Appeared in:
Seventh International Fuzzy Systems Associations (IFSA'97) - World Congress,
pages 496-500, Prague, Czech, June 25-29, 1997.

Fuzzy Logic Control of a Situated Agent

G. Hailu†, J. Bruske, G. Sommer

Christian Albrechts University
Department of Cognitive Systems
Preußerstraße 1-9, D-24105 Kiel, Germany

Abstract

In this paper we present a sensor preprocessor and a fuzzy logic based navigation control for a situated agent. We address the problem of sensor inaccuracy in mobile robots by partitioning the total perceptual space into overlapping regions and designing a Kalman filter for each region. The filter estimates the true depth of each region by propagating an assumed conditional probability density function from sometime, arbitrary deep in the past, up to the present time. In applying fuzzy control we depart from Saffiotti et al. [13] by discarding the context blender. Here we build a clean *monolithic* rule bank for an *atomic* task. Our controller has ≈ 1000 fuzzy rules and two crisp outputs. The outputs of the controller are motor velocity and jograte. The implemented controller has successfully steered both the TRC mobile robot that has a belt of ten ultrasonic range sensors around its waist and its simulator. The observed collision free, smooth and uninterrupted trajectories demonstrate the robustness of the fuzzy controller and the noise tolerance nature of the preprocessor, even in the presence of large sensor noise.

Keywords: Situated agent, Kalman filter, Fuzzy control.

1 Introduction

Since the first notion of fuzzy sets [18], fuzzy control has emerged as one of the most active research areas. Over the past several years, application of fuzzy control in backing up an autonomous vehicle [5, 7], real time target tracking [9], avoiding collision of ships [3], and navigation of mobile robots [14, 16] have shown the potential and fruitfulness of fuzzy control. This is because fuzzy logic control is able to cope with lack of models and can handle vaguely defined processes which in many case can only be controlled by a skillful human operator.

In the context of mobile robots, fuzzy control is used for reactive navigation. Reactive navigation can be defined as a mapping between sensory data and motor commands [12]. Unlike classical reactive methods, such as potential field [6], which searches for a function that best fits this mapping, fuzzy system use a small number of structured *linguistic* input-output samples. We shall refer to these input-output samples as Fuzzy Associative Memory (FAM). Each FAM rule defines a *patch* in the input-output space and the fuzzy control approximates the unknown function by covering its graph with FAM-rule patches [9]. The FAM rules can thus be regarded as forming the *skeleton* of a fuzzy control architecture.

Here we are interested in constructing a fuzzy controller to steer a TRC mobile robot in an unknown indoor environment. Our controller uses only the sensory information, coming from the environment, and the vehicle velocity to provide a control command at each control cycle. However, on the one hand, the sensors are still far from perfect to perceive the environment and on the other, their raw data are too huge and redundant to explain the condition to be dealt with by the agent. Therefore a special sensor preprocessor that provides only few and relevant data for decision making, has to be designed.

The work described in this paper bears some similarity to [14, 15, 16]. However, they all work on a simulated environment and robot that usually hides the real dynamics and sensor uncertainty of

†Corresponding author.

the actual system. To control Toto, Matarić [10] has relied solely on the high accuracy of the sensors at small incident angle. Even though sonar sensors have such a characteristic, it is very difficult to maintain always the sensors mounted on the robot at a small incident angle from the surface normal. In addition, her controller is crisp and she has tackled the whole navigation task from a different point of view - behavioral decomposition. Borenstein et al. [1] have eliminated the noise due to sensor crosstalk using a special sensor firing scheme called (EERUF). However, their method is not easily transferable to a system, such as ours, where both firing sequence and firing intervals are hardware fixed. Perhaps our work is close to [13] who implement a hierarchical fuzzy control architecture, similar to [4], on the Flakey. They decomposed the overall control into a number of modules, which they call fuzzy behaviors, and have used a context dependent blending scheme to combine behaviors outputs. But we argue that in fuzzy logic, since we can incorporate all available knowledge to write clean rules, there is little benefit, except the extra work of designing the blender, in breaking down the atomic task `avoid obstacle` into a number of small modules[a] like `corridor follow`, `keep off`, `door pass`, etc. In addition, they didn't discuss the well-known problem of sensor noise and uncertainty in mobile robot research!

