Teaching a Robot How to Read Symbols

Paper 35
Autonomous Robots, Coordination of Multiple Activities, Lifelike Qualities, Real-Time Performance,
Knwoledge Acquisition and Management, Symbol Recognition

ABSTRACT

Symbols are used everywhere to help us find our way and they provide useful information about our world. Autonomous robots that would operate in real life settings could surely benefit from these indications. Research on character recognition have been going on for quite some time now, and demonstrations have been made of machines that can read printed and handwritten characters. To give an autonomous robot the ability to read symbols, we need to integrate character recognition techniques with methods to position the robot in front of the symbol, to capture the image that will be used in the identification process, and to validate the overall system on a robot. Our goal is not to develop new character recognition methods, but to address the different aspects required in making a mobile robotic platform, using current hardware and software capabilities, recognize symbols placed in real world environment. Validated on a Pioneer 2 robot, the approach described in this paper uses colors to detect symbols, a PID controller to position the camera, simple heuristics to select image regions that could contain symbols, and finally a neural system for symbol identification. Results in different lighting conditions are described, along with the use of our approach by our robot entry to the AAAI'2000 Mobile Robot Challenge, making the robot attend the National Conference on AI.

1. INTRODUCTION

The ability to read and recognize symbols is certainly a useful skill in our society. We use it extensively to communicate all kinds of information: exit signs, arrows to give directions, room numbers, name plates on our office doors, street names and road signs, to list just a few. In fact, even if maps are available to guide us toward a specific destination, we still need indications derived from signs to confirm our localization and the progress made. Car traveling illustrates well what we do: if we were to drive from Boston to Montréal, we would look at a map and get a general idea of what route to take. We are not going to measure the exact distance to travel on each road, and change direction based on tachymeter readings. Errors in measurements and readings will likely occur. Instead, we would rely on road signs to give us cues and indications on our progress toward our destination, in accordance with the map.

We believe the same to be true for mobile robots. Work on mobile robot localization using maps and sensor data (sonars and laser range finders for instance) [14, 13, 11] has been going on for quite some time now, with good results. However, to ease the task, such approaches could also exploit indications already present in the environment. One possibility is to extract visual landmarks from the environment [10]. Another possibility it to make the robot recognize symbols in the environment, which is the topic of this paper. Making a robot recognize symbols is an interesting idea because it can be a method shared by different types of robots, as long as they have a vision system. The information is also accessible by humans, which is not possible when electronic communication media are used by robots.

The idea of making machines read is not new, and research has been going on for close to four decades [12]. For instance, in 1958, Frank Rosenblatt demonstrated his Mark I Perceptron neurocomputer, capable of character recognition [5]. More recently, various commercial products capable of handwritten recognition are also on the market. So, making robots recognize symbols is surely a feasible project, and to our knowledge this is a new capability that has not yet been addressed for the design of autonomous robotic agents. But for a robot, symbol recognition is not the only step required. The robot has to detect the presence of a potential symbol, and to position itself to get an image sufficiently clear of the symbol to be recognized. In the following sections, the paper presents the specifications of our approach, the mechanisms implemented for perceiving a symbol, for positioning the robot in front of it and for identifying the symbol. Results in different lighting conditions are described. The paper also presents the use of our approach by our robot entry to the AAAI'2000 Mobile Robot Challenge, making the robot attend the National Conference on AI.

2. DESIGN SPECIFICATIONS

For this project, our goal is to address the different aspects required in making an autonomous robot recognize symbols placed in real world environments. Our objective is not to develop new character recognition algorithms or specific hardware for doing that. Instead, we want to integrate the appropriate techniques to demonstrate that such capability can be implemented on a mobile robotic platform,

Autonomous Agents '01 Montréal, Québec

Figure 1: Pioneer 2 AT robot in front of a symbol.

using their current hardware and software capabilities, and by taking into consideration real life constraints.

