on arrival. In a more general way, the robot will turn until it faces the goal and then move on an almost straight line. The accuracy to face the objective will be given by how narrow is the C membership function of the relative angle variable.

D. System Simulation

To test the performance of the controller, the whole system was simulated using Simulink. The goal of the robot was to move from and initial position (0,0) and a variable initial angle, to a final position (-2,1) on the XY plane. Figure 7 shows the simulation results.

Four different initial angles were used: $-\pi$, $-\pi/4$, $\pi/4$, and $3\pi/4$, to consider the behavior of controller in different cases. As figure 7 shows, the robot moves using small turns by rotating first from its initial position and then moving in an almost straight line towards the destination point.

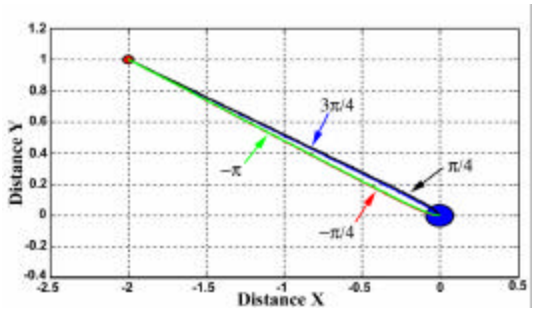

Fig. 7. Robot trajectory for different initial angles: -π, -π/4, π/4, and 3π/4.

IV. Genetic Optimization of the Fuzzy Controller

A. Method Description

The simulations show that the performance of the controller is very sensitive to the position of each MF on the fuzzy controller, indicating that it could be optimized to improve the performance. An interesting way to do this is by using evolutionary computation algorithms, to determine a better position for each MF based on a performance parameter also known as "fitness" [3],[4].

Genetic optimization algorithms work in a similar way to what evolution theories describe. The algorithm starts with an initial population of possible solutions. Each one is tested and a fitness value is assigned to them depending on the performance of the solution, which helps to determine the better solutions within the population. Using one of the several methods [4], a group of solutions (generally the ones with a higher fitness) are selected to be combined, with some probability, with the other solutions of the population, hoping that the mixture between them could create a better solution. The cycle is repeated several times and it is stopped after a certain number of generations. There is a large number of ways to implement a genetic algorithm [10], depending on the goals of the optimization. Most of them use "elitism", which means that the best solutions are always copied directly into the next generation, ensuring that the "genes" of these solutions remain in the population. The use of elitism gives an advantage over other implementations, because the process can be stopped at any time and it will always have a better or at least equal solution to the best solution in the initial set. On the other hand, when using genetic algorithms there is no demonstration that the achieved solution is the global optimum.

Another evolutionary element added is the use of mutation within the genetic algorithm. This means that with a certain probability the genes from some individuals change randomly, adding new elements to the population and eliminating or at least diminishing the possibilities that the whole population is kept within a local optimum.

Several researchers have applied genetic optimization on fuzzy logic systems, achieving a better performance on their systems compared to benchmark solutions. This optimization approaches include parameter tuning on the MFs and rule optimizations as in [11].

In this work, a recently applied method for selecting the better solutions of the population is used [2]. This algorithm is based on the evolution scheme used by bees, in which only one single member of the colony, the queen, is able to combine with the rest of the population to create a new generation. This makes easier choosing the parent solutions and helps to keep the best solutions within the population.