In section II, we briefly describe the agent we are working with. Section III provides details of the partitioning of the perceptual space and the type of sensor preprocessing employed. Section IV and V discuss the fuzzy linguistic sets, the fuzzifier and the controller. Here, no attempt is made to discuss the basics of fuzzy theory, it is explained elsewhere [9, 12]. The last two sections present our experimental results and the on going work.

2 The Physical Agent

The agent which we are dealing with (Fig. 1) is a two wheeled 75 cm L × 70 cm W × 28 cm H autonomous vehicle with a belt of ten ultrasonic sensors around its front waist. The vehicle can turn in place by an arbitrary angle `rotate` (rad), can move a straight path at an arbitrary velocity `go` (mm/sec) and can turn while moving `jog` (rad/sec). On board, two processors are mounted. The Motorola 68HC11 micro-controller fires and reads individual sonars, sends motor commands to the servo controller and handles all time related synchronization details at low level. The micro-computer, which comes in between the host and the micro-controller, runs various high-level motor and sensor utilities and routes sensor values from the agent to the host and motor commands from the host down to the agent. The host, a SUN workstation, implements the main fuzzy controller and the sensor preprocessing stages. It communicates with the micro-computer through an Ethernet radio link (Fig. 2).

Figure 1: The TRC mobile robot.

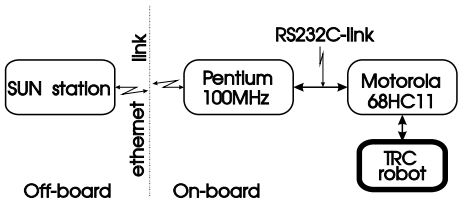


Figure 2: The distributed computing layout.

3 Sensor Preprocessing

3.1 State space partitioning

Because of sensor uncertainty and combinatorial explosion of the fuzzy rules, it is very difficult to formulate the fuzzy rules directly from the huge raw sensory data. To reduce the number of fuzzy rules, Song et al. [14] have built two fuzzy controllers for their simulated robot, each handling a portion of the total sensor space. This requires a separate switching mechanism between the two controllers and makes control quite complex. Reignier [12] suggests to keep from each region only the sensors with minimum depth and to discard all the others. This method does not consider the agent's inaccuracy in sensing world state and is easily defeated in the actual laboratory.

Rather than designing two controllers or discarding all readings except the minimum, we have partitioned the ten ultrasonic sensors into five regions, corresponding to the physical geometry of the agent and the task at hand. These regions are: `I-right corner`, `II-right`, `III-front`, and `IV-left`, and `V-left corner`. Moreover, we adopt sensor *overlapping* across neighboring regions to increase the

[a] But this is not to say that task decomposition, in general, is not necessary. In fact, when a task gets complex, task decomposition is the only preferred method.

accuracy of depth measurement as well as to account for the beam angle of the sonars. Following partitioning, the sensor values of each region are passed through a median filter, which gives as output a measurement depth of a region. We have employed a Kalman filter to estimate the true depth from the present and past measurement values. After estimating the true depth of each region, we proceed to the fuzzification process, which is common to all fuzzy based systems. Before describing the fuzzification process, we will discuss the Kalman filter formulation.

3.2 Kalman Filter Formulation

Unlike [2] in which we use heuristic to estimate the true depth of each region from multiple readings taken at each perceptual cycle, here a cascade of two filters (Fig. 3) and a sliding window of size three, to hold the present and the past two measurement profiles [17], are used. The first filter is a non linear median filter that estimates the current *measured depth* of a region j using,

$$\mathcal{Z}_{j,t} = median\left(\mathcal{S}^j_{1,t}, \mathcal{S}^j_{2,t}, \ldots, \mathcal{Z}^j_{N_j,t}\right) \qquad (1)$$

where $\mathcal{S}^j_{i,t}; i = 1 \ldots N_j$ is the reading at time t of sensor i located in region j, N_j the number of sensors in region j, and $\mathcal{Z}_{j,t}$ is the measured depth of region j at time t. The measured depth $\mathcal{Z}_{j,t}$, however, is still noisy and unreliable for reactive control[b] without further processing! Hence, we propose a Kalman filter to process the measured depths further.