The robot used in this project is a Pioneer 2 robot, as the one shown in Figure 1. The robot is equipped with 16 sonars, a compass, a gripper, a pan-tilt-zoom (PTZ) camera with a frame grabber, a RF Ethernet-modem connection and a Pentium 233 MHz PC-104 onboard computer. The programming environment is Ayllu [17], a development tool for multi-agent behavior-based control systems. The PTZ camera is a Sony EVI-D30 with 12X optical zoom, high speed auto-focus lens and a wide angle len, pan range of ± 90 degrees (at a maximum speed of 80 degrees/sec), and a tilt range of ± 30 degrees (at a maximum speed of 50 degrees/sec). The camera also uses auto exposure and advanced backlight compensation systems to ensure that the subject remains bright even in harsh backlight conditions. This means that only by zooming on an object from the same position, brightness of the image is automatically adjusted. The frame grabber is a PXC200 Color Frame Grabber from Imagenation, which provides 320×240 images at a maximum rate of 30 frames per second. However, commands and data exchanged between the onboard computer and the robot controller are set at 10 Hz. Note that all processing for controlling the robot and recognizing symbols is done on the Pentium 233 MHz computer, so the decision processes of the robot must be optimized as much as possible.

To accomplish the goal stated previously, we also made the following assumptions:

- The approach is designed to recognize one symbol at a time, with each symbol made of one segment. Symbols made of multiple segments are not considered.
- Each symbol is placed parallel to the ground, on flat surfaces, as shown in Figure 1.

Our symbol recognition technique is done in three steps:

1. **Symbol perception.** To be able to do this in real time, our approach assumes that a symbol can be detected based on color.
2. **Positioning and image capture.** A behavior-based [1] approach is used for positioning the robot and controlling the PTZ camera, and for capturing an image of the symbol to recognize with sufficient resolution.
3. **Symbol identification.** A neural network approach is used to recognize a symbol in an image.

Necessarily, different mechanisms could be used to accomplish these steps. But since our goal is to demonstrate the feasibility of recognizing symbols on an autonomous robot, our approach uses simple mechanisms for each step and see how they perform. These mechanisms are described in the next sections.

3. SYMBOL PERCEPTION

A popular way to recognize objects in the world using a vision system on a mobile robot is to do color-based region segmentation, having objects perceived from their color [8, 15]. The best example is for object tracking and localization in the RoboCup soccer tournament [16]. The main reason is that such processing can be done in real time with common hardware and software. For the same reason, we chose to use color to detect the presence of a symbol in the world.

For color segmentation, first a color space must be selected from the one available by the hardware used for image capture. Colors can be represented in different spaces or formats: RGB, YUV, HSV, etc. Each of them have their advantages and drawbacks. HSV color representation is much more intuitive and is often used by color pickers in painting programs. However, most of the capture card do not use this color format and converting HSV format to RGB or YUV requires too much calculations which are difficult to do in real time. In our case, the robot has a BT848 capture board that can only support RGB and YUV color formats. Using rules that combines the intervals of R, G and B (or Y, U and V) values that belong to a specific color is an inefficient method because it requires six conditions to evaluate for evaluating the color class of each pixel [3].

Bruce et al. [3] presents a good summary of the different approaches for doing color segmentation on mobile robotic platforms, and describes an algorithm in YUV color format stores color membership values (trained or manually selected) into three lookup tables (one for Y, U and V respectively). The lookup values are indexed by their Y, U and V components. With Y, U and V encoded using 8 bits each, the approach uses three lookup tables of 255 entries. Each entry of the table is an unsigned integer, where each bit position corresponds to a specific color. With unsigned integers of 32 bits long, membership values (1 or 0) of up to 32 different colors can be stored. For the particular Y, U and V values of a pixel, a color is recognized if the membership values stored in the tables are all set to 1. In other words, membership of all 32 colors is done with three lookups and two logical AND operations, which is very efficient. Full segmentation is accomplished using 8 connected neighbors and grouping pixels that corresponds to a color into blobs.

In our system, we use a similar approach, but instead of using the YUV color representation, we use the RGB format for two reasons. First, the YUV representation available from our BT848 capture board is YUV 4:2:2 packed and requires a bit more computations to reconstruct pixel values since U and V are sampled at every 2 pixels and Y at each one. second, our capture board can give pixel values in RGB15 format,i.e., 0RRRRRGGGGGBBBBB, 5 bits for each of the R, G, B components. This makes it possible to generate one lookup table of 2^{15} entries (or 32768 entries, which is a reasonable lookup size). Only one lookup is required instead of three, but it uses more memory. The RGB components are encoded using less bits than YUV, but we assume that such precision may not be necessary for our

