II. LEG DETECTION

The leg detection algorithm has the role of providing the information needed by the fuzzy inference system to perform the control of the robot motion. The algorithm processes the data read by the laser sensor to detect the person's legs. It provides the position of the person's legs relative to the robot, the differential of which is the relative velocity of the person with respect to the robot. The obtained relative distance and relative velocity are then made available as input information to the fuzzy controller, which will determine how much speed change is needed by the robot in order to follow the person's legs.

The leg detection algorithm is based on the work of Milella et al. [8]. It works as follows. First, a 180^o laser scan is performed with a 1^o separation between two consecutive readings [9]. Actually, the laser has a maximum resolution of 0.5^o, but this was not used since it was not necessary in the practical setting of the problem and also to improve real-time performance. Then, the data is processed as indicated in the flowchart of Fig. 1 and explained in the following two paragraphs.

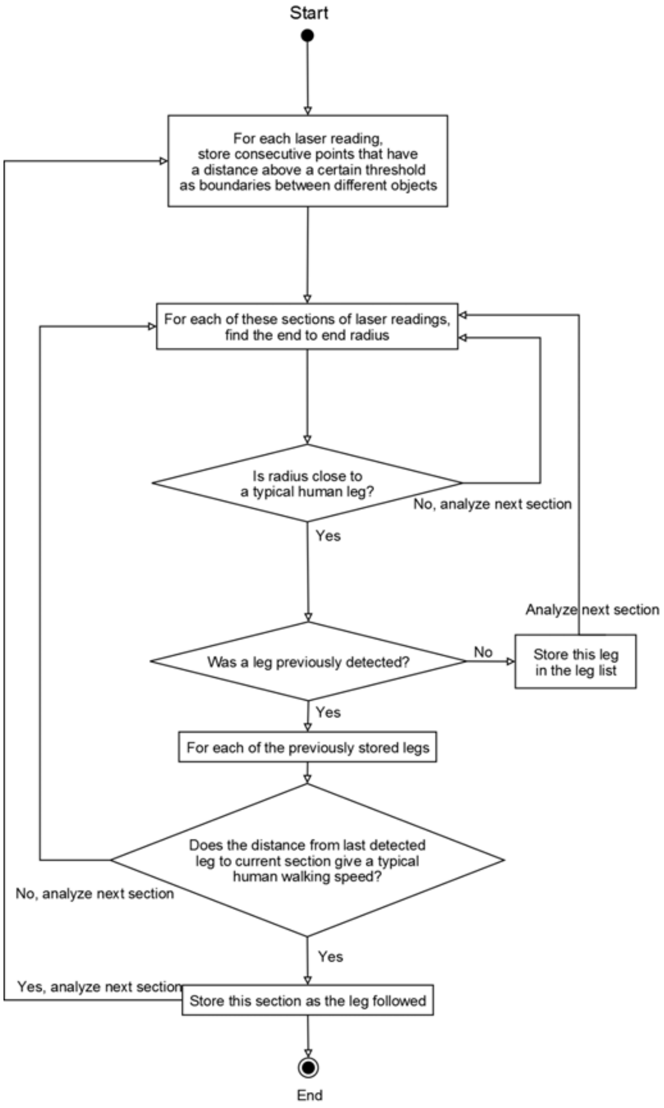


Fig. 1 Leg Detection Algorithm

The laser range readings are scanned consecutively looking for a large difference defined to be above a certain threshold, here taken to be equal to 1 m. This threshold was chosen because a moving person will typically not have an object in front of him closer than 1 m. So, any laser beam passing next to or between his legs will stop at a distance of at least 1 m above the one blocked by his legs. The boundaries we are looking for are shown in dots in Fig. 2. These also include the first and last laser readings. The regions between these boundaries constitute possible legs. Then, each region's width is compared with that of typical human's leg. If it matches, then the location of the leg and its size are stored. On the next scan, this information is used to make sure that the same set of legs is being followed.

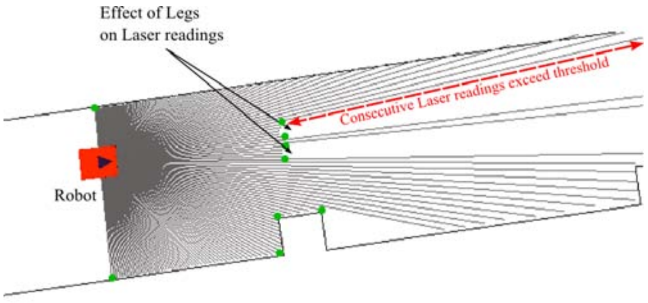


Fig. 2 The laser readings are shown as lines and the identified boundaries are marked with dots.