The Kalman filter operates on the present and past measurement profiles $(\mathcal{Z}_{j,t}, \ldots, \mathcal{Z}_{j,t-n})$, stacked in the sliding window, to estimate the current true depth, $\mathcal{D}_{j,t}$, of a region j. To avoid the influence of very past measurements on the present estimate, only a limited window size ($n = 2$) is taken. Since a Baysian view point is adopted, we need to select a model for the conditional probability density function (CBDF) of the true depth given the measured depth $\mathcal{P}(\mathcal{D}_j/\mathcal{Z}_j)$, that best fits the data generated by the real world. In this paper a Gauss[c] CPDF is chosen. The main motivations for making this assumption is that the Kalman filter so designed is optimal with respect to virtually any criterion that makes sense [11]. As our view point is Baysian, we require the filter to propagate the assumed CPDF from some arbitrary time in the past up to the present time. Once the CPDF is propagated the optimal estimate is computed using the *maximum likelihood* criterion.

[b] When these values are used to generate control commands the robot is seen moving arbitrary.

[c] There is no mathematical or experimental prove that guarantees a Gaussian noise distribution in ultrasonic sensors.

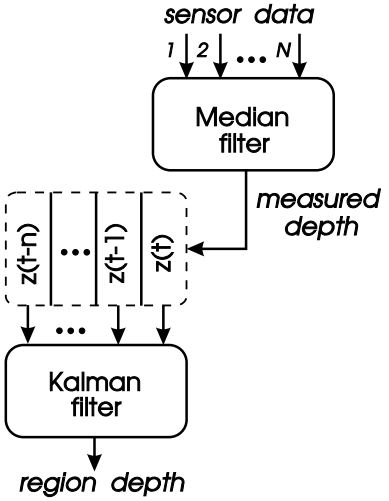


Figure 3: The proposed sensor preprocessing scheme.

The Kalman filter algorithm is tailored to suit the agent at hand. To proceed with the algorithm, at each perceptual time the filters in each region are initialized by estimating the parameters of the Gauss CPDF, i.e., mean μ_j and variance σ^2_j. We estimate the mean by equating it with the measured value at time $t - n$ i.e.,

$$\mu_{j,0} = \mathcal{Z}_{j,t-n}; \quad j = 1 \ldots 5 \qquad (2)$$

and the variance $\sigma^2_{j,0}$ by equating it with the measurement variance of the sonars, ς^2.

To compute the measurement variance, we have picked a sensor at random[d] and made a separate experiment on it. The selected sensor was placed in different environments and at different orientations and depths that can be faced by the robot when it is in operation (such as corners, corridors, doors edges, walls, free ways, ...). For all environments and depths, the sensor was fired and the true (d) and measured (r) depths were recorded. After recording 1000 (d, r) pairs, the measurement variance ς^2 was computed by,

$$\varsigma^2 = \frac{1}{N}||\boldsymbol{d} - \boldsymbol{r}||^2 \quad ; \quad \varsigma = 137\ mm \qquad (3)$$

where $N = 1000$. But this is a low value which does not represent the actual measurement variance of the sensors when they are fired one after the other and when the robot moves. To account for these dynamics, the value obtained above is multiplied by a factor of 2.5 to result in $\varsigma \approx 350\ mm$. At the beginning of the updating algorithm the statistical variance is set to this measurement variance, i.e,

$$\sigma^2_{j,0} = \varsigma^2 \quad j = 1 \ldots 5 \qquad (4)$$

[d] All the sensors are of the same Polaroid type.