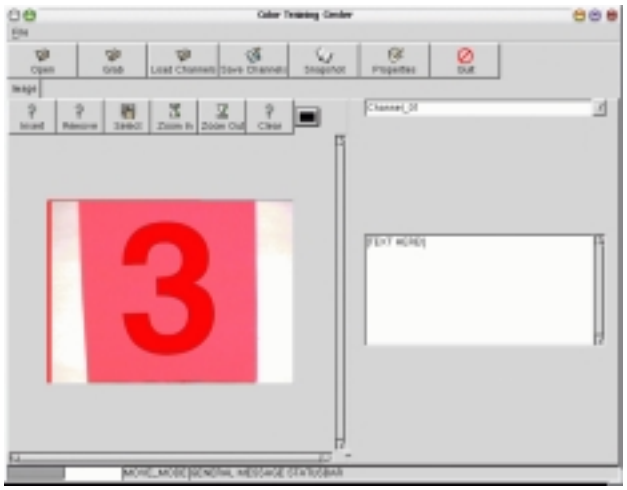

Figure 2: Graphical interface for manual color training.

task.

Another advantage of YUV over RGB is that the chrominance values (UV) are separated from the intensity component, making the representation more robust to light intensity. Rectangular regions to delimitate color can then be used. With RGB, special care must be used to set these regions with different light intensities. We use two methods to set the colors to be recognized:

- GUI interface. The GUI we have designed is shown in Figure 2. On the right part of the interface the color channel number (from 1 to 32) is selected. An image used for color training can be grabbed from the camera or loaded from a file. The user manually select pixels on the image corresponding to the desired color, or can also remove colors detected in particular regions. Zooming capabilities are also provided to facilitate the selection of regions in the image. Finally, once trained, it is possible to save the color channel in a file that the robot vision system can use for color segmentation.

- Color training using HSV representation that are converted into the RGB format and inserted in the lookup table. The idea is to start with a initial thresholds derived from a more comprehensive representation of colors. Using previously trained colors from thresholds in the HSV color format prevents the user from missing colors that are not present in the images selected for training a specific channel. If other colors are still missing for a particular channel, the missing colors can be added manually using the GUI interface. That way, the combination of these two methods increases the reliability of the color segmentation in different lighting conditions.

Using such algorithm for color segmentation, symbol perception is done by looking for black blob completely surrounded by an orange background. If more than one black blob surrounded by an orange blob are found in the image, the biggest black blob is used for symbol identification. Width/height proportion and density of the blob can also be used to determine which blob is most likely a symbol.

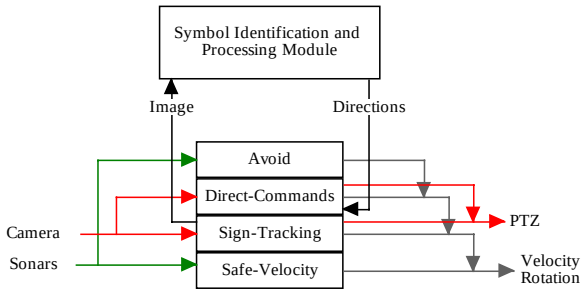

Figure 3: Behaviors used to control the robot.

Although, they are not used in the experiments described in this paper.

As indicated in Section 2, each recognizable symbol is assumed to be contained in one segment, i.e., all pixels of the same color representing the symbol must be connected (i.e., by 8 neighbors) together to avoid recombination of boundary boxes.

4. POSITIONING AND IMAGE CAPTURE

The idea is to have a robot move in the world, perceive a potential symbol and position itself in front of it to get a good image that could be used to identify the symbol. The behavior-based approach used to do this is shown in Figure 3. It consists of four behaviors arbitrated using Subsumption [2] to control the velocity and the rotation of the robot, and also to generate the pan-tilt-zoom commands to the camera. This approach allows the addition of other behaviors for controlling the robot and its camera in different situations and tasks. These behaviors are described below:

- *Safe-Velocity* makes the robot move forward without colliding with an object.

- *Sign-Tracking* tracks a symbol of a specific color (black in our case) surrounded by another color (orange). The camera link represents an image plus the pan, tilt and zoom positions of the camera.

- *Direct-Commands* change the position of the robot according to specific commands generated by the *Symbol Recognition and Processing Module*, described in Section 5.