Once one or more legs are detected, the closest moving two legs are considered those of the person, which needs to be followed. To get the position of that person (*i.e.*, distance to the robot) , the positions of the person's two legs are averaged to obtain the center position of that person. In case only one leg was detected, the center of that leg is taken.

III. THE FUZZY INFERENCE SYSTEM

Fuzzy inference is called for when the problem to be dealt with is humanistic and complex and where the use of classical mathematical models is deemed too precise for a situation, which is imprecise by nature. Hence, fuzzy inference is most suitable when the available information is noisy, and the solution requires an approximate type of reasoning. Following a person while maintaining smoothness and safety of motion would be better dealt with using approximate rather than precise reasoning because first, it is a human or intelligent behavior that is being duplicated, and second, the data acquired by the robot from the detection of moving human legs is sometimes noisy and could be occasionally wrong.

The fuzzy inference system that is designed and used in this study to control the robot motion while following a person is based on the one developed for car-following [2]. The system has been adapted, however, from the car-following scenario to following people using a specially developed visualization program, which gives real-time feedback in response to direct manipulation of the membership functions. It is important to note here that such adaptation is one of the major advantages of fuzzy logic over artificial neural networks since the fuzzy system allows redesign and modification based on its easy-to-understand linguistic description of the problem solution.

As was previously emphasized, the robot should follow the person's legs in a smooth way. That is, it should accelerate or decelerate smoothly if the legs change speed. It should also keep a safe distance to account for situations in which the person stops abruptly. Hence, the robot should remain at a larger safe distance if it were moving faster. This implies that the safe distance should be dependant on the current robot speed. After some experimentation, it was observed that a parabolic formula is the most suitable to relate the safe distance to robot speed. The empirically derived parabolic equation is as follows:

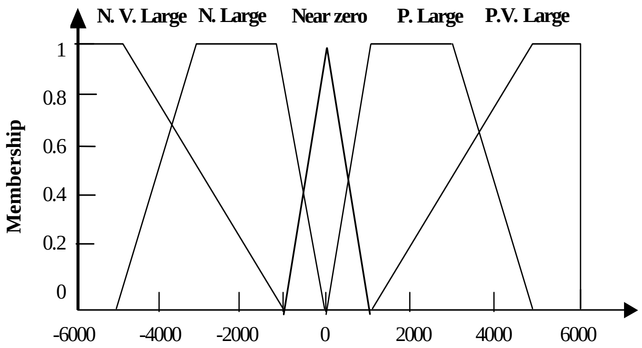
$$Safe\ Dist = 88\times 10^{-6}V^{2} + 0.66336V + Min\ Safe\ Dist,$$

where V is the current robot velocity in mm/s. The minimum safe distance was taken here to be equal to 1 m to account for the bounding radius of the used robot.

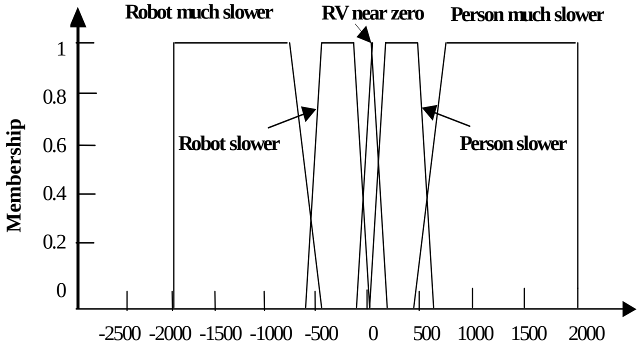
The fuzzy system uses two inputs, the relative velocity of the robot to the person's legs and the difference between the relative distance and the safe distance. The output is the needed acceleration. In order to account for distances and velocities of human beings, the ranges of the input and output variables of the fuzzy inference system as well as the ranges and shapes of the assigned membership functions, as in [2], were modified appropriately using the visualization program developed for the adaptation purpose. The final membership functions are shown in Fig. 3.