With Eqn. (2) and (4) the CPDF of each region is initialized. The next step is to propagate this CPDF forward up to the present time. Inherently our system is dynamic, i.e., agent position and hence sensor values change with time. Therefore, the dynamic Kalman filter is more appropriate to our scenario. Unfortunately, this filter requires a model for the rate of change of the sonar return. For a situated agent, this change depends among other things on: the speed and rotation of the robot, the direction of motion, the environment and its acoustic property, the dynamic property of each sensor, the position of the sensors on the robot and, the frequency of sensor crosstalk. Looking at the parameters involved, it is extremely difficult to come up with a clean mathematical model of the form Eqn. (5) and (6) which the dynamic filter requires.

$$\dot{\mathcal{X}}(t) = \mathcal{A}(t)\mathcal{X}(t) + \mathcal{B}(t)\mathcal{U}(t) + \mathcal{V}(t) \quad (5)$$

$$\mathcal{Z}(t) = \mathcal{C}(t)\mathcal{X}(t) + \mathcal{W}(t) \quad (6)$$

Here matrices $\mathcal{A}(t)$, $\mathcal{B}(t)$ and $\mathcal{C}(t)$ are system time varying coefficients incorporating all the above mentioned parameters, vectors $\mathcal{X}(t)$ and $\mathcal{Z}(t)$ are estimated and measured depths respectively, and $\mathcal{V}(t)$ and $\mathcal{W}(t)$ are system and measurement noises respectively.

Because of lack of the above system coefficients, a linear recursive Kalman filter is employed, and the CPDF is updated only at discrete time, i.e., when measurement value is available. At each update step, $i = 1, \ldots, n$, and for any perceptual region, $j = 1 \ldots 5$, the updating algorithm is given by:

- compute the Kalman gain:

$$\mathcal{K}_{j,i} = \frac{\sigma^2_{j,i-1}}{\sigma^2_{j,i-1} + \varsigma^2} \quad (7)$$

- update the mean:

$$\mu_{j,i} = \mu_{j,i-1} + \mathcal{K}_{j,i}(\mathcal{Z}_{j,t-n+i} - \mu_{j,i-1}) \quad (8)$$

- update the variance:

$$\sigma^2_{j,i} = (1 - \mathcal{K}_{j,i})\sigma^2_{j,i-1} \quad (9)$$

At the last update, we have the CPDF of the estimated depth given the present and the past two measured values, $\mathcal{P}(\mathcal{D}_{j,t}/\mathcal{Z}_{j,t-2}, \mathcal{Z}_{j,t-1}, \mathcal{Z}_{j,t})$. Once this CPDF is determined, the maximum liklihood criterion is used to extract the best estimate from the CPDF, i.e.,

$$\begin{aligned} \mathcal{D}_{j,t} &= max\ \mathcal{P}(\mathcal{D}_{j,t}/(\mathcal{Z}_{j,t-2}, \mathcal{Z}_{j,t-1}, \mathcal{Z}_{j,t})) \\ &= \mu_{j,2}; \quad j = 1 \ldots 5 \end{aligned} \quad (10)$$

We have implemented Fig. 3, Eqn. (2),(4) and Eqn. (7)-(10) for each region separately. These taken together define our *sensor preprocessing* stage.

4 Sensor Fuzzification

After the preprocessing stage is completed, the next step is to define fuzzy linguistic sets for each region. Akin to Matarić, we have defined three fuzzy linguistic zones, namely: `dangerous (d)`, `maneuver (m)`, and `safe (s)` zones. Instead of designing a separate membership functions for each region, we exploit domain knowledge to reduce the size of membership functions. We have taken identical membership functions (Fig. 5) for the three regions (II-IV), which are basically on the front side of the robot, and different membership functions (Fig. 6) for the two regions (I and V), which are on the corner side of the robot. With these we account for the difference in the degree of danger and degree of safety between front and corner regions [12]. Kosko's [9] rule of thumb `25 percent overlapping between adjacent fuzzy sets` performs poorly for our scenario. In our application the degree of overlap is different between adjacent fuzzy sets. Note also that the micro-controller is preprogrammed for a time out distance of 2 m, a reasonable choice for our robot size.