- *Avoid*, the behavior with the highest priority, moves the robot away from nearby obstacles based on front sonar readings.

The *Sign-Tracking* behavior is the key behavior for our symbol recognition approach, and deserves a more complete description. When a black blob on an orange blob is detected, this behavior makes the robot stop. The behavior then tries to center the black blob in the image matching the center of area of the blob with the center of the image. The algorithm works in three steps. First, since the goal is to position the symbol in the center of the image, the x, y coordinates of the center of the black blob is represented in relation to the center of the image. Second, the algorithm must determine the distance in pixels to move the camera to center the black blob in the image. This distance must be carefully interpreted since the real distance vary

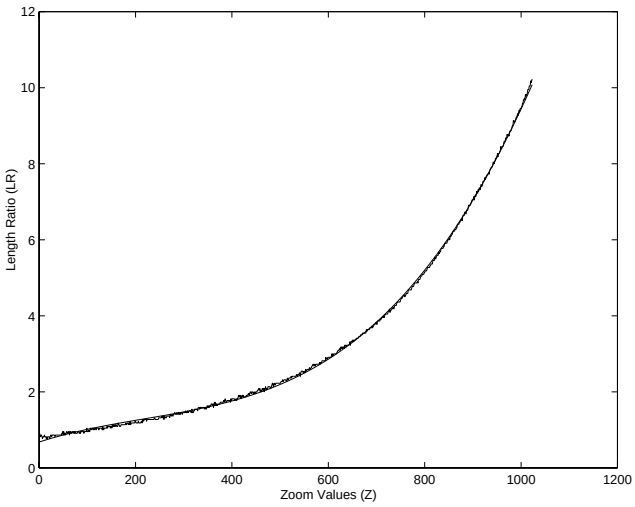

Figure 4: Scaling ratio according to zoom values.

with current zoom position. Intuitively, smaller pan and tilt commands must be sent when the zoom is high because the image represents a bigger version of the real world. To evaluate this influence, we put an object in front of the robot, with the camera detecting the object in the center of the image, with a zoom value of 0. We measured the length in pixels of the object, and took such readings with different zoom values (from minimum to maximum range). Considering that the length of the object at zoom 0 as the reference, we then calculated the length ratios at different zoom values. The result is shown in Figure 4. We then found a equation that fits these lengths ratios, as expressed in Equation 1. For a zoom position Z, the x, y values are divided by the corresponding LR to get the real distance $\tilde{x}, \tilde{y}$ in pixels of the symbol to the center of the image.

$$LR = 0.68 + 0.0041 \cdot Z + 8.94 \times 10^{-6} \cdot Z^2 + 1.36 \times 10^{-8} \cdot Z^3 \quad (1)$$

Third, pan-tilt-zoom commands must be determined to position the symbol at the center of the image. For pan and tilt commands (precise to a 10th of a degree), a PID (Proportional Integral Derivative [9]) controller is used, given in Equation 2. We chose a PID controller because it does not require a very precise model of the Sony EVI-D30 camera. The proportional part can also be used alone with no integral or derivative part. However, the proportional and the derivative components are very useful since it enables the camera to predict movement of the object being tracked. In other words, the robot can center objects moving at a constant speed in the image quite easily, which is very useful. The PID parameters are set to optimize the displacement of the camera by maximizing the movement and minimizing overshoot and stabilization time.

$$Pan = 0.3 \times \tilde{x} + 0.01 \int_t^{t+1000} \tilde{x} dt + 0.0075 \frac{d\tilde{x}}{dt}$$

Figure 5: First image captured by the robot in front the charging symbol.

$$Tilt = 0.3 \times \tilde{y} + 0.01 \int_t^{t+1000} \tilde{y} dt + 0.0075 \frac{d\tilde{y}}{dt} \quad (2)$$

For the zoom command, the minimal distance in pixels between the black blob and the edge of the orange blob, z, is used with the following heuristic:

(1) IF $|\tilde{x}| < 30$ AND $|\tilde{y}| < 30$
(2) IF $z > 30$ $Zoom+ = 25/LR$
(3) ELSE IF $z < 10$ $Zoom- = 25/LR$
(4) ELSE $Zoom- = 25/LR$

