Technical Report

for

The 5th Annual International

Autonomous Ground Vehicle Competition

Submitted to:

Mr. William G Agnew
3450 31 Mile Road
Romeo, Michigan 48095

Submitted by:

Team-B from The University of Tulsa
Dan Hemphill
Earl Hammon
Brent Hendeson
Travis Day
Sean McBeath
Kaveh Ashenayi, Faculty Advisor
Email: kash@ohm.ee.utulsa.edu

1.0 Introduction

The University of Tulsa is fielding two teams for the 1997 Ground Robotics Competition. This report will discuss the car being built by Team B. The team and its car are in their second year of competition. The goal for last year's car was a simplistic design that utilized as many off the shelf components as possible. The team members were assembled with the idea of having them together over a span of two or three years instead of just one.

These goals were all met and this year, the team will redesign the electronics to increase durability, decrease cost, allow for more efficent use of space, and make the car easier to maintain. There are five different subsytsems of the car being considered for redesign, along with two subsystems that will undergo minor changes.

2.0 The Team

The team is composed of five members. All members from the previous year's team are returning to this year's contest. The team's faculty advisor is Dr. Kaveh Ashenayi, captain--and only graduate student--is Daniel Hemphill, seniors Travis Day and Earl Hammon, and juniors Brent Henderson and Sean McBeath round out the team. Each member was picked for their specific skills and their ability to function in a team setting.

3.0 The Chasis

The car was built from the ground up. The chasis is constructed from hollow steel tubing to make the frame strong but relativly light with materials on hand. The frame is in the form of a rectangle 3 feet wide and 4 feet long. A tube cuts the car in half lengthwise. In each smaller rectangle, diagonal tubes are welded to support the platform. An aluminum platform in the middle portion of the frame provides the foundation for all the components. Two motor mounts are located at the middle on each side of the car. Tire mounts are located at each corner of the car. Four pole mounts are welded to the frame, two are by the motor mounts, at a 90 degree angle to the frame, while the other two are mounted at the rear corners of the car at 40 degree angles. Steel plumbing tubes were then used to construct the camera mount. The results give us a roll cage similar to a jeep, and keeps the camera very stable.

Previously the batteries were held in place using aluminum framing. This year, the 2x4 framework of the plywood box covering the platform will hold the batteries, allowing us to add a fourth battery. The electronics will be moved out of the computer frame they were in, and placed in a plexiglass box. The box will be smaller than last year, and will allow easier access to the different components. Also, the plexigalass box will have a fan on it to eliminate excess heat. The plywood box will also have a fan on it to prevent any heat related damage to the PC and/or the DC-AC inverter.

4.0 The Drive Train

This is the first part of the car being looked at for major modifications. Since the goal was to keep the car simple, a chain drive and differential steering was used. From the experience at the competition, differential steering is the best choice, considering the high speed the car runs at. The motors used on the car are 24 volt DC motors like those found on a wheel chair. They both have electromechanical brakes and run at 3600 rpm with an output of 150 horsepower. These motors, when geared down, give us the high torque we determined necessary to handle the inclines and the sand pits present on the course.

The car is driven by two motors. Each motor is connected by chain to the pair of tires on each side. Due to the small tires the chain sprocket on the wheel axle was only slightly smaller in radius than the tire itself. A 3 to 1 sprocket ratio was chosen to keep the car within the speed limit, but this made us unable to manage the sandpit during last years competition. The chain contacted the sand causing a lock-up of the right motor. Two solutions are being considered for this problem: smaller sprockets or replacing the chain drive with a gear box.

A gear box that would reduce the motor speed to the necessary 5mph would be the best choice. A motor could then be connected directly to each tire or just two tires. The four tire method would also add two more batteries to keep the range from being decreased. The gear boxes would eliminate the need for chains, eliminating possible derailments of the chains and clogging by sand. The main drawback to the gear box option is cost. The extra batteries needed for the quad-drive option would create a premium on space.

Finding smaller sprockets is the option we will pursue to keep the car under budget. The drawback is finding sprockets small enough to meet our needs and tough enough to handle the stress created by the high torque. This method would prevent the chains from contacting sand or derailing due to large drop offs.

5.0 The Electronics

The electronics used can be clarified into two categories. The first category consists of what is needed to drive the motors. The second category consisits of what is needed for control of the car. In what follows, we will briefly describe the control circuit.

5.1.1 The Original Design

The next part of the car being considered for modification is the motor control hardware. The concept for the design used on the car came from Team A's car. The orginal goal was to keep the two cars compatible so spare parts could be shared. However, as designs for each car matured and uderwent changes this goal was not practiced. The motors we have drew higher startup current, which in turned required bigger relays. The use of different microprocessors also lead to the need for changes.