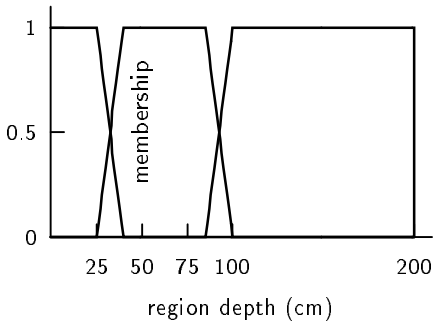


Figure 4: Overlapping trapezoidal membership functions for regions II-IV.

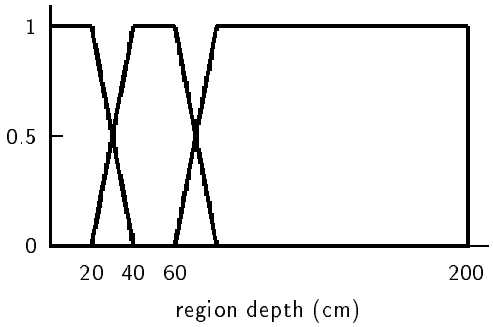


Figure 5: Overlapping trapezoidal membership functions for regions I and V.

The plots of the membership function actually define the fuzzifiers, which give as output a *fit vector* of dimension three. The $i-th$ fit vector measures or indicates the degree to which the input belongs to the $i-th$ fuzzy set. As a concrete example, for a sensor value of 350 mm in regions II-IV, the fuzzifier outputs a fit vector $\boldsymbol{f}=(0.33, 0.67, 0.0)^T$.

Apart from the depth information coming from external sensors, the controller also use the base velocity that is coming from the internal sensor. Four trapezoidal fuzzy sets: `reverse (r)`, `slow (s)`, `normal (n)`, and `fast (f)` with membership functions shown in Fig. 7 are defined to encode the crisp velocity into a fit vector of dimension four. Again as an example, a base velocity value of 80 mm/sec is encoded as $\boldsymbol{v}=(0, 0, 1, 0)^T$.

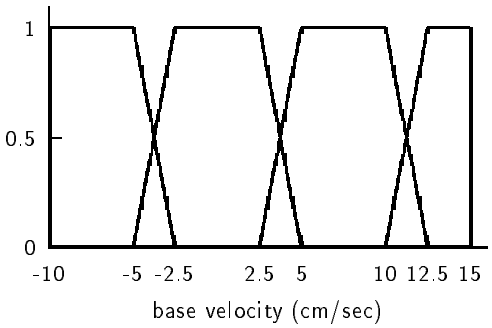


Figure 6: Velocity fuzzifier.

5 The Fuzzy Controller

Our fuzzy controller uses a bank of fuzzy associative rules of the form $(A_i, C_i); i = 1 \ldots k$ to capture the relationship between the observed variables and the controlled variables. A_i represents the antecedent `(if)` part of rule and C_i is the conclusion `(then)` part. Unlike the more generalized fuzzy rules, which have fuzzy sets in both antecedent and conclusion parts, our fuzzy rules have fuzzy sets only in the antecedent part and crisp values in the conclusion part - Sugeno type[e]. The antecedent part of the rules are boolean combinations of six propositions corresponding to the base velocity (taking any one of the four fuzzy values, Fig. 7) and the depth information of the five perceptual regions (each taking any one of the three fuzzy values, Fig. 5 and 6).

The controller (Fig. 8), has a total of 19 fuzzy inputs, $k = 3^5 \times 4 \approx 1000$ fuzzy rules and two crisp outputs. Note the size of the fuzzy rules even for low granularity level of description used for depth and velocity. A representative of our FAM rule is:

[e]In our subsequent work, we have changed this to a more generalized one.