Rule (1) implies that the black blob is close of being at the center of the image. Rule (2) indicates to increase the zoom of the camera when the distance between the black blob and the edge of the orange blob is still too big, while rule (3) decreases the zoom if it is too small. Rule (4) decreases the zoom when the black blob is not centered in the image, to make it possible to see more clearly the symbol and facilitate its centering in the image. The division by the LR factor allows slower zoom variation when the zoom is high, and higher when the zoom is low. Note that one difficulty with the camera is caused by the auto exposure and advanced backlight compensation systems of the Sony EVI-D30 camera. By changing the position of the camera, the colors detected may vary slightly, and our approach must take that factor into consideration. The zoom is adjusted until stabilization of the pan-tilt-zoom controls is observed over a period of 5 processing cycles. The image with maximum resolution of the symbol is then obtained, with the symbol properly centered and scaled, the image is sent to the *Symbol Identification and Processing Module*. Figure 5 is a typical image captured by the robot when it first see a symbol, and Figure 6 is the optimal image obtained to maximize resolution of the symbol detected.

5. SYMBOL IDENTIFICATION

For symbol identification, we decided to use standard back-propagation neural networks because they can be easily used for simple character recognition, with good performance even

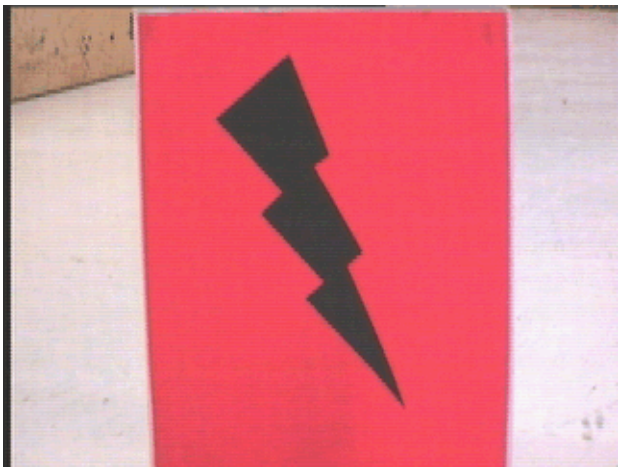

Figure 6: Optimal image captured by the robot in front the charging symbol.

with noisy inputs [4]. The first step is to take the part of the 320×240 image delimitated by the black blob inside the orange blob previously selected, and scale the black blob down to a 13×9 matrix. Each element of the matrix corresponds to -1 and 1, which is the absence or the presence of a black pixel symbol pattern. This image is then given as input to a neural network, with each element of the matrix associated with an input neuron.

To design our neural network, we started by generating data sets for training and testing. Since the symbol recognition ability was required to accomplish specific experiments under preparation in our laboratory [7], we selected only the useful symbols for other related research projects. These symbols are: numbers from 0 to 9, the first letters of the names of our robots (H, C, J, V, L, A), the four cardinal points (N, E, S, W), front, right, bottom and left arrows, and the charging station signs for a total of 25 symbols. The data sets were constructed by letting the robot move around in an enclosed area with the same symbol placed in different locations, and by memorizing the images captured using the optimal strategy described in Section 4. Fifteen images for each of the symbols were constructed this way. We also manually placed symbols at different angles with the robot immobilized to get a more complete set of possible angles of vision for each symbol, adding 35 new images for each symbol. Then, of the 50 images for each symbol, 35 images were randomly picked for the training set, and 15 images left were used for the testing set. Note that no correction to compensate for any rotation (skew) of the character is made by the algorithm. However, images in the training set sometimes contain small angles depending on the angle of view of the camera in relation to the perceived symbol.

Training of the neural networks was done using delta-bar-delta [6], which adapts the learning rate of the backpropagation learning law. The activation function used is the hyperbolic tangent, with activation values between -1 and +1. As indicated previously, the input layer of these neural networks is made of 117 neurons, one for each element of the 9×13 matrix. This resolution was set empirically: we estimated that this was a sufficiently good resolution to identify a symbol in an image. Learning was done off-line, using three network configurations:

1. One neural network for each symbol. Each network has no hidden neurons, and one output neuron. It was trained to recognize a specific symbol, and not to recognize other ones.

2. One neural network for all of the symbols, with different numbers of hidden neurons and 25 output neurons, one for each symbol.

