PALADYN Journal of Behavioral Robotics

VERSITA

Research Article · DOI: 10.2478/s13230-012-0014-3 · JBR · 3(1) · 2012 · 35-44

Fuzzy Control of a Log Carrying Robot on Tree-Filled Steep-Sloping Terrains

Richard Beranek, Aliasgar Morbi, Mojtaba Ahmadi*

Department of Mechanical and Aerospace Engineering, Carleton University, Ottawa, Canada

Received 2011/10/20

Accepted 2012/02/13

Abstract

A modular robotic system and its fuzzy logic based controller are proposed for use in logging operations in forest environments with steep slopes. The Log-Carrying Robot (LCR) concept is composed of two modular wheeled robotic agents with individual wheel steering that connect to the ends of a log to a form a centrally controlled robot. A fuzzy controller specifies the desired direction of travel using four factors: the presence of obstacles, boundaries limiting the robot's travel space, the heading of the goal position relative to the robot, and the slope of the terrain. The capabilities of the proposed controller are demonstrated in simulation using a rectangular robot with four individually actuated and steered wheels. Results indicate that the controller successfully steers the robot towards the goal position while avoiding obstacles using only eleven fuzzy rules. Additionally, the simple rules are shown to be effective at automatically compensating for sloped terrain by avoiding direct travel down hills, as well as adapting for various robot lengths.

Keywords

mobile robot · fuzzy control · behavior based control · log carrying · modular robot

1. Introduction

Logging operations are generally separated into three phases: felling, a process where trees are cut and transformed into logs; transporting the log to a landing area where logs are temporarily stored; and loading logs onto trucks to be transported from the landing area to a mill [1, 2]. The objective of the work being presented is to propose a new robotic concept and suitable control strategies that enable transporting felled trees through difficult terrains to the landing area by turning each log into a mobile robotic system.

Current solutions to transporting logs to the landing area include using vehicles known as 'forwarders' or 'skidders' to carry or drag logs to the landing area, cable systems which drag logs to the landing area, or as is often the case in steep and challenging terrain, using helicopters to transport logs to the landing area [2]. The latter solution involves high costs and difficult operating conditions for forestry workers. Forwarders and cable logging may have lower operating costs, but they have a significant environmental impact and require large initial investments. Both systems cause damage to the terrain and remaining trees, increasing the time needed for forest regeneration [1]. Forwarders and skidders also require additional paths to be created, leading to more underdeveloped trees to be cut [1].

The proposed approach is an intelligent multi-agent robotic system with each robotic agent connected at one end of the log; the goal of the robotic system is to transport felled logs from the forest to the landing area (the goal point/area of the robot). To allow the LCR to travel in forested areas, as well as to avoid the need for logging paths, the vehicle must have a small profile and high maneuverability. Additionally, the control strategy must be capable of adapting to different log lengths and mitigating the effects of sloped terrains. Such a system could potentially replace current methods at lower cost and cause a smaller environmental footprint.

The controller presented in this paper introduces a novel navigation controller capable of fulfilling these requirements. Various machine-level designs and locomotion systems can be considered, such as wheeled agents, legged agents, or leg-wheel designs, each having different advantages with respect to criteria such as ability in negotiating irregular terrains, stability, robustness, or power requirements. As this paper is primarily focused on high-level navigation issues, the approach we describe is adapted to suit our proof-of-concept LCR design. Should any other locomotion platforms be designed, the controller can be adapted to the constraints imposed by the kinematics of the new locomotion platform.

1.1. Mobile Robot Planning and Control

The principal objectives of any autonomous mobile robot controller or path planner are to avoid obstacles and navigate the robot towards a goal position. In addition, the navigator may also be required to satisfy other objectives and constraints such as finding the fastest route to the goal, satisfying kinematic and dynamic constraints, or avoiding challenging terrain.

Local path planning methods determine the desired path or trajectory of the robot based on the current state of the robot [3]. As a result, local path planning methods are more reactive, and rely primarily on sensor data. The Vector Field Histogram (VFH) [4], for example, uses sensor data to continuously update a world model used to generate control commands that steer a robot towards the goal position while avoiding obstacles. The more recent curvature-velocity method [5] and dynamic window approach [6, 7] extend the VHF method by directly taking into account the kinematic and dynamic constraints of the robot's drive sys-

*E-mail: mahmadi@mae.carleton.ca.

tem. While these approaches can safely navigate robots in unknown environments at relatively fast speeds, they remain susceptible to local minima [7]; in some situations, these approaches can drive a robot to a rest position different from the goal position.

Behavior-based controllers [8, 10] and fuzzy logic based controllers [11, 16] are also commonly used for autonomous mobile robots. Behavior-based control architectures consist of a set of task-specific behavior modules (e.g. goal seeking, wall following, obstacle avoidance, etc.) that each propose control actions based on their respective sensor inputs; an arbitration scheme uses the current state of the robot to determine which behavior module's control action is to be applied. In contrast, fuzzy logic based controllers rely on a set of if/then rules that encode a specific navigation strategy to specify the desired path or trajectory of the robot.

A fusion of both approaches has also been shown to be effective [8, 10]; some hybrid approaches use fuzzy logic to represent behavior modules, while others use fuzzy logic in the arbitration scheme to generate weighted control actions that take into account the relative precedence of each active behavior module [8]. While these methods have been shown to be effective in experiment [8], their primary drawback is challenging controller design. In behavior-based control architectures, the challenge is in avoiding conflicting actions when numerous behavior modules are specified. Fuzzy controller design can also become challenging when a fuzzy inference system (FIS) is used to represent a complicated navigation strategy with a variety of objectives; typically, the size of the rule base is proportional to the complexity of the navigation strategy, and manually designing and tuning such a rule base is difficult and time consuming.