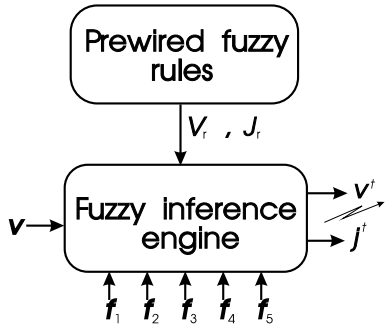


Figure 7: The fuzzy control structure.

```
rule 143:  if(
right corner=maneuver zone,
left corner=maneuver
zone,right=danger zone,
front=safe zone,
left=safe zone and
basevelocity=reverse) then
velocity=-50, jograte=10.
```

At each control cycle, the antecedent part of all the rules are satisfied simultaneously, but to a different degree. Let us say for rule r at time t its antecedent part is satisfied to a degree α_r^t, where $0 \leq \alpha_r^t \leq 1$. To impose competition among rules [9], Eqn. (11) must be satisfied:

$$\sum_{r=1}^{k} \alpha_r^t = 1; \quad \forall t \tag{11}$$

For each rule $r = 1 \ldots k$ there are two crisp suggestions: motor velocity V_r and jograte J_r. For Sugeno rules, these quantities are fixed during the writing of the rules. In addition, let the contribution of a rule r at time t to the velocity and jograte be designated by v_r^t and j_r^t respectively. Note that these quantities are variable at each time. The strength of the conclusion part of rule r at any control time t is based on the degree to which its antecedent part is satisfied, i.e.,

$$v_r^t = \alpha_r^t V_r; \quad \forall t \tag{12}$$

$$j_r^t = \alpha_r^t J_r; \quad \forall t \tag{13}$$

where $r = 1 \ldots k$. The final crisp control command $v^t$and j^t is then simply the sum of the contributions of each rules, i.e.,

$$v^t = \sum_{r=1}^{k} v_r^t; \quad \forall t \tag{14}$$

$$j^t = \sum_{r=1}^{k} j_r^t; \quad \forall t \tag{15}$$

In each time step the host computes Eqn. (12)-(15) and sends v^t and j^t to the on board microcomputer, who initiates appropriate motor routines for executions.

6 Experimental Results

To avoid the pitfalls of simulation, we decided to do most of our experiment on the actual agent and the results are available on video. However, for the sake of this report, we have included test results of the proposed sensor preprocessor and simulation outputs of our fuzzy controller.

To test the proposed preprocessor, the robot was placed at a distance of 2 m in front of a wall. After firing all the sonars the robot is set to move against the wall at a constant velocity. While it is moving, we keep on recording the readings of the sonars in the front regions until the robot approaches the wall. Later we have applied our preprocessing algorithm and Reignier's [12] suggestions on the data gathered. We have plotted both results against time in figures 9 and 10. As can be seen clearly, our preprocessing scheme, Fig. 10, far exceeds Reignier's suggestion, Fig. 9. Specifically, notice how the Kalman filter holds (sustains) the depth estimate at a relatively high value without much swing while the robot is far from the wall. Undeniably, there is some swing in our preprocessor, too. However, the fuzzy controller is not sensitive to such little noise, because such noise affects the membership function only slightly, and therefore changes the final control command in minor way.

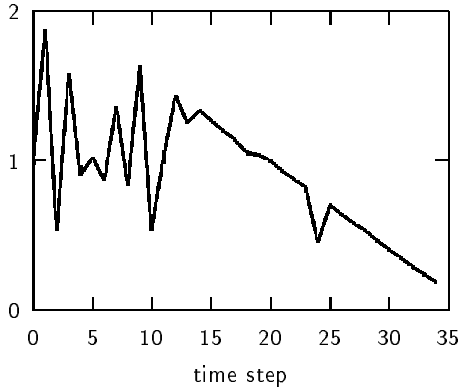


Figure 8: Performance test using the method suggested in [12].

Apart from the physical TRC robot, experiments have been carried out on a TRC simulator. The simulator we have built has only a simplified dynamics to account for the inertia of the robot, and it doesn't simulate the behavior of the sensors to test our preprocessor[f]. However, it takes into account the physical dimension of the robot by reducing its size proportionally and attempts to place each sensor in exactly the same locations as in the real robot.