The optimization will only modify the MFs of the distance and relative angle variables, leaving the MFs of the speed of each wheel without change. As the MFs are triangular, they can be expressed as a three element vector containing the start, peak, and stop coordinates of each of them. Each controller contains 3 distance MFs and 5 relative angle ones, making it possible to describe the whole controller by a 8x3 matrix, called C_i. Each matrix describes one element in the population. On every generation, all solutions are tested and the one with the highest fitness is combined with all the other solutions using a certain probability. The combination is done by averaging both individuals:

$$C_{new_i} = \frac{C_{best} + C_i}{2} \tag{16}$$

Elitism and mutation is used within the optimization to ensure that the best solution is kept and to minimize the chance that the population converges to a local optimum. The two conditions of the control problem are that the robot achieves the goal as fast as possible, and that the end velocity is low enough so the robot is able to stop. As a way of including these two restrictions, the fitness function used is a linear combination of both, as described in equation 17, where T is the time used to reach the objective and ω_i is the final speed of each wheel:

$$F = T + \alpha r\left(w_1 + w_2\right) \tag{17}$$

The optimization seeks to get the lowest possible fitness, which means that the robot must reach the goal fast, and with low final speed. The α factor is used to give a relative weight between the time and speed constrains, having units of [sec^2/mt] to leave the fitness in [sec]. A higher value of α will imply that the optimum will have a slower end speed than the one with a low α value.

B. Optimization

The optimization is done using an initial population of 20 different fuzzy controllers. Each of these is created using the original robot fuzzy control as a base, but with all its genes modified randomly. The combination probability is set to 95% and a mutation probability to 5%, with a simulation time of 50 generations. Each individual is tested using $[0\ 0\ 0]^T$ as the

initial posture, and setting the goal at (-1,1). The α factor in the fitness function is set to 1600 [sec^2/m], to make the time taken to reach the goal and the final speed comparables. Using these parameters the fitness value for the original control system is 67,81 [sec].

The optimization cycle is repeated 3 times to check if the achieved solutions have things in common. In all three cases the fitness of the best solution is in average 26 [sec], needing 25,88 [sec] to achieve the objective and arriving at a speed of 7,47x10^{-5} [m/sec]. The MFs obtained after the optimization are shown on figures 8 and 9. In all three cases the best solutions share an element in common: the Far (F) membership function is moved away from the operating range, which was from 0 to 1,4 [m]. This means that this MF is not needed in the system and only introduces delays, making the controller less efficient.

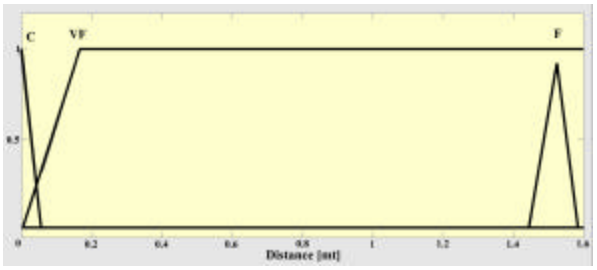

Fig. 8. Optimized set of MFs for the distance variable.

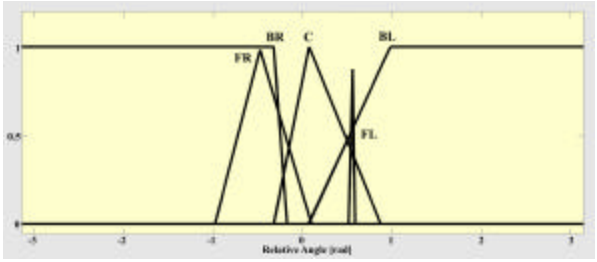

Fig. 9. Optimized set of MFs for the relative angle variable.

For the relative angle MFs, a similar effect occurred. All the optimal solutions eliminated the Front Left (FL) membership function from the operating range, either by making it so narrow that it never becomes activated (as shown on figure 9) or by moving it away from the operating rage in the simulation, which was from 0 to 3π/4. This also implies that this MF is not needed in the control system. As the MFs associated to the right side of the robot are never active, no important changes are observed on them, whereas the Center (C) MF is deformed sideways in all 3 solutions.

To check if the optimum position for the relative angle MFs is symmetrical, the optimization is done again with the goal set on (-1,-1). The optimization shows that the optimal solution for the distance variable is the same, whereas the solution for the relative angle variable is almost symmetrical to the ones obtained before.