While it is possible to use reinforcement learning [3, 17, 18] or supervised learning [19] to automatically learn the parameters of membership functions or behavior modules, both techniques may be impractical for use on many real robots. Reinforcement learning can require a prohibitively long learning period, and the success of supervised learning is strongly dependent on having sufficient and appropriate input/output training data. Choosing to carry out the learning phase in simulation can address some of these concerns. However, there is no guarantee that a controller learned in simulation will be effective on an experimental platform. As such, it may only be practical to use learning as a means for refining an already working control strategy.

The control strategy for the proposed robotic system must be able to navigate challenging forest environments. These environments are filled with trees or other obstacles of varying shapes and sizes in random locations. Additionally, steep slopes demand that the robot explicitly avoid unsafe, direct descents towards the goal position. This work extends the approach suggested in [11] to satisfy these additional challenges imposed by the forestry application being considered; the controller chooses the midpoint of one of six, body-fixed sonar scanning cones that best satisfies the goal seeking, obstacle and boundary avoidance goals as the desired direction of travel of the robot. In addition, the proposed controller explicitly accounts for the length of the robot and the need to safely navigate extreme slopes. In contrast to other fuzzy logic based approaches, the proposed approach requires only a small number of simple rules. The proposed control strategy is also advantageous because it doesn't require a complex arbitration scheme for choosing (or fusing) the outputs of different behavior modules.

Section 2 describes the LCR design, and section 3 describes simulation model and motion constraints of the robotic system. A description of the fuzzy controller and simulation results follow in Sections 4 and 5. Section 6 concludes the paper and outlines future work.

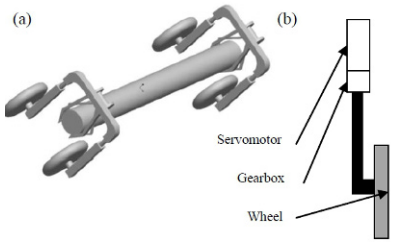

Figure 1. (a) Log carrying robot conceptual design and (b) proposed steering method for each wheel arm.

2. Vehicle concept

This section describes the LCR proof-of-concept design and simulation model. To meet the requirements of the log carrying forestry application, the robot must satisfy the following requirements: *(1) Effective at extreme downhill slopes (up to 60 degrees); (2) Autonomous navigation; (3) High maneuverability and minimal profile; (4) Suitable for rough terrain.*

The design is shown in Fig. 1(a), and consists of two, two wheeled robots. Each robot has a supporting frame, a tree hoisting and locking mechanism, and individually actuated wheels. Each wheel is mounted on actuated arm, as shown in Fig. 1(b), allowing each wheel to be steered independently. The log acts as a link that connects each unit to form the complete system. Position data is gathered from a GPS system, and obstacle distance data is provided by six sonar sensors mounted on one of the units. During operation, one unit would act as master and the other as a slave. The two units would communicate with each other using a wireless communication protocol, and an onboard PC on the master would determine desired direction of travel and the actuation commands to be sent to both the master's and slave's actuators.

3. Simulation model

The objective of the simulation is to model the key aspects of navigation in a forested environment including: randomly assorted obstacles, varying log lengths (which, by extension, leads to varying robot lengths) and the average variation of the slope of the terrain. Given that other terrain features such as bumps, depressions or general terrain roughness do not directly affect the robot's high-level navigation strategy, a planar dynamic model is sufficient for testing the high level navigation strategy.

The robot model, as shown in Fig. 2, consists of a rigid rectangular body with four independently actuated and steered wheels subject to external forces at the locations of the four tires. These external forces include propulsion forces, cornering forces, and rolling resistance. The simulation model was generated using Matlab^{TM}, Simulink^{TM}, and the SimMechanics^{TM} blockset.

The planar vehicle model is shown in Fig. 2. The vehicle is free to move and rotate in the plane, where the X-Y plane is the global coordinate frame attached to the plane of travel; i.e., if the robot is traveling on a sloped plane, the X-Y coordinates correspond to the axes defining the slope and not a global coordinate frame. Tires are connected to the main vehicle body via actuated revolute joints.

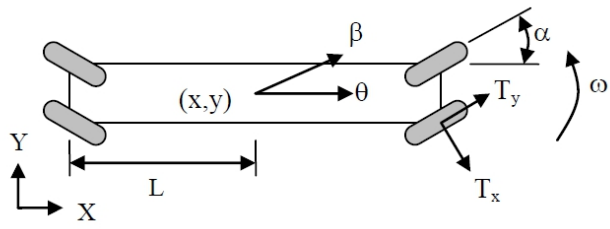

Figure 2. *Four wheel steered vehicle mode.*

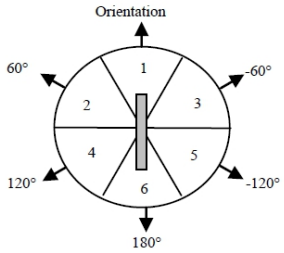

Figure 3. *Six sensing areas are defined around the robot.*

The inputs to the model are the tire forces mentioned above as well as gravitational force G. Tire forces are defined in the tire's reference frame with axis T_x and T_y and are individually computed for each tire. Driving force D and rolling resistance R act along T_y and simulate actuator force and terrain resistance, respectively. Driving force and rolling resistance are computed using (1) and (2) where s is the vehicle's speed, K is a terrain specific constant and H is a user defined gain which affect the velocity control of the vehicle. Cornering forces C act along T_x and are computed using (3) [20]; in (3), α is the tire steering angle, θ is the vehicle's orientation, β is the direction of the vehicle's velocity, ω is the vehicle's yaw rate and E is a user defined constant. Variables θ and β are computed with respect to the absolute reference frame. Given the planar modeling assumption, static friction limits are not considered for the T_x and T_y. However, when the robot is on a slope, the Y-axis component of the gravitational force can exceed the Y-axis wheel forces, resulting in side slip down the slope.
Slopes are simulated by projecting the gravity vector G on the Y axis in accordance to the slope angle φ. The variable m is the vehicle mass, and L is the vehicle half length. SimMechanics^{TM} computes the model's response to the input forces and provides vehicle data such as velocity.

$$D = Hs \tag{1}$$

$$R = Kmg \tag{2}$$

$$C = E(\alpha - (\beta - \theta)) - \frac{\omega L}{s} \tag{3}$$

$$G = \cos(\varphi)\hat{j} + \sin(\varphi)\hat{k} \tag{4}$$

$$T_x = C \tag{5}$$

$$T_y = D - R \tag{6}$$

An ideal suspension is assumed because terrain irregularities are not considered. The effects of tire slippage and other terrain aspects such as local surface height variations and underbrush are also not considered in this simulation.