5.1.2 The New Design

During testing and competition, current spikes caused by the motors would destroy the transistors and optical isolators, thus the electronic controls will be given a separate current path and their own small fuse.

A new design is being tested to eliminate the expensive isolators. The 12 volt and 24 volt sources currently have separate grounds. The first test is to see if the two grounds being connected will interfere with performance. If no problems arise from this, the isolators can be removed. The current thought is to use a 7402(2 input NOR) gate to protect the microcontroller from feed back, and to drive the transistors.

5.2 New Features

Two new features being considered are solar cells and an onboard battery charger. The power inverter used to supply the computer with 120 VAC shuts down if its input volatage goes below 10-11 volts. The use of the solar cells may prevent this problem, because the trickle current may be enough to augment the battery volatge enough to prevent the inverter from shutting down during testing and the competition.

The battery charger would be capable of charging all batteries at once. By flipping a switch and plugging in, the car can go from running mode to charging mode. Cost and difficulty associated with finding the needed parts may limit or eliminate the battery charger.

6.0 The Microcontroller

The microcontroller is based on the design used in a senior class offered at the University of Tulsa. It uses an 8031 Intel microcontoller and a 64K Erasable/Programable Read Only Memory (EPROM). The microcontroller uses RS232 serial communications to talk with the computer, while it uses an output port to send the commands to the electronics. The program is burned into an EPROM so it can be modified at any time. The program takes an ASCII hex code sent by the computer and compares it to the different options until a match is found. Once the match is found, a four bit number is sent to the electronics. If no match is found, the last command sent to the electronics will be repeated. The only change to the program will be an input signal from the emergency stop, discussed in the next section.

7.0 The Emergency Stop

We discovered last year at competition, our manual emergency stop (e-stop) switch was not what the judges had intended since it was ordered from a catolog and assembled by the team.

The RF e-stop, caused the most problems last year. We had trouble getting a long enough range from the RF e-stop, and the car always came to a rolling stop. We did not think much of the rolling stop, since the car never rolled more than a foot.

Both e-stops do the same thing. They cut the 24 volt supply to the electronics and the motors. Due to a feature in the motors, the car coasts for a distance before the brakes lock. Because of the incline next to the water on the course, something had to be done. The solution was a simple one: when the vision system code is turned off, a stop command is sent to the microcontroller, causing the car to immediatly stop, without coasting. Any key pushed on the keyboard does this. The keyboard was mounted to the rear of the car to serve as the e-stop. The manual or RF e-stop would then be triggered to engage the brakes. The rules for this years competition have eliminated this solution.

The solution created for this year requires a small change to the e-stop wiring and the microcontroller code. When either e-stop is activated, a 5 volt signal will be sent to the controller, causing a stop command to be sent. A stop command will continue to be sent until the e-stop is reset. The e-stop will still cut the 24 volt supply to the motors causing the mechanical brakes to engage. The old RF e-stop has been replaced with a new RF system that gives us close to 150 foot range.

8.0 The Vision System

The University of Tulsa has been involved with this competition since its inception in 1993, taking first place that year. Subsequently, all TU AUVS teams have used the same basic vision algorithm. This makes use of the knowledge that all boundary lines are white, and that all obstacles are either red or white. The basic algorithm is outlined below:

```
Take a video snapshot
Break the resultant bitmap into discrete regions
For each region
        count red/white pixels by comparing RGB values to color thresholds
        If this number > count threshold
                flag zone as blocked
        Else
                flag zone as safe
        Compare resulting blocked/safe pattern with a set of rules in memory
        If found a match
                move according to matching rule
        Else
                move according to default
```

This algorithm is effective, but depends on noiseless input. Both the discrete color thresholds and the perfect-fit-only rule assume all data is undistorted. Experience shows the color video camera is quite noisy, so particular regions might be interpreted as blocked in one frame and safe in the next. This combined with the exact match requirement means the car is likely to make the wrong decision, either by selecting a rule it should not have or by selecting the default when a rule should have matched. The exact match criterion also requires an exhaustive list of rules for effective control. Lastly, the algorithm

assumes constant lighting conditions--if a cloud blocks the sun during part of the run the computer will interpret too much of the scene as obstructed when the sun is shining and too little when it is covered.