As all solutions indicate that some MFs are not needed, these are eliminated from the fuzzy controller, and the optimization is done again to check if some improvement is possible. With the goal set on (-1,1) the optimization algorithm is able to reduce the fitness function of the optimal controller to 25,12 [sec]. The MFs obtained after the second optimization are shown on figures 10 and 11. Notice the non symmetrical shape of C on the relative angle MFs.

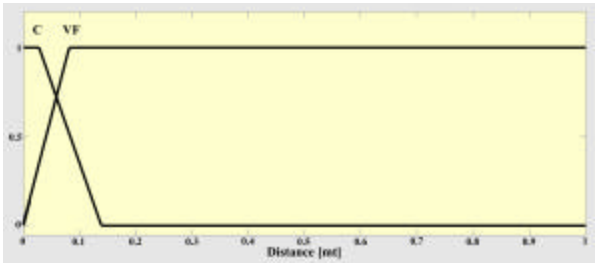

Fig. 10. Optimized set of MFs for the distance variable, after eliminating the Far (F) MF from the original set.

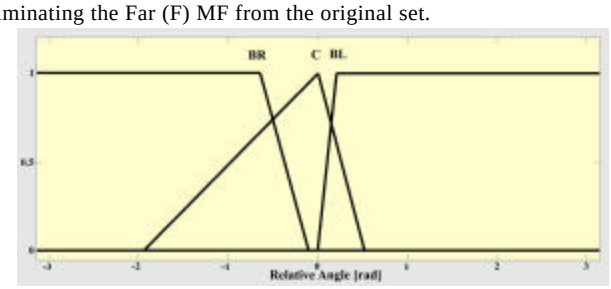

Fig. 11. Optimized set of MFs for the relative angle variable, after eliminating both Front Right (FR) and Front Left (FL) MFs from the original set.

V. Design of an Adaptive Fuzzy Controller

The different solutions show that the optimal positions for the MFs depend on the position of the goal. The optimum solution for going from the origin to the coordinate (-1,1) is not as good if the goal is set on (-1,-1). To create a general adaptive control system, the optimal solutions for both cases are combined depending on the final destination, creating a controller that is able to go efficiently from one point to another, with an overall performance better than the optimized controllers by themselves, outperforming the original fuzzy controller, and without the need of optimization cycles for every new destination goal.

The adaptive controller is created by a linear combination of the two solutions obtained in the optimization stage. This is done by combining the matrices that describe the controller as equation 18 shows:

$$C(\lambda) = \lambda C_1 + (1-\lambda) C_2 \qquad (18)$$

Where C_1 is the control matrix that describes the fuzzy controller optimized to go to the point (1,1), whereas C_2 describes the controller optimized to go to (-1,-1). The value of λ, the adaptation parameter, is selected depending on the angle of the goal with respect to the angle of the robot.

This type of adaptive controller can be used for trajectories based on checkpoints, where the control system can

recalculate the fuzzy MFs parameters every time a checkpoint is reached, adapting the controller to have an improved performance depending on the position of the next checkpoint. In this way, the control strategy is optimized based on the actions the robot must take on the future.

To compare the adaptive controller with the previous control systems, the robot is set to move from the origin to (1,1) and then to (2,0). Three controllers are used in the simulation: the original fuzzy controller, one of the optimized controllers from section IV, and the adaptive controller. For all three cases the different trajectories are compared, as well as the angular speed of the wheels over time.

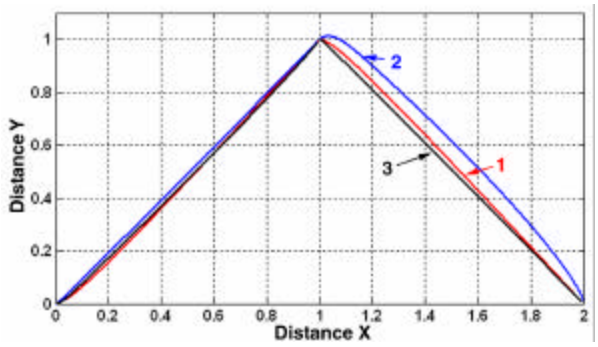

Fig. 13. Trajectory comparison between the original fuzzy controller (1), one of the optimized controllers (2) and the general controller (3).