3.1. Sensing

During simulation, obstacle distance data is generated assuming 6 range sensors with non-overlapping, 60° sensing cones and 6m ranges are mounted at the center of the robot. These sensors provide the distance from the robot's geometric center to the closest obstacle within a sensing cone. An illustration of the sensor arrangement can be seen in Fig. 3. In addition, the controller requires that the robot's position, velocity, orientation and the terrain slope be known. This data can be measured with a combination of GPS, inertial navigation system (INS) and inclinometer data.

3.2. Steering Constraints

The robot's four individually actuated and steered wheels allow for increased maneuverability. However, only a combination of car-like motions and spin steering are considered in this study. Car-like steering motion is generated by fixing the rear wheel steering angles and varying the front wheel steering angles; when traveling backwards, the reverse strategy is applied. This type of motion is used whenever obstacles are identified to be far (i.e., when the risk of collisions is low). When car steering is active, the reference velocity of the robot is set to a predefined maximum value. As shown in (1), the reference velocity is tracked using a proportional controller.
Spin steering is initiated whenever the distance to the closest obstacle falls below a user-defined threshold. It involves slowing down the robot to a stop, and turning the front and rear wheels in equal but opposite directions to allow the robot to rotate about its centre. Spin steering is activated to avoid collisions in situations where the robot is traveling too fast to maneuver around obstacles using car-like steering.

4. Fuzzy controller

The goal of the fuzzy controller is to navigate logs of different lengths to a goal position/area (denoted by x_{ref}, y_{ref}) and to ensure that the LCR avoid obstacles and is capable of traveling down steep slopes when traveling to the goal position. The overall control strategy is summarized in the control loop diagram shown in Fig. 4.
The output of the fuzzy controller is a desired direction of travel that is selected by evaluating a scalar we call "traversability" for each of the 6 sonar scanning cones illustrated in Fig. 3. Traversability is a measure of the desirability of traveling along any of the directions contained within a sensing cone. It is evaluated based on four factors: the presence of obstacles around the robot, the distance to the boundaries limiting the robot's allowed travel space (permitted field of operation), the

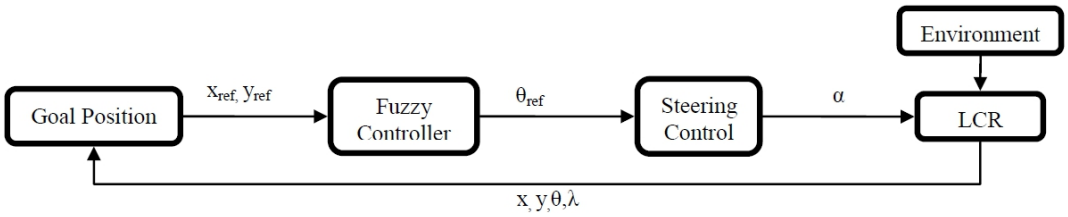

Figure 4. *Control Strategy block diagram.*

orientation of the goal position relative to the robot's current orientation, and the slope of the terrain. The cone with the highest traversability is ultimately used to determine the desired direction of travel of the robot. Since the controller assigns no particular preference to any one direction contained within a cone, the midpoint of the cone with the highest traversability is taken as the desired direction of travel. The desired direction is then decomposed into corresponding steering commands for each of the robot's individually actuated wheels.

In the current implementation, the fuzzy controller only specifies the desired orientation of the robot – the desired forward speed is kept constant and is selected in accordance to the sonar sensor range to ensure that the robot has sufficient time to stop in the event that an obstacle is detected at the boundary of the range. However, a similar rule base could be used to automatically assign the desired speed of the robot if necessary.

4.1. Input Fuzzyfication

A Mamdani-type FIS evaluates the traversability of each sonar cone. The membership functions for input fuzzyfication are shown in Fig. 5. The five inputs to the FIS include: (1) the distance to the nearest obstacle within the cone. Smaller obstacle distances have the effect of reducing the traversability of the cone; (2) the distance of the cone's midpoint to the motion boundary. Motion boundaries restrain the areas within which the robot can travel. These can simulate borders limiting the current logging area or boundaries constraining the robot to a user defined path; (3) the absolute value of the angle between the cone centerline and the line joining the centre of the robot and the goal position. Small angles correspond to high cone traversability as they indicate that the robot is heading in a direction that brings it closer to the goal; (4) the smallest obstacle distance in the cones the robot will travel through to reach the midpoint of the cone being evaluated. An obstacle at a distance close to or below the robot's half-length corresponds to low traversability. As discussed below, the rules associated with this input help avoid collisions that may result during spin steering; (5) the difference between the desired effective slope and the effective slope E_f that would result from traveling along the direction specified by the cone's midpoint. Effective slope is the slope angle of a path along a particular heading angle θ as define by (7). As seen in Fig. 6, a path directly down a slope, or along the Y-axis, has an effective slope equal to the terrain slope φ and results in the highest possible value for effective slope (the extreme effective slope). A path perpendicular to the slope, or along the X-axis, has an effective slope of 0°. Traversability is increased if traveling along the direction specified by the cone's midpoint brings the robot closer to tracking the desired effective slope.

