Mobile Robot that can Read Symbols

François Michaud and Dominic Létourneau
LABORIUS - Research Laboratory on Mobile Robotics and Intelligent Systems
Department of Electrical and Computer Engineering
Université de Sherbrooke, Sherbrooke (Québec Canada) J1K 2R1
{michaudf,letd01,audj01}@gel.usherb.ca, http://www.gel.usherb.ca/laborius

Abstract

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 approach on a robot. Our goal is to address the different aspects required in making a mobile robotic platform recognize symbols placed in real world environment, using current hardware and software capabilities. 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 presented.

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, etc. 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 that well. Instead of only looking at a map and the vehicle's tachymeter, we rely on road signs to give us cues and indications on our progress toward our destination.

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) [9] 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 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 [8]. But for a robot, symbol recognition is not the only step required. It has to detect the presence of a potential symbol and to position itself to get an image sufficiently clear of the symbol to recognize. In the following sections, the paper presents the specifications of our approach, the mechanisms implemented for making a mobile robot recognize symbols, and results obtained in different lighting conditions.

2 Design Specifications

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

Shown in Figure 1, the robot used is a Pioneer 2 robot that has 16 sonars, a pan-tilt-zoom (PTZ) camera with a frame grabber and a Pentium 233 MHz PC-104 onboard computer. The PTZ camera is a Sony EVI-D30 with 12X optical zoom, high speed autofocus lens and a wide angle len, pan range of $\pm 90°$ (at a maximum speed of $80°$/sec), and a tilt range of $\pm 30°$

(at a maximum speed of 50°/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 when 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 in our design 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 onboard computer, so the decision processes of the robot must be optimized as much as possible.

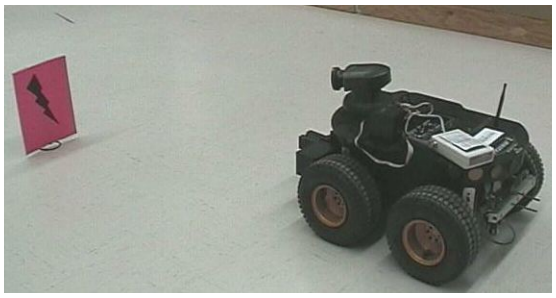

Figure 1: Pioneer 2 AT robot in front of a symbol.

To accomplish the goal stated previously, our 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. Also, each symbol is placed perpendicular to the floor on flat surfaces, as shown in Figure 1. The symbol recognition technique is done in three steps:

Symbol perception. To be able to do this in real time, our approach assumes that a symbol can be detected based on color.

Positioning and image capture. A behavior-based [2] 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.

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 in each step. These mechanisms are described in the following sections.

3 Symbol Perception

A popular way to recognize objects using a vision system on a mobile robot is to do color-based region segmentation, i.e., having objects perceived from their color. 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.

Bruce et al. [4] present a good summary of the different approaches for doing color segmentation on mobile robotic platforms, and describe an algorithm using the YUV color format and rectangular color threshold values stored 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 256 entries. Each entry of the tables is an unsigned integer, where each bit position corresponds to a specific color. With unsigned integers of 32 bits long, we can then store the color thresholds of 32 colors simultaneously. Instead of defining colors using boundary check for minimum and maximum for each color component and for each color, thresholds are stored in the lookup table as binary masks. More specifically, thresholds verification of all 32 colors for a specific Y,U and V values can be calculated 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 the same 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. It is then possible to generate only one lookup table of 2^{15} entries (or 32768 entries), which is a reasonable lookup size. In that case, the color bits in the unsigned integer value in the lookup table correspond to the presence (0 or 1) of the 32 designated colors. 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 task.

With RGB, special care must be taken because rectangular thresholds do not perform well under different lighting conditions. Using the lookup like a membership table, we are not constrained of using rectangular like thresholds since each combination of the R,G and B values corresponds to only one entry in the table. To help us build the membership lookup table, we used colors represented in HSV space that we translated back into RGB. Color adjustments were made by directly selecting pixels from an image grabbed by the camera or loaded from a file, using our own GUI interface. It is also possible to remove colors detected in particular regions, or to zoom in and out to facilitate the selection of regions in the image. Configurations for particular color can then be saved in a file and used by the robot.