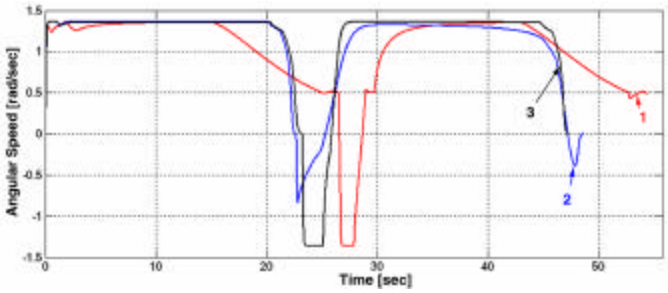

Fig. 14. Angular speed for the original fuzzy controller (1), one of the optimized controllers (2) and the general controller (3).

As figure 13 shows, the general controller makes the robot move almost in straight lines towards the checkpoints, using less time and wasting less energy than the other controllers. Figure 14 shows that the general controller also allows the robot to move faster, arriving in less time and with a lower end speed than the other controllers. The fitness value for the different controllers in this test is: 132,14 [sec] for the original controller, 48,89 [sec] for the optimized one, and 47,52 [sec] for the adaptive controller.

VI. Conclusion

Through this work it is showed that the new described control scheme results in an excellent control system for a 2 wheel mobile robot. It is also demonstrated, that the "Queen Bee" based genetic optimization algorithm is a very good tool to optimize the performance of fuzzy logic controllers, and that by modifying the parameters that create each membership function the efficiency can be improved.

Finally, this work presents an adaptive fuzzy controller that can modify its membership functions based on the goals ahead, without the need of an optimization cycle every time the goal is changed.

References

[1] S. Coradeschi, S. Tadokoro, A. Birk, "RoboCup 2001: Robot Soccer World Cup V", Springer Verlag, 2002.

[2] S. H. Jung, "Queen-Bee Evolution for Genetic Algorithms", IEE Electronic Letters, 20 March 2003, pp. 575-76.

[3] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, "Genetic Programming : An Introduction", Morgan Kaufmann, 1997.

[4] M. Michell, "An Introduction to Genetic Algorithms", MIT Press, 1998.

[5] P. Goel, G. Dedeoglu, S. Roumeliotis, G. Sukhatme, "Fault Detection and Identification in a Mobile Robot using Multiple Model Estimation and Neural Network", Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, USA, 2000.

[6] J. Craig, "Introduction to Robotics: Mechanics and Control", 2nd ed., Addison-Wesley Pub. Co., 1989.

[7] R. Palm, D. Driankov and H. Hellendoor, "Model Based Fuzzy Control", Springer Verlag, 1997.

[8] M. Reinfrank, H. Hellendoorn, D. Driankov, "An Introduction to Fuzzy Control", 2nd ed., 1996

[9] G. Goodwin, S. Graebe, M. Salgado, "Control System Design", Prentice Hall, 2000.

[10] X. Yao, "Evolving artificial neural networks", Proceedings of the IEEE, September, 1999, vol. 87, n°. 9, pp. 1423-1447.

[11] M. Maniadakis, H. Surmann, "A Genetic Algorithm for Structural and Parametric Tuning of Fuzzy Systems", European Symposium on Intelligent Techniques, 1999.

[12] L. Kleeman, "Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning", IEE International Conference on Robotics and Automation, Nice, France, 1992, vol. 3, pp. 2582-2587.

[13] The Mathworks Inc., "Fuzzy Logic Toolbox. User's Guide", 1998.