$$E_f = abs((\cos(\theta - \lambda))^* \varphi \qquad (7)$$

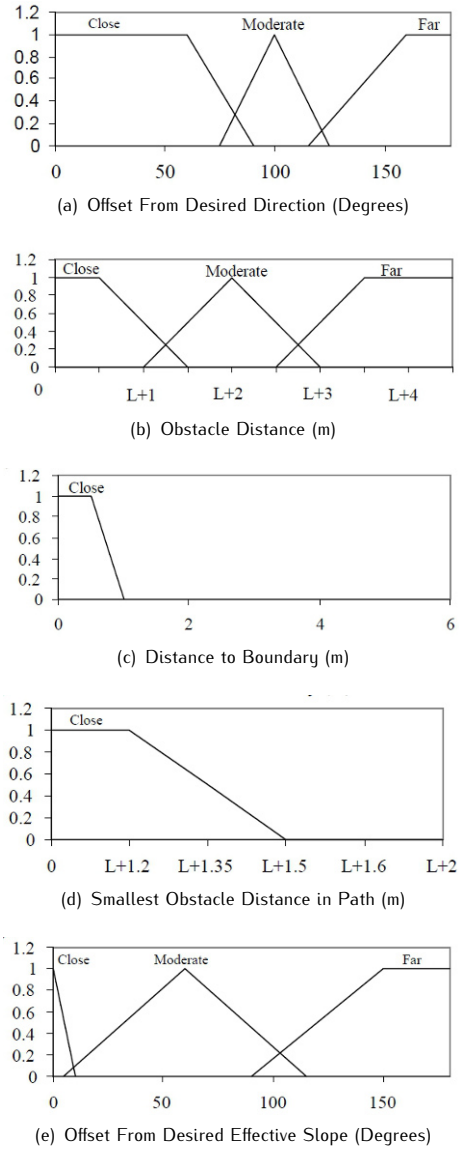

Figure 5. *Input membership functions.*

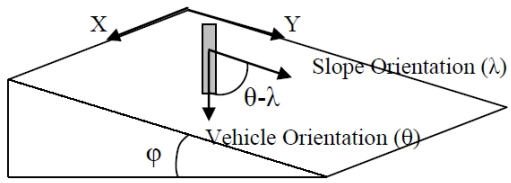

Figure 6. *Effective slope.*

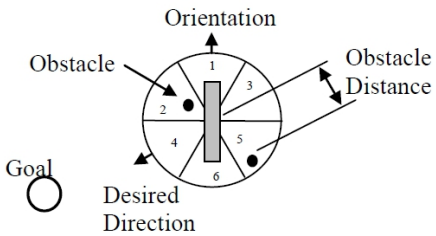

Figure 7. *Controller inputs.*

4.2. Adapting to different robot lengths

As seen in Fig. 5, the membership functions for obstacle distance are made a function of the robot's half-length, *L*. This allows the controller to adapt to different lengths by ensuring that obstacles will be considered 'close' or 'far' at different distances when the length of the robot changes. For example, an obstacle distance of 4 m will be considered 'close' for a 6 m long robot, but 'far' for a 2 m long robot.
The traversability of a cone is also made dependent on the obstacle distances in the surrounding cones. Not accounting for obstacles in surrounding cones is likely to result in collisions during spin steering. In Fig. 7, when considering only the obstacle distance, and goal location, cone 4 would appear to have the highest traversability. However, the close proximity of the obstacle in cone 2 would initiate spin steering; as such, the robot would likely collide with the obstacle in cone 2 as it rotates towards cone 4 from cone 1. Thus, the rules associated with 'smallest obstacle distance in path' have been designed to avoid such collisions. For the scenario in Fig. 7, these rules would avoid potential collisions by ensuring that the traversability of cone 4 is sufficiently reduced. Similar to the membership functions for 'obstacle distance', the membership functions for 'smallest obstacle distance in path' are also made a function of the robot's half-length, *L*. This means that for a given obstacle arrangement and robot orientation, the rules associated with 'smallest obstacle distance in path' will more quickly reduce the traversability of a cone as the length of the robot is increased. This is necessary because collisions due to spin steering are more probable for longer robots.

4.3. Adapting to different terrain slopes

Tracking a desired effective slope gives a means for avoiding excessive acceleration on extreme slopes as it forces the robot to avoid a path directly down a steep slope. This is similar to how a skier moves downhill at an angle that constantly varies with respect to the steepest direction in a zigzag-like pattern. The choice of the desired effective slope to be tracked by the controller is dependent on surface contact

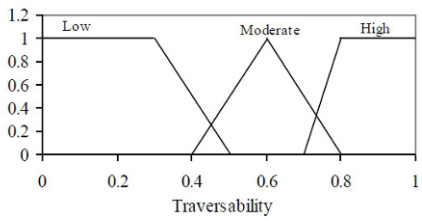

Figure 8. *Output (traversability) membership.*

properties, vehicle loading and the actuation system capabilities. With proper selection of the effective slope for the vehicle, an appropriate balance between using gravity to accelerate the vehicle downwards to reduce power requirements and avoiding excessive strain on the braking system can be reached.

4.4. Rule Description and Output Defuzzuyfication

All 11 of the fuzzy rules that evaluate each cone's traversability consist of simple and direct connections between a single input and its corresponding traversability. Output (traversability) membership functions are shown in Fig. 8. The rules used by the controller include:

1. *If obstacle distance is close then traversability is low*
2. *If obstacle distance is moderate then traversability is moderate*
3. *If obstacle distance is far then traversability is high*
4. *If offset from desired direction is close then traversability is high*
5. *If offset from desired direction is moderate then traversability is moderate*
6. *If desired direction offset is far then traversability is low*
7. *If distance to boundary is close then traversability is low*
8. *If smallest obstacle distance in path is close then traversability is low*
9. *If offset from desired effective slope is close then traversability is high*
10. *If offset from desired effective slope is moderate then traversability is moderate*
11. *If offset from desired effective slope is far then traversability is low*

Complicated composite rules such as those applied in the approaches described in [12, 13, 16] are not necessary because the controller only evaluates the traversability of each cone rather than using sensor data to directly specify the desired direction of travel. In the latter approach, the rules naturally become more complicated as the multitude of different situations (e.g., all the possible obstacle arrangements, terrain slopes, boundary distances, etc.) the robot could potentially face must be accounted for within the rule base.