[f] As we don't know how real world sensors behave, it is difficult to simulate them. Therefore, any *claimed* sensor preprocessor can only be tested on the actual system.

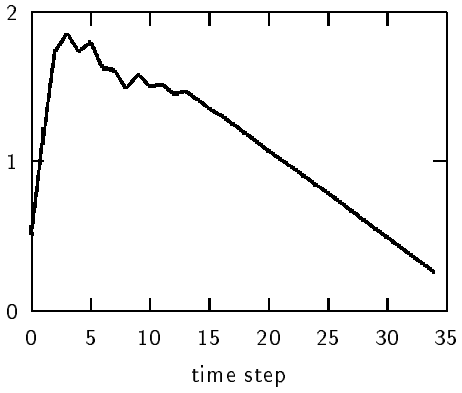


Figure 9: Performance test using the method suggested in this paper.

In the simulation, neither the rules nor the shape of the membership functions build for the actual robot have been tampered.

After creating a fairly dense block world, we let the robot free to navigate under the fuzzy controller. As can be seen from the ghost path of the robot in Fig. 11a, the simulated robot is able to navigate safely, smoothly, and slowing down rarely (as the robot tries to pass between the dining table and the walls) without halting.

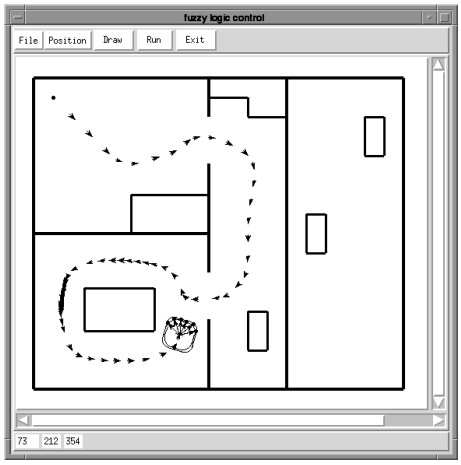
To ground our claim, a monolithic rule bank for an atomic task, we subjected the TRC robot to the same environment used by Saffiotti et al. page 14 in [13]. As shown in Fig. 11b our monolithic rule exhibits the same behavior without breaking down the task into two modules, `corridor follow` and `avoid obstacle` and later blending them together.

7 Summary and Future Work

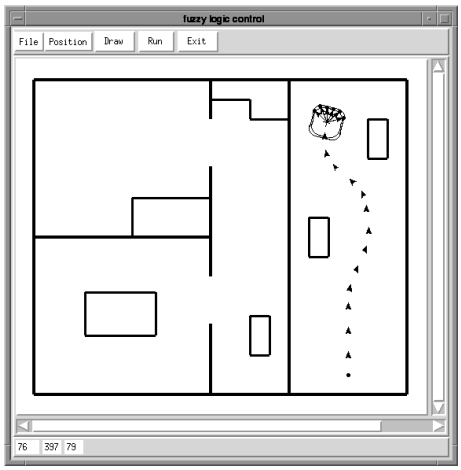
The paper has introduced a noise tolerant sensor preprocessor for an autonomous robots and has used it together with fuzzy logic control to navigate a TRC mobile robot in a noise prone environment. The preprocessor estimates the depth by propagating a Gaussian CPDF through the present and past measured values. Even though nothing is known about the noise properties of ultrasonic sensors, by assuming this particular distribution a satisfactory result is obtained! In the application of fuzzy control, we have departed from Saffiotti et al. by building a well written monolithic rule bank for an atomic task. In our system Saffiotti's et al. fuzzy behaviors appear as some part(s) of the rules and the *blender* exists implicitly within the rule bank.

The problem we faced in fuzzy logic approach is the trial and error nature of determining the right membership functions and tuning the fuzzy rules [g],

[g] But this is common for any fuzzy logic based system.



(a)



(b)

Figure 10: The simulated robot cruising through a block world environment.

so that the robot can exhibit satisfactory behavior for a broad range of environments. In addition, it is observed that the oscillatory behavior of the robot which is inherent in other methods, such as potential field [8], and the susceptibility of being trapped into local minimum also exist in fuzzy control. These undesirable behaviors can only be avoided when the robot has some learning capability.