3. Three neural networks that are able to recognize all of the symbols, with different number of hidden neurons (5, 6 and 7 respectively). A majority vote (2 out of 3) determines that the symbol is correctly recognized or not.

Table 1 summarizes the performances observed with these configurations. For configuration #2, the number of hidden units used is given in parenthesis. A symbol is considered correctly recognized when the output neuron activation is greater than 0.80. The column "Unrecognized" refers to a symbol without any identification given by the neural system, and the column "Incorrect" counts the symbol identified as another one by the neural system. Neural networks in configuration #2 with less than 5 hidden neurons can recognize less than 37 % of the training sets, and so are not suitable for the task. With more than 5 hidden units performances are getting better with minimal number of weights (and necessarily less processing power), but performance can be improved. The voting mechanism in configuration #3 gives better performances than for individual neural networks with 5, 6 and 7 hidden units, but the overall number of weights is still high. The best overall performance is observed using 15 hidden neurons in configuration #2, with all the training and testing sets recognized. Configuration #1 also gives good performances, but has the highest number of weights. So we chose to use the configuration #2 with 15 hidden neurons.

Once the symbol identified, different things can be done according to the meaning associated with the symbol. For instance, the symbol can be processed by a map planning algorithm to confirm localization, to associate a symbol with a particular place in the map, or decide where to go based on this symbol. In a simple scheme, the mechanism could be that once a symbol is identified, a command can be sent to the *Direct-Commands* behavior to make the robot move away from the symbol, and not continuously perceive the symbol. The position of the symbol relative to the robot can be derived from the pan-tilt-zoom coordinates of the camera.

6. EXPERIMENTS

Two sets of experiments are reported in this paper. First, results of tests done in controlled lighting conditions are presented. Second, experiments done in real environment are reported, and particularly during our participation at the AAAI'2000 Mobile Robot Challenge. The symbols used in these tests are printed on 8.5×11 inches orange sheets.

6.1 Tests in Controlled Conditions

The objective of these tests is to characterize the performance of the proposed approach in positioning the robot in front of a symbol and in recognizing symbols in different lighting conditions. Two sets of tests were conducted. First, we placed a symbol at various distances in front of

Table 1: Neural Network Configurations and Performances

Config. #	Training %	Testing %	% Unrecognized	% Incorrect	Number of Weights
1	100.00	98.40	0.00	0.36	2950
2 (5)	98.86	92.00	0.98	1.69	740
2 (6)	99.31	93.20	0.36	1.69	883
2 (7)	97.94	94.00	1.16	1.78	1026
2 (9)	100.00	95.20	0.89	0.18	1312
2 (10)	100.00	98.80	0.09	0.18	1455
2 (15)	100.00	100.00	0.00	0.00	2170
3	99.54	96.40	0.00	1.16	2649

Table 2: Average capture time at different distances

Distance (feet)	Time (seconds)
2	10.0
3	17.3
4	21.33
5	21.33
6	21.5
7	24.0
8	22.3
9	27.7
10	43.5

the robot, and we measured the time required to capture the image with maximum resolution of the symbol to identify using the heuristics described in Section 4. Results are summarized in Table 2. The time to capture the image varies between 0 and 45 seconds, depending on the proximity of the symbols and in good lighting conditions. When the symbol is farther away from the robot, more positioning commands for the camera are required, which necessarily takes more time. For distances of more than 10 feet, symbol recognition with the size of symbols used and the 9×13 image resolution for the neural networks is not possible.

The second set of tests consisted in placing the robot in an enclosed area where many symbols different background colors (orange, blue and pink) were placed on the ground at random positions. Letting the robot move freely for around half an hour in the pen for each of the background color, the robot tried to identify as many symbols as possible. Recognition rates were evaluated manually from HTML reports generated for each test. These reports contain all the images captured by the robot along with the identification of the recognized symbols. Symbols unrecognized, i.e., all of the outputs of the neural system have an activation value less than 0.8. Table 3 presents the recognition rates according to the background color of the symbols and three lighting conditions. The standard lighting condition is the fluorescent illumination of our lab. The low light condition is generated by spotlights embedded in the ceiling. Results show that recognition rate vary slightly with the background color. The way we positioned symbols in the environment may be responsible for that, since the symbols were not placed at the same position for every experiment, and light conditions sometimes gives us unwanted light reflection on the symbols. The unrecognized symbols were most of the time due to the robot not being well positioned in front of the symbols. In other words, the angle of view was too big and caused too much distortion on the symbols. Since the black blob of the symbols does not completely absorb white light, reflections may segment the symbol into two or more components. In that case, the positioning algorithm uses the biggest black blob that only represents part of the symbol, which is either unrecognized or incorrectly recognized as another symbol.