Table 1. Model parameters used for simulation.

Parameter	Value
H	0.5 [N/(m/s)]
K	0.3
E	0.1 [N/rad]
Tree Radius	0.3 to 0.8 [m]

The min and max operators are used for implication and aggregation, and the centroid method is employed for output defuzzyfication. During defuzzyfication, the rules associated with 'obstacle distance' and 'smallest obstacle distance in path' are weighted three times more strongly than the remaining rules. This is done to ensure that obstacle avoidance takes precedence over the remaining goals and constraints.

4.5. Artificial Obstacles

Since the robot is only aware of the obstacles in its immediate vicinity, it is possible for the robot to get trapped in a repeating loop [7, 13]. As shown in Fig. 9, a goal position located behind a large U-shaped obstacle is one example of a scenario that generates this kind of behavior. To prevent the robot from traveling in repeating loops in these types of situations, the robot's position is kept in memory. If the robot travels over the same position twice (within a tolerance bound), an artificial obstacle is generated directly behind the robot. This location is stored in memory, and sensor data is then altered to indicate an obstacle in the position of the artificial obstacle anytime the robot revisits the surrounding position (within the same tolerance bound). These altered sensor readings provide a disturbance that lets the robot explore different paths that ultimately help it escape the repeating loop.

5. Simulation results

Numerous simulation test cases were run to determine the controller's ability to navigate the robot under different conditions. In all of the following figures, a small circle at one end of the rectangle representing the robot specifies the front end of the robot. Only different arrangements of varying diameter circular obstacles were considered during testing. The model parameters used for the simulation are listed in Table 1.

5.1. U- Shaped Obstacle

In this test, a U-shaped obstacle arrangement is placed between the goal and the initial position as shown in Fig. 9. Since the robot is only aware of the of the obstacles in its vicinity, it initially cycles between moving away from the wall to avoid obstacles and heading directly towards the wall again when the obstacles lie just outside the sensors' range; as shown in Fig. 9(a), the robot will continue this behavior indefinitely unless artificial obstacles are present. In Fig. 9(b), artificial obstacles were generated anytime the robot traveled to the same position twice (artificial obstacle positions are denoted by '*' symbols in Fig. 9). This artificial obstacle then remains in memory. When the robot is within sensor range of the artificial obstacle, the sensor data is modified to include the presence of the artificial obstacle in the controller. The robot ultimately reaches the goal position once the artificial obstacles have provided sufficient influence to allow it to escape the U-shaped obstacle. In the case where several LCR's are acting in the same area,

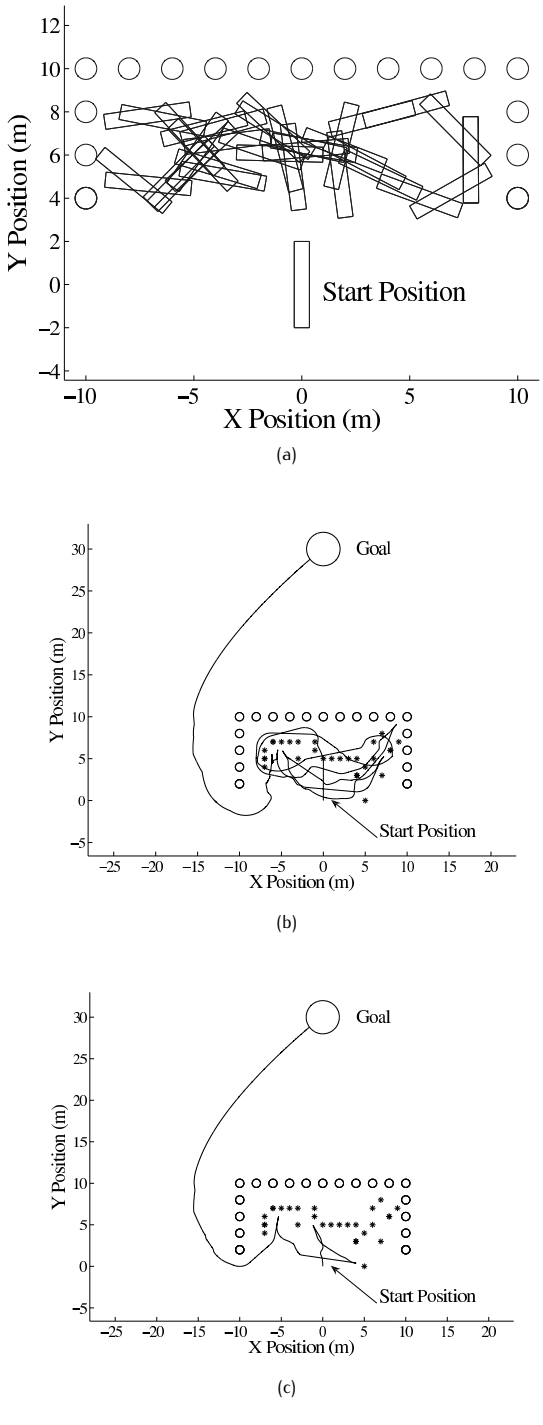

Figure 9. U-Shaped obstacle test conducted without artificial obstacles (a) and with artificial obstacles (b), with learned artificial obstacles (c).