The implementation imposed further restrictions in its hard-coded nature. To change the thresholds one had to change the code and recompile. To change the rule set one had to manually calculate the bit patterns. To change the region structure one had to completely overhaul the entire program. The program did offer several useful diagnostic features. One could filter the image against color thresholds, view the pattern the filtering process gave the classifier, and test the computer's decision for a given image.

8.1 The Current Vision System

The current algorithm kept the good ideas of the original program while fixing most of the design inadequacies. It was also adapted to detect the yellow lines added for the fourth AUVS competition. The pseudo-code for this algorithm looks like the following:

```
Recompile the rule base
Convert the region structure from ground coordinates to camera coordinates
While initial calibration is not complete
        Take a video snap shot
        Find the maximum and minimum values for each component--red, green, and blue
        Modify average minimum and maximum to initial auto-calibration values
Take a video snapshot
Recalculate fuzzy rules based on current ambient lighting compared with startup lighting
For each region
        Count red/white/yellow pixels using a fuzzy membership function for each color
        If this count > count threshold
                Flag zone as blocked
        Else
                Flag zone as safe
Classify the resulting pattern with a best fit in the rules by a neural network
```

The fuzzy color thresholding and neural network classification make the algorithm noise-resistant and responsive to unpredicted situations. Also, the dynamic threshold recalibration makes the algorithm resistant to intermittent cloud cover. Furthermore the rule base is much smaller, both because of the best fit matching and because the rules were expanded to allow "don't care" regions instead of just blocked/safe ones.

The region shape on the image of the two implementations also changed. Earlier versions had used rectangular regions of the image for zones, translating into trapezoidal arcs along the ground. Instead, this version used rectangular regions of the ground for zones, translating into trapezoidal regions of the image (Fig. 3, 4). This made editing the rule set easier because one dealt with standard shapes in a standard reference frame, rather than having to translate coordinate spaces.

8.3 Fuzzy Color Membership

The line-follower also demonstrated that absolute color thresholding was inadequate because of the very faint "snow" superimposed by the electronics over the noiseless image. To handle this, a simple trapezoidal rule fuzzy membership function was created for each color component. This was implemented as three look-up tables for speed considerations.

For each color a zero-membership level and a full-membership level were calculated. Zero membership was defined as anything at or below the minimum value in the calibration regions on the fenders; full membership was defined as anything above a certain percentage of the difference between the minimum and maximum calibration values. The look-up table was then constructed by assigning 0 to all values up to and including the zero membership level, assigning 255 to all values at or above the full membership level, and linearly interpolating intermediate values.

The range [0, 255] was selected so integer arithmetic could be used instead of floating point arithmetic, thereby increasing processing speed. Essentially each value in the range [0, 1] is multiplied by 255 and rounded. The error in this process is less than 1/510, or approximately 0.2%. This means the worst-case error in summing the three look-up tables is always less than 0.6%. This loss in precision is trivial to the performance, particularly in comparison to the speed increase gained by integer arithmetic.

The input image contained 24 bits of color information, giving 8 bits each to the red, green and blue components. Therefore these fuzzy tables were each 256 bytes large, with each entry representing the fuzzy membership of an RGB component with the same value as its index. Therefore the RGB color intensity could be used as the look-up table index, and fuzzy membership could be calculated in just two assembly language integer instructions.

Once the fuzzy membership for each color component of a pixel is calculated, the computer determines if the pixel is blocked or safe. A blocked pixel is defined as anything red, yellow, or white; a safe pixel is anything else. In RGB components red is R & !G & !B, where "&" denotes fuzzy union or fuzzy AND, and "!" denotes fuzzy complement or fuzzy NOT. Similarly, yellow is R & G & !B, and white is R & G & B. For this implementation the fuzzy complement was defined as 255-x, where x denotes the membership value. This translates directly into 1-x for x in the range [0, 1] instead of [0, 255]. Also, fuzzy union was defined to be the sum of the individual memberships. Again, this translates directly into the average for the range [0, 1] instead of the range [0, 765] (765 is the high end because three components in the range [0, 255] are summed and 765 = 255*3). The alpha cut is then taken on the calculated fuzzy membership to red, yellow and white; a logical OR between these defuzzified values is used to determine if the pixel is to be considered blocked or safe.

8.4 Neural Network Classifier

For this competition an original neural network classifier was developed based on the Hamming net, itself a simple binary classifier. The Hamming net assumes a set of binary exemplar patterns, and that the input and exemplars have an equivalent number of bits. It is particularly useful in such applications as determining what letter an unknown input character is.

Basically the Hamming net counts the number of differing bits between the input and each exemplar, defining this number to be the Hamming distance. The best match is the exemplar with the minimum corresponding Hamming distance.