Moreover, the amount of acceleration outputted by the system is influenced by the inference rules. The rules in [2] used with the membership functions in Fig. 3, or even with those obtained before the final ones, for the case of person following produced a lot of oscillations and very slow movement towards the person when the person is obviously far away waiting for the robot to arrive. Therefore, the rules in [2] were modified in a major way to make the robot keep going forward towards the person in such scenarios. Also, they were modified to make the robot arrive to a complete stop when it is within the defined safe minimum distance. In addition, the highly negative decelerations, which were used in [2], were not applied here. While such decelerations are justified for the case of automobiles in order to avoid accidents, robots move much slower and therefore can use lower accelerations while ensuring relatively speedy and safe arrivals with minimal oscillations. The final inference rules are shown in Table 1.

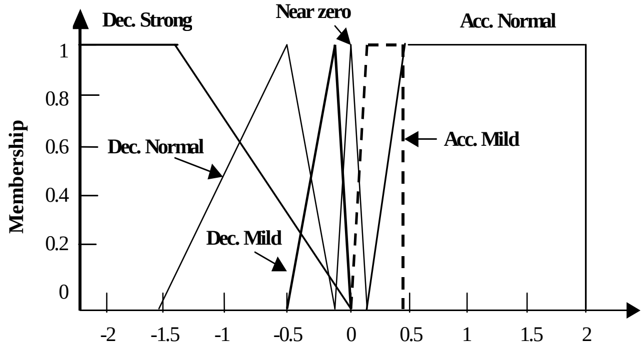
As for the defuzzification technique used, the first of maxima was tested in the beginning but it gave large values of acceleration between different acceleration levels. Going from Normal to Mild acceleration, for example, resulted in a large change in speed. This is why this method was replaced by the centroid defuzzification method, which yielded a smoother change in robot velocity.



(a) Relative distance-safe distance (mm)



(b) Relative velocity (mm / s)



(c) Acceleration/deceleration (mm / s^2) $\times 10^3$

Fig. 3 Membership functions for the fuzzy system input and output variables.

For all possible inputs, the output acceleration can be visualized in a 3D graph as seen in Fig. 4. The drawn surface shows a smooth transition in different acceleration plateaus (*i.e.*, whenever smoothness is needed). This translates into an even smoother transition in velocity given that acceleration is its derivative. Note here that the flat parts appearing in the surface are simply limits put on the boundaries of the system so that the robot will not exceed the maximum velocity once it is attained. Modeling such a non-linear function would have been very difficult using classical mathematics and without the aid of fuzzy logic, and if that were to happen, it would have been computationally expensive. Moreover, the developed fuzzy system is much easier to modify and adapt to other robot movement situations.

Table 1. Fuzzy inference rules for the robot controller

Acc/Dec		RD – SD				
		N. V. Large	**N. Large**	**Near Zero**	**P. Large**	**P. V. Large**
RV	**Robot M. Slower**	Near Zero	Acc. Mild	Acc. Mild	Acc. Normal	Acc. Normal
	Robot Slower	Dec. Strong	Dec. Mild	Acc. Mild	Acc. Normal	Acc. Normal
	RV. Near Zero	Dec. Strong	Dec. Mild	Near Zero	Acc. Mild	Acc. Normal
	Person Slower	Acc. Normal	Acc. Normal	Acc. Normal	Acc. Normal	Acc. Normal
	Person M. Slower	Dec. Strong	Dec. Strong	Acc. Mild	Acc. Mild	Near Zero