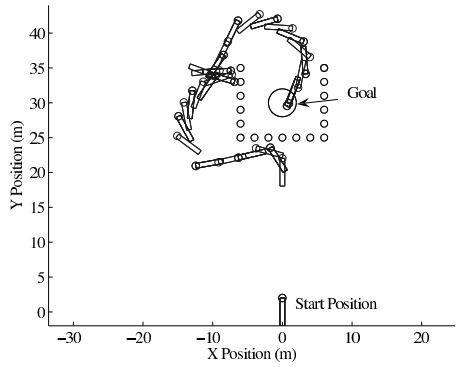

Figure 10. *Covered goal test.*

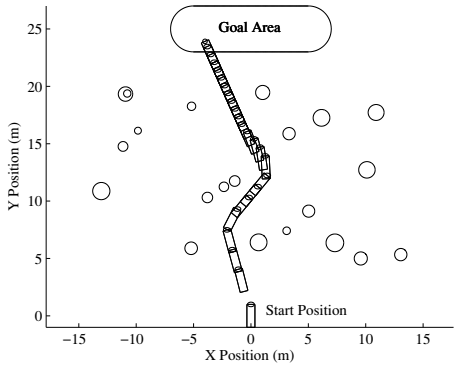

Figure 11. *Large goal area.*

artificial obstacle positions could be communicated between robots so that problematic areas, such as the U-shaped area, are avoided without repeating the same process. This concept was tested by running an additional test where the robot was given knowledge about the artificial obstacles generated in the previous test. The result shown in Fig. 9(c) indicate that the robot exits the U-shaped obstacle significantly quicker than its predecessor. Hence, artificial obstacles may be communicated between robots to progressively increase the efficiency of their paths towards the goal position. As expected, additional testing indicated that more artificial obstacles were required as the size of the U-shaped obstacle was increased.

5.2. Covered Obstacle

This test required the robot reach a goal position surrounded by obstacles on three sides. As seen in Fig. 10, the robot successfully navigated around the obstacles to reach the goal position. Accordingly, this test indicates that the controller can successfully avoid obstacles and reach the goal position. However, this test also reveals that the controller is not guaranteed to generate the shortest or fastest path towards the goal. The results in Fig. 10 indicate that the robot is prone to traveling too far to avoid obstacles and then overcompensates by heading directly towards other obstacles in order to take the most direct route to the goal position. This motion also highlights a spin-steering manoeuvre. To point towards the goal, the robot stops and turns in place, before turning back. Given that the robot does not have any forward motion, it can easily turn in place and change its heading without colliding with obstacles.

The controller's limited knowledge of the environment is the primary cause of selecting an inefficient path, and this issue is common to most local path planning and control methods; by only relying on data about the immediate surroundings, the controller does not have enough foresight to plan a better trajectory. Using sensors with a larger range or a supervisory controller that generates a series of intermediate waypoints from a coarse map are potential solutions to this problem. Wall following behaviors [13] can also be used to generate reference trajectories that allow the robot to follow to closely follow the wall, rather than taking a less direct path to avoid the wall as in Fig. 10. Finally, increasing the number of membership functions may also help smooth the motion of the robot around corners.

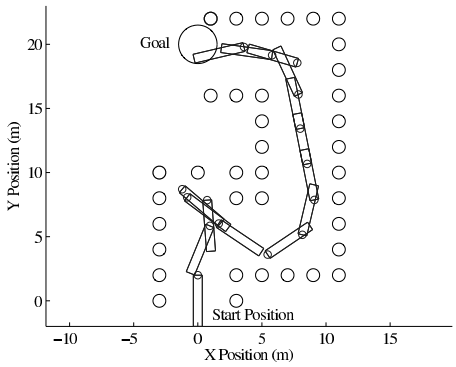

Figure 12. *Constrained path test route.*

5.3. Large Goal Area and Constrained Path

The robot would likely be traveling to a landing area (instead of a goal point), where a large open space is available for temporary log storage. Rather than defining the goal as a point as in previous tests, a rectangular goal area was considered in this test. As seen in Fig. 11, the robot successfully reaches the goal area without taking an excessively long path towards it.

The constrained path test is meant to assess the controller's effectiveness in an environment where only one possible path is available. Such a situation may simulate the environment created by a small path in the forest or the landing area where stacks of trees have created limited paths for the robot to travel in. As seen in Fig. 12, the robot successfully navigates through the passable area to reach its goal. This test also demonstrates the robot's reverse driving ability. As it reaches the end of the first corridor, it changes driving direction and continues the rest of its trip to the goal traveling in reverse. Thus, difficulties associated with re-orienting the front end of the robot in tight spaces is avoided because the robot is able to negotiate the environment equally well traveling forwards or backwards.

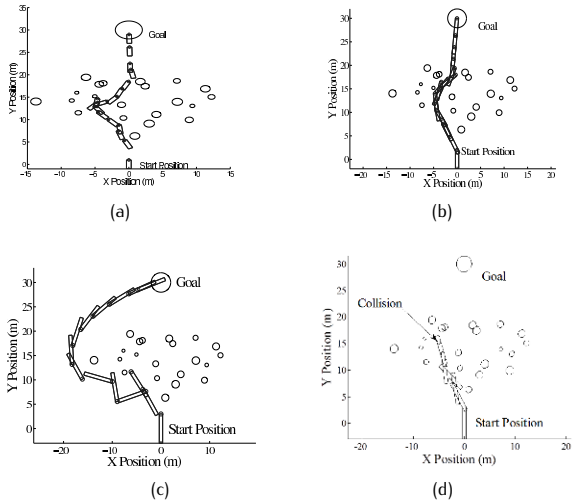

Figure 13. Adjustable length tests, (a) 2 m robot, (b) 4 m robot, (c) 6 m robot, (d) 6 m robot with collision.