In the vision algorithm, the size of the exemplar patterns (henceforth "rules") is identical to the size of the input bit stream (Fig. 3,4). However, these rules have the additional *don't care* state (denoted by crosshatching), making the Hamming net no longer applicable. One could expand the *don't cares* into all possible combinations, but that would significantly increase the rule base size and slow the decision-making process. For real-time image processing and driving, this slowdown is not acceptable.

A modified version of the Hamming net was developed with the idea that treating the *don't cares* as *always matching* would handle the problem. However, the implementation revealed this biased the network towards those rules with a large number of *don't cares* in their definition. For example, a rule with 10 *don't cares*, 3 matched and 2 mismatched regions would be selected before a rule with 3 *don't cares*, 9 matched and 3 mismatched regions. In most cases, this second rule would be preferable to the first, having 75% of its significant regions matched as opposed to only 60% in the first rule.

It was also thought one might treat don't cares as "half-truths"; i.e., let *don't cares* count as halfway between a mismatch and a match. It quickly became apparent this would bias against rules with large numbers of *don't cares*.

Two possible solutions existed then. One is just to calculate a matching ratio and completely ignore the *don't cares*, and the other is to include the *don't cares* as *matches* only if at least some percentage of the other zones do coincide. The latter implementation was selected because it more closely matched the original.

The classification process, then, is to first calculate the "Hammon distance". This is defined to be the number of *matches* between the rule and the input pattern; *don't cares* are always *mismatches* at this stage. The result is compared with a precalculated threshold-- established as some fraction of the total number of zones in a rule which are not *don't cares*-- rounded up to the nearest integer. The modified Hammon distance is then calculated, which is just the Hammon distance if there are fewer *matches* than the threshold, or is the Hammon distance plus the number of *don't cares* if there are at least as many *matches* as the threshold value. Note that the Hammon distance is a measure of how well the rule matches the input, whereas the Hamming distance was a measure of how much the rule differed from the input.

To find the best match, all the Hammon net has to do is find the rule with the maximum Hammon distance. In a true neural network this would be accomplished by a maxnet on all the Hammon distances calculated in parallel; in the program it was just the global maximum of all distances calculated serially. If a subsequent node had the same distance as a preceding one, the move for the preceding node was kept.

Example:

Figures 2-4 provide a typical example input from the video camera. This particular image was saved before the fenders of the vehicle were painted. Figure 2 shows the camera input; the fuzzy membership filters applied to the red, green, and blue components of the input; and the color classification of each pixel. Figure 3 shows the zone structure mapped from the ground coordinate system to the screen coordinate system, and the same with those zones considered blocked shaded in. Typically the camera image would be directly beneath this grid. However, the Video Blaster SE 100 image is added to the VGA output continuously at display time, rather than inserted into graphics memory. Therefore screen capture utilities do not also capture the Video Blaster portion of the image, and it must be saved separately. Figure 4 shows two representative rules, those closest matching the bit pattern for this particular input. Notice the use of *Don't cares*. Also notice that it shows the camera's field of view along the ground, and limits the zone structure to that region; hence the trapezoidal editing region. The grid denotes one-foot intervals from the camera mounting.

For this image, the neural net would compare its input to the first rule. There are 18 matches and 4 mismatches, giving a preliminary Hammon distance of 18. Say the threshold for inclusion of *don't cares* is 60%; 18 of 19 zones which are not *don't cares* match, so the three *don't cares* get added. The Hammon distance is 21, with a perfect match being 22.

Similarly, it would continue checking against other rules. For the second example, there are 16 matches and 6 mismatches. This is also above the inclusion percentage, so the Hammon distance is 19. The computer would select the first rule, if these were the only options.(may need a break here)

It was in first place after each of the first two runs, even though early in the first run the camera came unplugged, essentially blinding the vehicle. After the second run it was more than twice as far along the course as the next vehicle, stopped only by the sand trap.

The rest of the afternoon was spent modifying the rule set so it could navigate the sand; it successfully did so in the practice course multiple times. In the third run another school managed to negotiate the sand trap. but it was too deep for the vehicle to push through; it finished second overall, losing by 15 feet.

Just to prove the vehicle was able to complete the course, we asked permission from the judges to let us set it beyond the trap. The competition was over so they agreed. Starting just as it would have come through the sand it calmly completed the

course, easily staying between the dashed lines that claimed the winning school. We then turned the vehicle around and had it run the course backwards, a feat the judges declared more difficult than running it forwards; the car handled the challenge effortlessly, once again stopped only by the sand.