Figure 7: Lolitta H, our Pioneer 2 robot that participated in the AAAI'2000 Mobile Robot Challenge. The robot is shown next to the charging station symbol, and is docked for recharge.

6.2 AAAI'2000 Mobile Robot Challenge

The Challenge provided a good setup to see how the ability to recognize symbols can benefit a robot operating in real life settings. The AAAI'2000 Mobile Robot Challenge is to make a robot attend the National Conference on AI. Our goal was to design an autonomous robot capable of going through a simplified version of the entire task from start-to-end, by having the robot interpret printed symbols to get useful information from the environment, interact with people using visual cues (like badge color and skin tone) and using a touch screen, memorize information along the way in a HTML report, and recharge itself when necessary [7]. Figure 7 shows a picture of our robot entry to the AAAI'2000 Mobile Robot Challenge.

At the conference, we made several successful tests in the exhibition hall, in a constrained area and with constant illumination condition. We also ran two complete trials in the convention center, with people and in the actual setting

Table 3: Recognition Performances in Different Lighting Conditions

Background color	Light condition	% Successfully recognized	% Not recognized	% Wrongly recognized
Orange	Std	81.1	13.5	5.4
Orange	Low	88.2	11.8	0.0
Blue	Std	95.6	4.4	0.0
Blue	Low	100.0	0.0	0.0
Pink	Std	91.2	8.8	0.0
Pink	Low	91.7	8.3	0.0

for the registration, the elevator and the conference rooms. During these two trials, the robot was able to identify symbols correctly in real life settings. Using configuration #3, identification performance was around 83 % (i.e., 17 % of the images used for symbol identification were identified as unrecognizable), with no symbol incorrectly identified. The symbols used for the challenge are the arrow signs, the charging symbol, the letters **L**, **H** and **E**, and the numbers **1**, **2** and **3**. Symbol identification was done depending on the different phases of the challenge (i.e., finding the registration desk, taking the elevator, schmoozing, guarding, going to the conference room and presenting). For instance, while the robot schmooze, no symbol identification were allowed. Also, depending on the symbol identified, the robot could do different things to position itself in regard to the symbol. For example, when the charging symbol is detected, the robot go to a particular position in front of the symbol, and makes a turn of 180 degrees to detect the charging station using its infrared ring. This way, the responses the robot made to a symbol identified was done according to its current state and intentions.

In these trials, the most difficult part was to adjust color segmentation for the orange and the black for different illumination conditions in the convention center: some areas were illuminated by the sun, while others (like the entrance of the elevator) were very dark. Even though the superposition of two color regions for the localization of a symbol gave more robustness to the approach, it was still difficult to find the appropriate color segmentation that worked in so diverse illumination conditions. So in some places like close to the elevator, we slightly change the vertical angle of the symbol to get more illumination. At that time, we only used the manual training method to train colors. Better results was obtained after the conference using the HSV representation. Overall, we found that having the robot interpret printed symbols to get useful information from the environment greatly contributed to its autonomy.

7. DISCUSSIONS

Using simple methods for color segmentation, robot positioning and image capture, and symbol recognition using neural networks, we have demonstrated the feasibility of making a mobile robot reads symbols and derives useful information that can affect its decision making process. But the approach can still be improved, and here is a list of limitations of our approach and potential ameliorations:

- Positioning and image capture is the process that takes the most time in our symbol recognition approach. At a distance of 10 feet, it can take around 45 seconds to get an image with maximum resolution. When the robot is moving, this does not happen frequently because by the time the robot detects the symbol and stops, it has move closer to the symbol. With the robot moving, it usualy takes around 20 seconds. However, simple solutions can be implemented to improve that. We are currently experimenting with a method that allows to successively send images of a symbol at different zoom value until a valid identification is obtained.

- Increasing the image resolution used by the neural network system would improve symbol recognition at a greater perceptual range.

- Other representations of the symbols could be generated as inputs to the neural network system for symbol recognition. For instance, features that are more rotation independent for the symbols could be extracted to increase the robustness of the identification process.