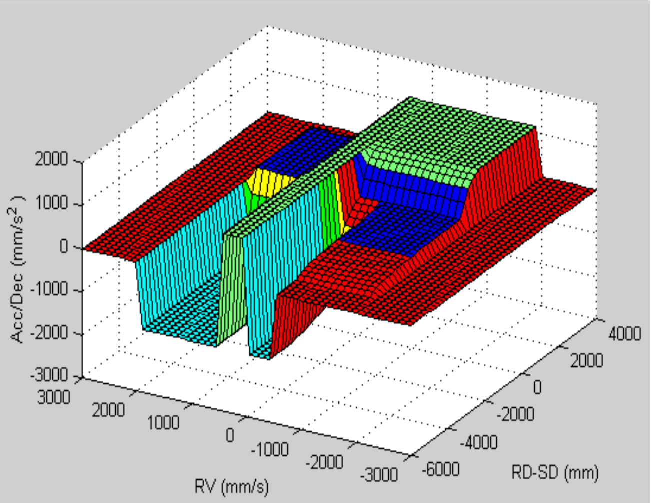
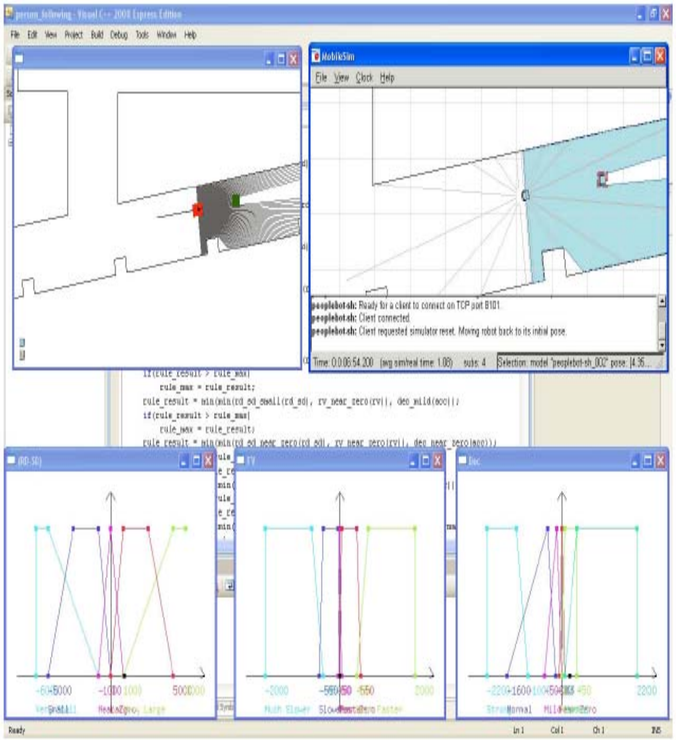


Fig. 4 Acceleration in response to all possible inputs.

Fig. 5 Lab hallway and robot where the testing is performed.

IV. TEST PLATFORM

The leg detection algorithm (Section II) and the designed fuzzy inference system (Section III) were made to work collectively to control the robot motion while following a person, and they were implemented on P3-AT Mobile Robot [10] shown in Figure 5a. The P3-AT robot is a research platform suitable for many research areas including mapping and localization. It has a Pentium 3 processor on board with USB and serial ports to mount different sensors. For the application at hand, a laser range finder [9], shown in Figure 5(a), was mounted on the robot and interfaced to the serial port of the on-board computer.



peoplebot-sh: Ready for a client to connect on TCP port 8101
peoplebot-sh: Client connected.
peoplebot-sh: Client requested simulator reset. Moving robot back to its initial pose.

Time: 0:0:06:54.200 (avg sim/real time: 1.08) subs: 4 Selection: model "peoplebot-sh_002" pose: (4.36...

```
if(rule_result > rule_max)
    rule_max = rule_result;
rule_result = min(min(rd_sd_small(rd_sd), rv_near_zero(rv)), dec_mild(acc));
if(rule_result > rule_max)
    rule_max = rule_result;
rule_result = min(min(rd_sd_near_zero(rd_sd), rv_near_zero(rv)), dec_near_zero(acc));
```

Fig. 6 Relative Algorithm Simulation and Visual Tweaking Platform. The OpenGL rendering of sensed data (Top Left), the MobileSim simulator (Top Right) and the membership functions of three fuzzy variables (Bottom three windows).

The algorithm was implemented in C++. The system sent its commands using the Advanced Robotics Interface for Applications (ARIA) [11] and then was simulated on MobileSim (Mobile Robots Simulator) [12]. After numerous successful simulations, the system was implemented in real-time on the robot.

For additional visualization, the results of laser scans, leg detection, and membership functions were drawn using the Open Graphics Library (OpenGL) [13] as seen in Fig. 6. These visualizations were made interactive or modifiable at runtime. This allowed the tweaking of the membership functions experimentally and online to find the ranges and shapes that gave the optimum results: no oscillations, smooth ride as well as reasonable time to arrive at the target (i.*e*., the person being followed).

V. RESULTS

The system was first tested on the MobileSim simulator. As seen in Fig. 7, the relative distance minus safe distance always tends to zero smoothly combating sensor noise. Also, Fig. 9 shows the velocity of the robot when the person's velocity varies and is perceived by the leg detection algorithm as shown in Fig. 8. Even though the velocity of the person's legs shows high-frequency and high velocity fluctuations, the velocity response of the robot (Fig. 9) shows much smaller fluctuations. In reality, when the system was tested on the