8.5 Algorithm Success:

The car traveled at 4.7 mph, or 6.9 feet/sec. It takes .29 seconds to process a single frame, so it travels about 2 feet per decision. Of that processing time, .25 seconds are spent waiting for the camera image to be loaded into memory; all the software processing is handled in the remaining .04 seconds. This was sufficient to handle 99% of the course; only when several quick adjustments in a row were needed did it fall short. A faster frame-grabber should overcome the other 1%.

As the authors understand, there were other teams moving at approximately 2 mph that handled around 10 frames per second; this translates into just 3 or 4 inches per decision. Furthermore they used sonar systems to give object information rather than relying on sight alone. Even so, these other vehicles frequently had to back up and try again to pass obstacles, closely resembling a human driver attempting to parallel park in a tight space. Not once did tulsa's Team B car back up; the move was deemed unnecessary for the competition, and so was never given as the action for any rule. However, the vision algorithm described in this paper hit only one obstacle and barely nicked two others, even with its faster motion, slower thinking, greater distance and tougher trials. From the vehicle's behavior in these instances it is obvious a quicker frame grabber would have enabled the vehicle to avoid the collisions.

It is the opinion of the authors that this algorithm and implementation have overwhelmingly demonstrated their effectiveness. The combined use of dynamic recalibration to adapt to changing conditions, fuzzy logic to identify colors, and neural networks to correlate any given image to a particular move met with triumphant success. This achievement is only the more astounding given the brief development time, along with the relative youth and complete inexperience of the team members.

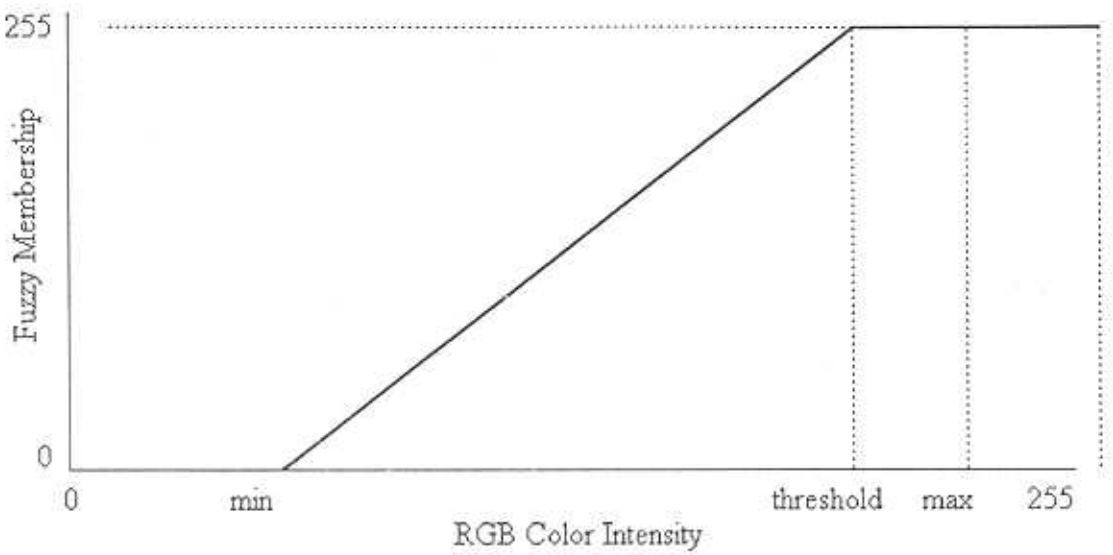
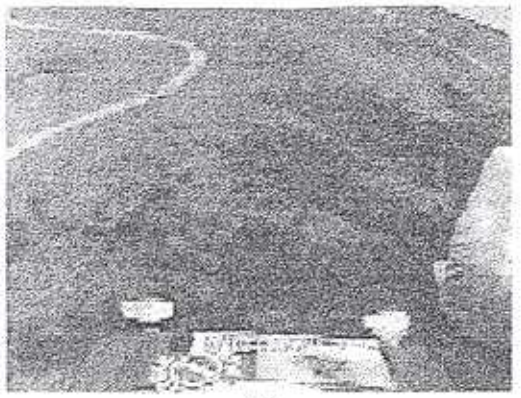
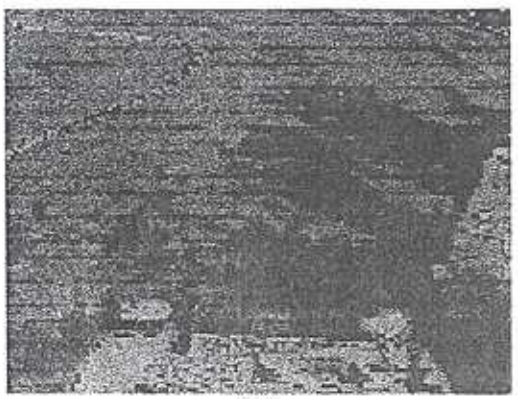