Using such algorithm for color segmentation, symbol perception is done by looking for a 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. 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 objective 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 2. It consists of four behaviors arbitrated using Subsumption [3] 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); *Direct-Commands* changes the position of the robot according to specific commands generated by the *Symbol Identification and Processing Module* (described in Section 5; and *Avoid*, the behavior with the highest priority, moves the robot away from nearby obstacles based on front sonar readings. Note that the Image link represented in Figure 2 also includes the pan, tilt and zoom positions of the camera.

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

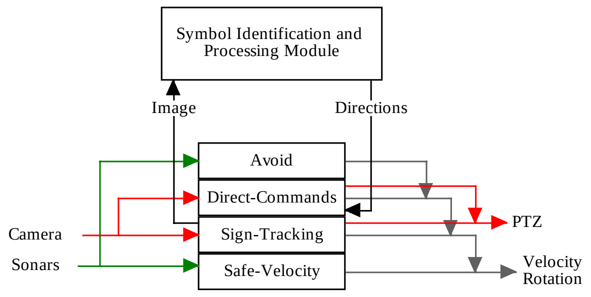

Figure 2: Behaviors used to control the robot.

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 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 at different zoom values (from minimum to maximum range). Considering as the reference the length of the object at zoom 0, we then calculated the length ratios at different zoom values. We then found a equation that fits these lengths ratios, expressed in Equation 1. For a zoom position Z, the x, y values are divided by the corresponding length ratio 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 [7]) controller is used, given in Equation 2 (the same relation holds for *Tilt* commands, using y values). We chose a PID controller because it does not require a very precise model of the Sony EVI-D30 camera. First, the proportional part was set alone with no integral or derivative part. The goal was to minimize overshoot. The other parameters of the PID were then added to optimize the displacement of the camera by maximizing the movement and minimizing overshoot and stabilization time, enabling the camera to predict movement of the object being tracked. The PID parameters were set empirically because no positional feedback is provided by the Sony camera, and small camera displacements are required to avoid loosing the symbol at various zoom values.

$$Pan = 0.3 \times \tilde{x} + 0.01 \int_{t}^{t+1000} \tilde{x} dt + 0.0075 \frac{d\tilde{x}}{dt} \qquad (2)$$