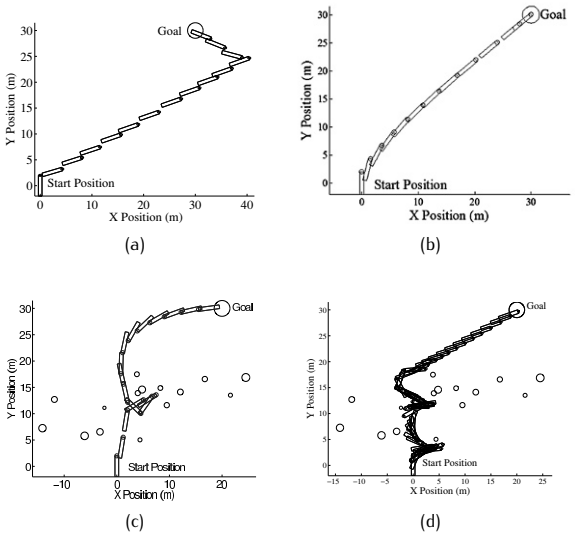

Figure 14. Sloped surface vs. flat surface behavior, (a) 45° slope, (b) 0° slope, (c) 0° slope with obstacles, (d) 45° slope with obstacles.

5.4. Adapting to Different Lengths

This test illustrates the controller's ability to adapt to different robot lengths (or log lengths, in the case of a robotic locomotion system for transporting logs). The obstacle field was randomly generated and three different robot lengths were tested. Since the key membership functions associated with obstacle distance automatically vary with the length of the robot (see Section III), the controller successfully accounted for the changes in the robot's length and ensured that the goal was reached without any collisions; this result can be observed in Fig. 13(a), (b), (c) where the 2 m, 4 m and 6 m long robots each successfully navigate to the goal in the same test environment. For comparison purposes, the 6 m long robot was retested with fixed membership functions tuned for a 2 m long robot. The results shown in Fig. 13(d) indicate that the long robot takes a very aggressive path through the obstacle field that eventually results in collision stopping the robots' motion. Accordingly, the results in Fig. 13(d) reinforce the importance of parameterizing membership functions with respect to the robot's length.

At longer vehicle lengths the controller had difficulty generating a direct path to the goal. As seen in Fig. 13(a) and (b), the 2 m and 4 m long robots found a path through the obstacle field and successfully reached the goal. However, Fig. 13(c) shows that the 6 m long robot traveled around the field to reach the goal. In general, more conservative trajectories will naturally emerge for longer robot lengths. This occurs because the membership functions are automatically varied in a manner such that the distance at which an obstacle is considered 'close' increases when the robot's length is increased. Thus, for a given orientation, obstacle density, terrain slope, and boundary location, the traversability value of any cone will drop rapidly as the length of the robot is increased. In some situations, this feature is advantageous because the probability of collisions increases when the robot's length is increased.

5.5. Sloped Terrain

This test examined how the robot changed its behavior on sloping surfaces. The robot's ability to steer towards the goal on a 45° slope when no obstacles are present in its path is illustrated in Fig. 14(a); the resulting slalom-like motion is due to the robot tracking a desired effective slope of 15° along its path. Also, to avoid the extreme effective slopes, the robot reverses the direction of travel. This allows the robot to continue tracking the desired effective slope without taking turns (U turns) that result in the vehicle seeing large effective slopes. In contrast, it can be seen that the robot takes a very direct path to the goal when the same test is repeated with flat terrain and a surface slope of 0° in Fig. 14(b). In Fig. 14(c) and (d) obstacles are added to the surfaces. On a flat surface, shown in Fig. 14(c), the robot tries to take the most direct path through the obstacles. In Fig. 14(d), the same obstacles are placed on a 45° slope. Rather than traveling directly down the slope, the robot chooses a path that simultaneously tracks an effective slope of 15 degrees and avoids obstacles. Again, an overall slalom-like motion can still be seen in the robot's trajectory; when obstacles are not obstructing the robot's path, backwards driving is activated when changing the direction of travel. The actual effective slope and desired effective slope are plotted in Fig. 15 for the 'slope with obstacles' test shown in Fig. 14(d). The maximum peaks show when the robot traveled along the extreme effective slope. This occurs when obstacle avoidance takes precedence and the robot is forced to deviate from the desired effective slope. Otherwise the robot successfully stays on paths close to or below the desired effective slope.

5.6. Large Field

This test subjects the robot to a larger field of obstacles, testing the controller's ability to find a path to the goal position with a larger gap between the start position and goal position. As seen in Fig. 16(a), the robot successfully navigates the field to reach the goal position while avoiding obstacles; when a 45° slope is added, the robot exhibits the

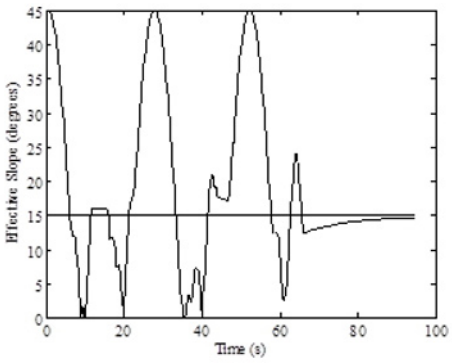

Figure 15. *Actual effective slope compared to desired effective slope.*

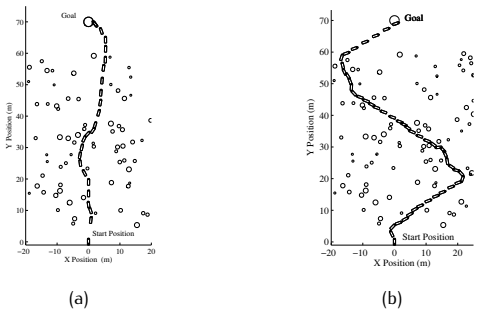

Figure 16. *Large Field Test, (a) 0° slope, (b) 45° slope.*

slalom-like motion shown in Fig. 16(b). These results indicate that the controller has the ability to successfully and efficiently navigate large, unstructured environments without requiring intermediate waypoints. Additionally, these tests further demonstrate the controller's ability to automatically balance obstacle avoidance and effective slope tracking.