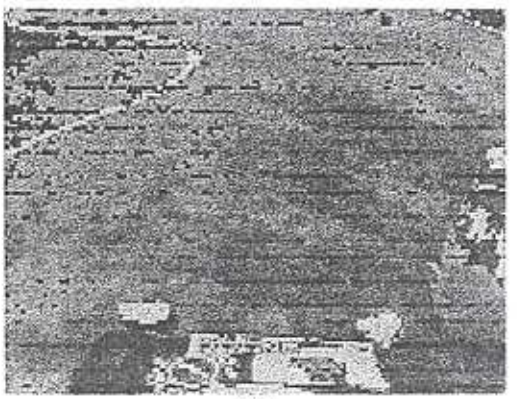
Figure . Basic fuzzy rule used in classifying belongedness of an input color



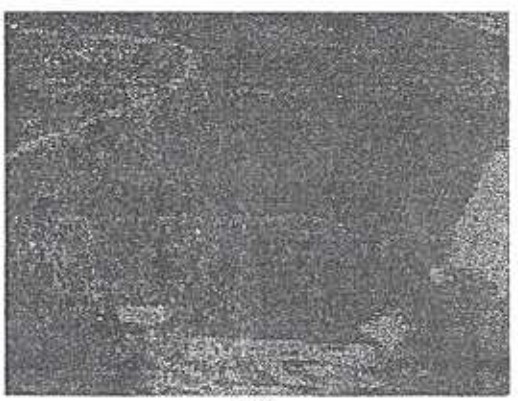
(1)



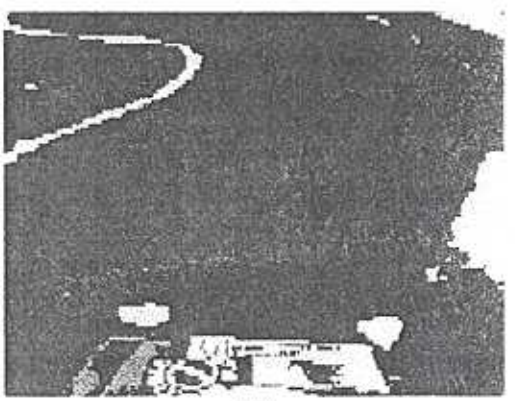
(2)



(3)



(4)



(5)

Figure 2. (1) unprocessed camera image; (2-4) red, green and blue fuzzy color filters; (5) color classification of each pixel in the picture.