- The angle of the robot in relation to the symbol could be evaluated from the direction of the edges of the orange blob. This could help position the robot right in front of the symbol. Using our simple approach sometimes cause the robot to lose track of the symbol when the robot is moving past a symbol and that the camera is not fast enough to continue tracking the symbol before the robot stops.

- The maximum resolution of a symbol depends on the position of the center of area of the black blob in the image. We reach the maximum resolution when the distance between the center of area of the black blob and the center of the image is bigger than the distance between the black blob and the border of the image. So, the symbol resolution varies with the shape of the symbol.

- Sometimes, color blobs of the same colors are separated by only few pixels that corresponds to noise or small objects in the foreground of the bigger ones. Merging those blobs together, when they are similar in shape or density, could improve greatly the tracking ability of the robot since our approach is based on a black blob completely surrounded by an orange one. In our experiments, loosing track of a symbol happens when the orange blob is segmented into two or more components.

- The PID controller can be optimized with the help of a better model of the camera. A non-linear controller can also be used to control the camera. The main idea is to send smaller pan and tilt commands when

the zoom gets bigger. Using a linguistic description of useful heuristics for doing that, we are exploring the use of a fuzzy controller.

8. CONCLUSIONS

This paper describes how we were able to integrate simple techniques for perceiving symbols by tracking a black blob over an orange region, positioning and capturing an image of the symbol using a behavior-based approach and a PID controller, and recognizing symbols with the help of a neural network system. The system works in real time on a Pioneer 2 robot, using no special hardware components, and so can be easily implemented on other robotic platforms that have a color vision system and an on-board computer. Results show that good recognition performances in various illumination conditions. Such capabilities can greatly benefit autonomous mobile robots that have to operate in real life settings, and one interesting objective is surely to develop appropriate algorithms that would allow symbol recognition on various backgrounds, with different shapes and part of a complete message. Robots would then have access to information that we commonly use to guide and inform us in our world.

9. ACKNOWLEDGMENTS

This research is supported financially by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canadian Foundation for Innovation (CFI) and the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) of Québec.

10. REFERENCES

[1] R. C. Arkin. *Behavior-Based Robotics*. The MIT Press, 1998.

[2] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[3] J. Bruce, T. Balch, and M. Veloso. Fast color image segmentation using commodity hardware. In *Workshop on Interactive Robotics and Entertainment*, 2000.

[4] H. Demuth and M. Beale. *Matlab's Neural Network Toolbox*. The Math Works Inc., 1994.

[5] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, 1989.

[6] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[7] F. Michaud, J. Audet, D. Létourneau, L. Lussier, C. Théberge-Turmel, and S. Caron. Autonomous robot that uses symbol recognition and artificial emotion to attend the aaai conference. In *Proc. AAAI Mobile Robot Workshop*, 2000.

[8] B. W. Minten, R. R. Murphy, J. Hyams, and M. Micire. A communication-free behavior for docking mobile robots. In L. Parker and G. B. andJ. Barhen, editors, *Distributed Autonomous Robotics Systems*, pages 357–367. Springer, 2000.

[9] K. Ogata. *Modern Control Engineering*. Prentice Hall, 1990.

[10] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *Proc. IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 1060–1065, 1998.

[11] R. Simmons, J. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan. Lessons learned from xavier. *IEEE Robotics and Automation Magazine*, 7(2):33–39, 2000.

[12] C. Suen, C. Tappert, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):341–348, 1990.

[13] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proc. IEEEE Intl Conf. on Robotics and Automation*, 2000.

[14] S. Thrun, D. Fox, and W. Burgard. A probabilistic approach toconcurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.

[15] I. Ulrich and I. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *Proceedings National Conference on Artificial Intelligence (AAAI)*, pages 866–871, 2000.

[16] M. Veloso, E. Winner, S. Lenser, J. Bruce, and T. Balch. Vision-servoed localization and behavior-based planning for an autonomous quadruped legged robot. In *Proceedings of AIPS-2000*, Breckenridge, 2000.

[17] B. B. Werger. Ayllu: Distributed port-arbitrated behavior-based control. In L. Parker and G. B. andJ. Barhen, editors, *Distributed Autonomous Robotic Systems*, pages 25–34. Springer, 2000.