At constant zoom, the camera is able to position itself with the symbol at the center of the image in less than 10 cycles (i.e., 1 second). However, at the same time, the camera must increase its zoom to get an image with good resolution of the symbol to interpret. 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 $z > 30$ $Zoom = Zoom + 25/LR$
(3)     ELSE IF $z < 10$ $Zoom = Zoom - 25/LR$
(4) ELSE $Zoom = Zoom - 25/LR$
```

Rule (1) implies that the black blob is close of being at the center of the image. Rule (2) increases 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, and is sent to the *Symbol Identification and Processing Module*. Figure 3 shows an image with maximum resolution of the charging symbol perceived.

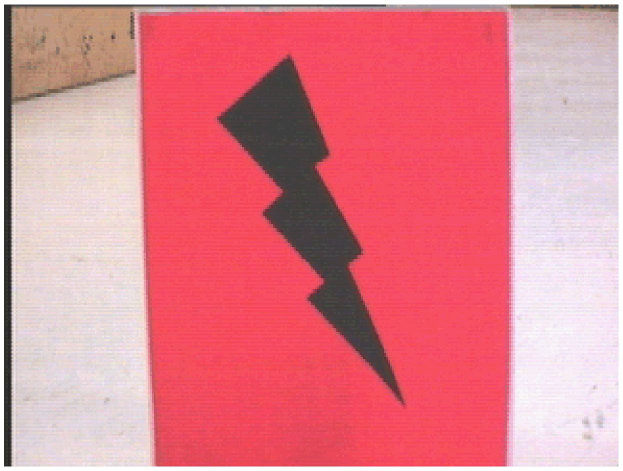
Figure 3: Image with maximum resolution captured by the robot in front the charging symbol.

5 Symbol Identification

For symbol identification, we decided to use standard backpropagation neural networks because they can be easily used for basic character recognition, with good performance even with noisy inputs. Our algorithm first takes the part of the 320×240 image delimitated by the black blob inside the orange blob previously selected, and scales the black blob down to a 13×9 matrix. Each element of the matrix corresponds to -1 and 1, which represents the absence or the presence of a black pixel. This image is the input presented 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 [6], 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 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 the 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 [5], 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 different network configurations such as one neural network for each symbol; one neural network for all of the symbols (i.e., with 25 output neurons), and three neural networks for all of the symbols, with different number of hidden neurons and using a majority vote (2 out of 3) to determine that the symbol is correctly recognized or not. The best performance is obtained with one neural network for all of the symbols, and 15 hidden neurons. With this configuration, all symbols in the training and the testing sets were recognized (a symbol is considered correctly recognized when the output neuron activation is greater than 0.80).

Once the symbol is identified, the predetermined or learned meaning associated with the symbol can be used to affect the robot's behavior. For instance, the symbol can be processed by a planning algorithm to change the robot's goal. In a simple scheme, 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. In relation to our work, Adorni et al. [1] uses symbols (surrounded by a shape) with a map to confirm localization. But their approach uses shapes to detect a symbol, black and white images and no zoom.

6 Experiments

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. Three sets of tests were conducted. First, we placed a symbol at various distances in front of the robot, and the time required to capture the image with maximum resolution of the symbol to identify using the heuristics described in Section 4. It takes between 10.0 sec (at two feet) to 27.7 sec (at nine feet) to capture the image used for symbol recognition. When the symbol is farther away from the robot, more positioning commands for the camera are required, which necessarily takes more time. 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 usually takes around 20 seconds. 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.

Table 1: Recognition Performances in Different Lighting Conditions

Background color	% Recognized	% Incorrect
Orange (std)	89.0	3.4
Orange (low)	94.1	2.6
Blue (std)	88.6	4.2
Blue (low)	94.7	2.1
Pink (std)	92.2	1.9
Pink (low)	95.2	1.0

The second set of tests consisted in placing the robot in an enclosed area where many symbols with different background colors (orange, blue and pink) were placed at specific positions. Two lighting conditions were used in these tests: standard (fluorescent illumination), and low (spotlights embedded in the ceiling). For each color and illumination conditions, 25 images of each of the 25 symbols were taken. Table 1 presents the recognition rates according to the background color of the symbols and the illumination conditions. Recognition rates were evaluated manually from HTML reports containing all of the images captured by the robot during a each test, along with the identification of the recognized symbols. A symbol is not recognized when all of the outputs of the neural system have an activation value less than 0.8. Results show that the average recognition performance is 92.3% with 2.5% incorrectly recognized, which is very good considering that the robot can encounter a symbol from any angle and at various distances. Recognition performances vary slightly with the background color. Putting symbols randomly in the enclosed area gave slightly lower performance, but the lowest was

81.1% for the orange in standard illumination condition. Incorrect recognition and symbol unrecognized were most of the time due to the robot not being well positioned in front of the symbols: the angle of view was too big and caused too much distortion. Since the black blob of the symbols does not completely absorb white light (the printed part of the symbol creates a shinning surface), 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. That is also why performances in low illumination conditions are better than in standard illumination, since reflections are minimized.

The third set of tests were done during the AAAI 2000 Mobile Robot Challenge, which is to make a robot attend the National Conference on AI. There were windows in various places in the convention center, and some areas had very low lighting (and so we sometimes had to slightly change the vertical angle of the symbols). Our entry was able to identify symbols correctly in such real life settings, with identification performance of around 83 %, 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.

7 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. We have demonstrated the feasibility of making a mobile robot reads symbols and derives useful information that can affect its decision making process. 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 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 recognition of symbols 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. Future improvements involve successively send images of a symbol at different zoom values to improve the identification rate, and increase the image resolution used by the neural network to improve symbol recognition at a greater perceptual range.

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.

References

[1] G. Adorni, G. Destri, M. Mordonini, and F. Zanichelli. Robot self-localization by means of vision. In *Proc. EUROBOT*, p. 160–165, 1996.

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

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

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

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

[6] 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.

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

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

[9] 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. IEEE Intl Conf. on Robotics and Automation*, 2000.