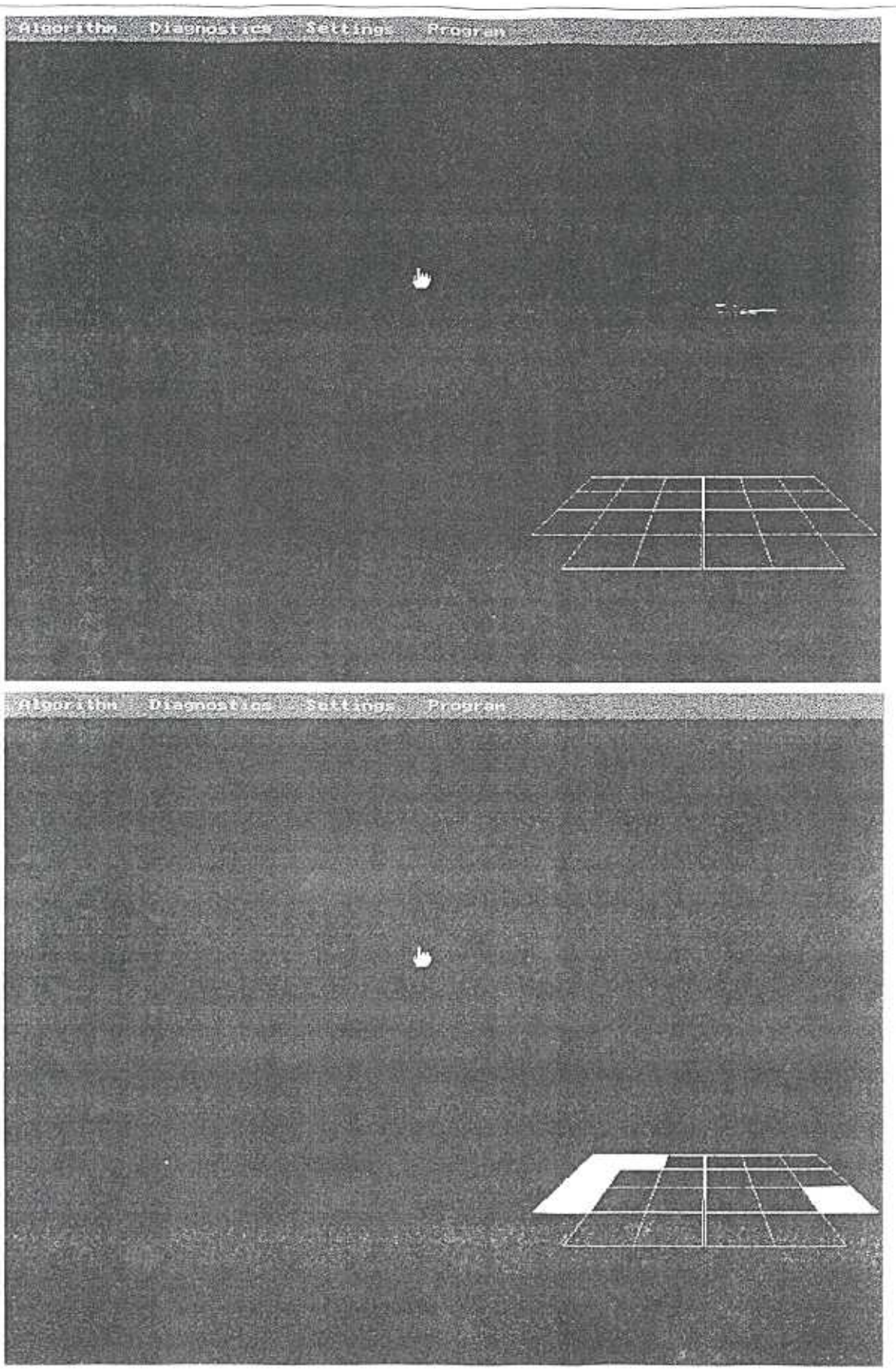
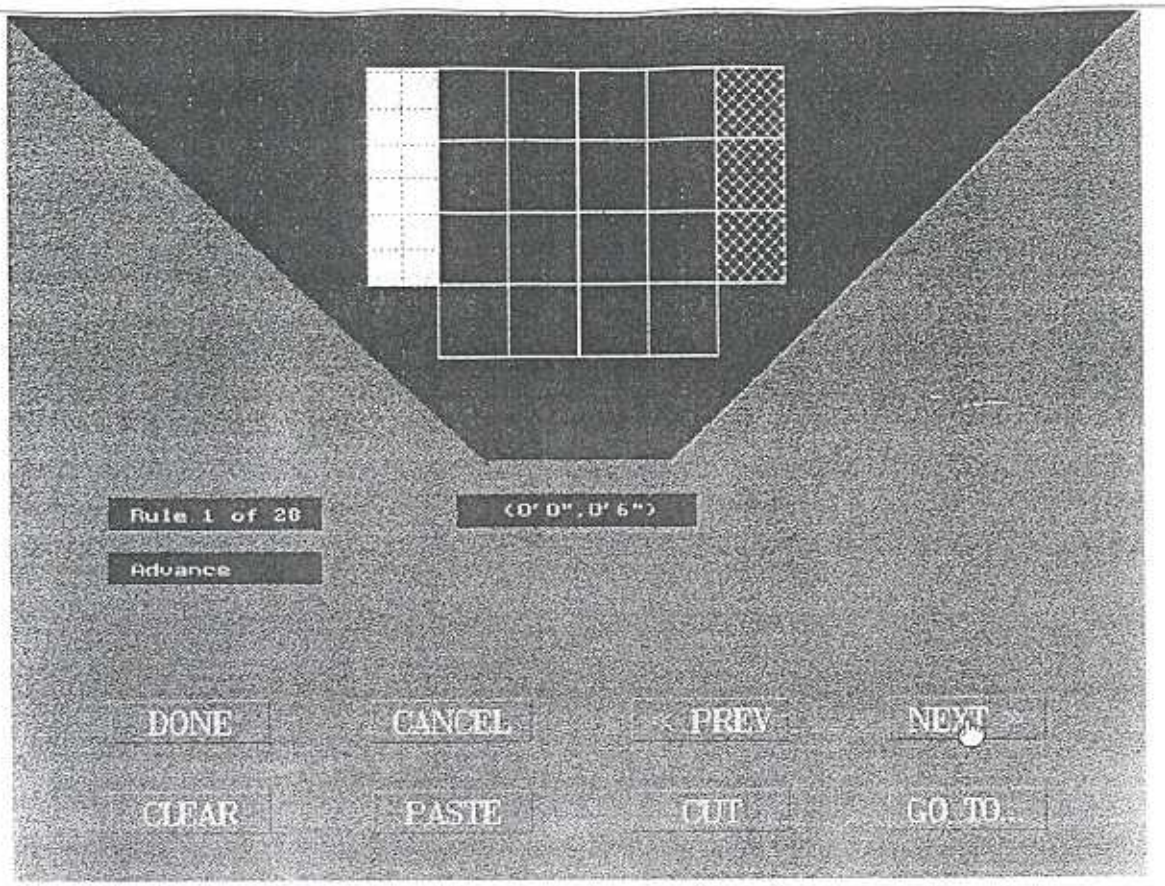