Our intent is to design a goal directed navigating agent that adapts itself to a particular environment and task by self tuning its parameters. Reinforcement learning methods together with an RBF networks allow the agent to achieve this goal by letting the agent to do its own experiment and select the best action among the viable candidates. However, without some coarse initial knowledge this learning algorithm is extremely slow and costly.

Hence as a first step, we have chosen 250 rules from the working fuzzy controller (only a quarter of the total knowledge) and gave it to an RBF neural network which has 250 neurons, where by the center of each neuron is represented by the antecedent part of one of the selected rules and the output weight is represented by the corresponding conclusion part. After hooking the RBF neuro-controller[h] with the agent, we observed that with this little knowledge, the neuro controller has steered the robot collision free for a long time, passing even through narrow doors of our laboratory. This result is also available on video. The above experiment has shown us the possibility of rendering coarse initial knowledge to a connectionist system, in particular to an RBF neural network.

8 Acknowledgment

The support given to the first author by DAAD under grant code 413/ETH-4-BOA is greatly acknowledged.

References

[1] Johann Borenstein and Yoram Koren. Error eliminating rapid ultrasonic firing for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, 11(1):132–138, 1995.

[2] Getachew Hailu. Distributed fuzzy and neural network based navigation behaviours. Technical Lab. Report H696, CAU, Cognitive Systems Laboratory, 1996.

[3] Ichiro Hiraga, Takeshi Furuhashi, Yoshiki Uchikawa, and Shoichi Nakayama. An acquisition of operator's rules for collision avoidance using fuzzy neural networks. *IEEE Transactions on Fuzzy Systems*, 3(3):280–287, 1995.

[4] Robert A. Jacob, Michael I. Jordan, Stevan J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[5] Chi-Cheng Jou and Nan-Ching Wang. Training a fuzzy controller to back up an autonomous vehicle. In *IEEE International Conference on Neural Networks*, pages 923–928, San Francisco, 1993.

[6] Oussama Khatib. Real time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[h] We have steered the same robot with a multi-layer perceptron [2], but now abandoned it because of its inability to learn on-line incrementally.

[7] Seong-Gon Kong and Bart Kosko. Adaptive fuzzy systems for backing up a truck-and-trailer. *IEEE Transactions on Neural Networks*, 3(2):211–223, 1992.

[8] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404, Sacramento, 1991.

[9] Bart Kosko. *Neural Networks and Fuzzy Systems, A Dynamical Systems Approach to Machine Intelligence*. Prentice-Hall International Editions, 1992.

[10] Maja J. Matáric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.

[11] Peter S. Maybeck. The kalman filter: An introduction to concepts. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*. Springer-Verlag, 1994.

[12] Patrick Reignier. Fuzzy logic techniques for mobile robot obstacle avoidance. *Robotics and Autonomous Systems*, 12:143–153, 1994.

[13] Alessandro Saffiotti, Enrique H. Ruspini, and Kurt Konolige. Using fuzzy logic for mobile robot control. In D. Dubois, H. Prade, and H.J. Zimmermann, editors, *Handbook of Fuzzy Sets and Possibility Theory*. Kluwer Academic, 1997.

[14] K. T. Song and J. C. Tai. Fuzzy navigation of a mobile robot. In *Proceeding of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 621–627, Raleigh, 1992.

[15] M. Sugeno and M. Nishida. Fuzzy control of model car. *Fuzzy Sets and Systems*, 16:103–113, 1985.

[16] Tomoyoshi Takeuchi, Yutaka Nagai, and Nobuyoshi Enomoto. Fuzzy control of a mobile robot for obstacle avoidance. *Information Science*, 45:231–248, 1988.

[17] Jun Tani and Naohiro Fukumura. Learning goal-directed sensory-based navigation of a mobile robot. *Neural Networks*, 7(31):553–563, 1994.

[18] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.