5.7. Sensor Noise Robustness

In this experiment, the robustness of the controller to sensor noise is examined. The use of fuzzy sets generally allows for greater noise robustness, given that small variations in the sensor signal will not cause large changes in the membership function value. In this experiment, a random obstacle field with 21 obstacles is generated and the robot must cross it to reach the goal position. Noise is added to the obstacle distance measurement. This noise can come from a number of sources, such as the resolution of the sensor, reflections of ultrasonic signals or readings from other features such as leaves and branches. The results for two noise levels are shown in Fig. 17. In Fig. 17(a), there is no sensor noise applied. Fig. 17(b) shows the path taken by the robot with normally distributed white noise with a standard deviation 0.3 m on the obstacle distance measurement. This noise level was found to be the maximum noise level with which the robot could successfully complete the task without collision. This result suggests that the fuzzy

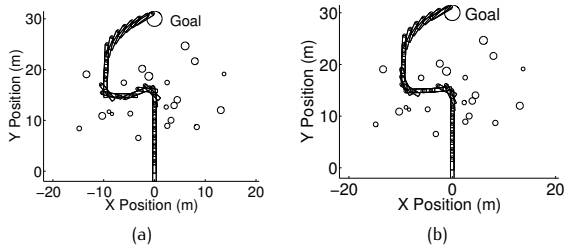

Figure 17. *Robustness to sensor noise test, (a) 2 m robot crosses a random obstacle field with no sesnsor noise, (b) 2 m robot crosses a random obstacle field with normally distributed white noise with a standard deviation of 0.3 m.*

based controller can robustly navigate with a noise level up to 5% of the maximum sensor range.

6. Conclusion

A fuzzy controller for an autonomous mobile robot targeted for log carrying applications in mountainous terrain is introduced. The controller operates on the principle of constantly re-evaluating its immediate environment, and choosing the best direction of travel from 6 possible choices. Simulation results show that the controller successfully navigates an autonomous mobile robot of adjustable length in a variety of environments while satisfying the following goals and constraints: goal seeking, obstacle and motion boundary avoidance, and controlled descent on sloped surfaces. Future work will focus on generating more efficient methods for exiting repeating loops and introducing more rules to naturally elicit wall following behaviors. Additionally, the simulation will be updated to include 3D terrain and suspension system dynamics. New rules for further exploiting the maneuverability of the platform will also be developed. A robotic platform for validating simulation results is currently under development.

References

[1] Nelson C. Brown, *Logging*. New York, New York: John Whiley & Sons, 1949.

[2] Ross Selversides, *Broadaxe to flying shear: the mechanization of forest harvesting east of the Rockies*. Ottawa, Canada: National Museum of Science and Technology, 1997

[3] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," IEEE Trans. Systems, Man, and Cybernetics, vol. 25, pp. 464–477, Mar. 1995.

[4] J. Borenstein and Y. Koren, "The vector field histogram - fast obstacle avoidance for mobile robots," IEEE J. Robotics and Automation, vol. 7, pp. 278–288, June 1991.

[5] T. Quasny, L. Pyeatt, and J. Moore, "Curvature-velocity method for differentially steered robots" in 2003 Proc. Modelling, Identification, and Control.

[6] O. Brock and O. Khatib "High-speed navigation using the global dynamic window approach," in Conf. Rec. 1999 IEEE Int. Conf.

on Robotics and Automation, pp. 341-346.
[7] P. Ögren and N. Leonard, "A convergent dynamic window approach to obstacle avoidance," IEEE Trans. Robotics, vol. 21, pp. 188–195, Apr. 2005.
[8] H. Seraji and A. Howard, "Behavior-based robot navigation on challenging terrain: A fuzzy logic approach," IEEE Trans. Robotics and Automation, vol. 18, pp. 308–321, June 2002.
[9] W. Li, "Fuzzy logic-based 'perception-action' behavior control of a mobile robot in uncertain environments," in Conf. Rec. 1994 IEEE Int. Conf. on Fuzzy Systems, pp. 1626-1631.
[10] A. Saffiotti, "Fuzzy logic in autonomous robot navigation a case study," Université Libre de Bruxelles, Brussels, Belgium, TR/IRIDIA/92-25, 1997.
[11] T. Lee, L. Lai, and C. Wu, "A fuzzy algorithm for navigation of mobile robots in unknown environments," in Conf. Rec. 2005 IEEE Int. Symp. Circuits and Systems, pp. 3039-3042.
[12] K. Song and J. Tai, "Fuzzy navigation of a mobile robot," in Conf. Rec. 1992 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 621-627.
[13] B. Chee, S. Lang, and P. Tse, "Fuzzy mobile robot navigation and sensor integration," in Conf. Rec. 1996 IEEE Int. Conf. on Fuzzy Systems, pp. 7-12.
[14] X. Yang, M. Moallem, and R.V. Patel, "An improved fuzzy logic based navigation system for mobile robots," in Conf. Rec. 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 1709-1714.
[15] X. Yang, M. Moallem, and R.V. Patel, "A novel intelligent technique for mobile robot navigation," in Conf. Rec. 2003 IEEE Int. Conf. on Controls Applications, pp. 674-679.
[16] L. Doitsidis, K. P. Valavanis, and N. C. Tsourveloudis "Fuzzy logic based autonomous skid steering vehicle navigation," in Conf. Rec. 2002 IEEE/RSJ Int. Conf. on Robotics and Automation, pp. 2171-2177.
[17] N. Yung, D. Wang, and C. Ye, "An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning," IEEE Trans. Systems, Man, and Cybernetics, vol. 29, pp. 314–321, Apr. 1999.
[18] C. Ye, N. Yung, and D. Wang, "A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance," IEEE Trans. Systems, Man, and Cybernetics, vol. 33, pp. 17–27, Feb. 2003.
[19] P. Rusu, E. Petriu, T. Whalen, A. Cornell, and H. Spoelder, "Behavior-based neuro-fuzzy controller for mobile robot navigation," IEEE Trans. Instrumentation and Measurement, vol. 52, pp. 1335–1340, Aug. 2003.
[20] M. Mostavi, M. Shariatpanah and R. Kazemi "A novel optimal four wheel steering control" in Conf. Rec. 2004 IEEE Int. Conf. on Industrial Technology, pp.1598-1601.