Figure 3 The zone structure converted to screen coordinates, and with blocked zones filled



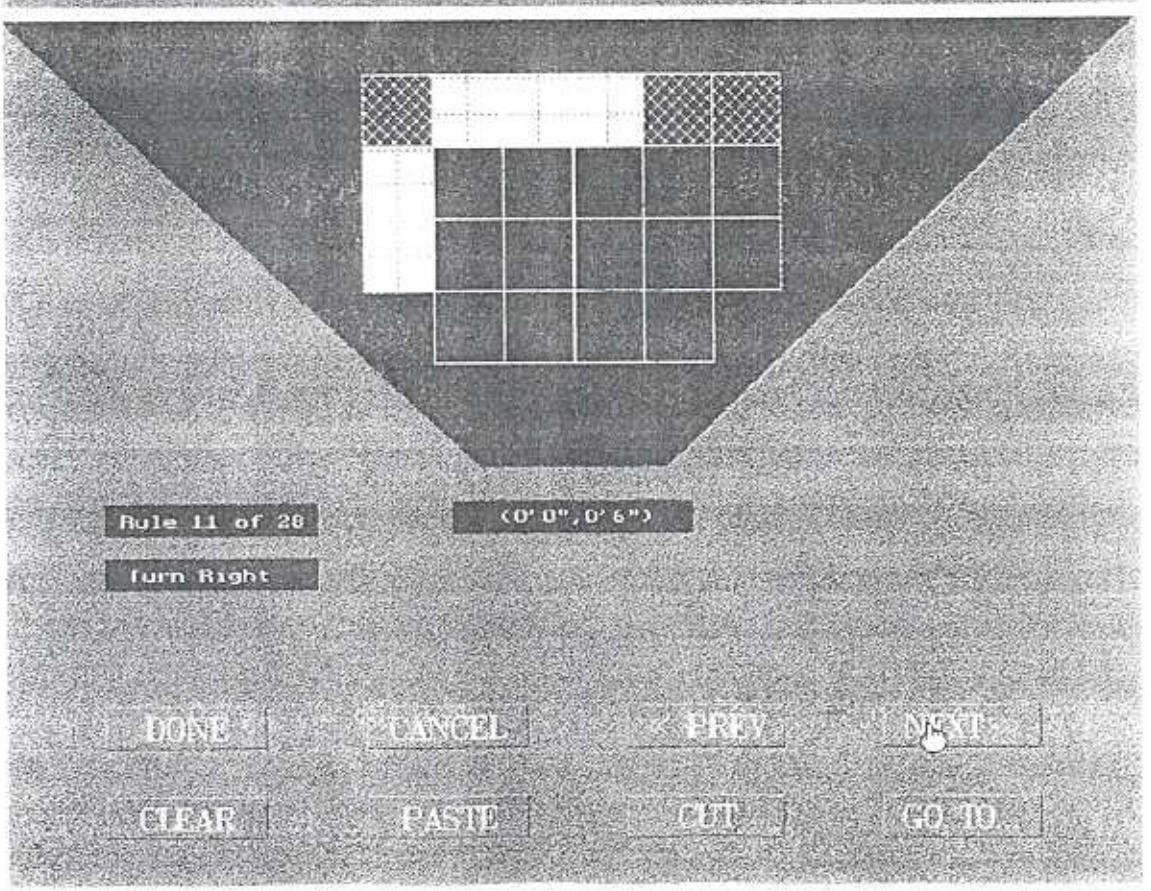


Figure 4 Two of the rules representative of the rule set The example uses these