Making a Mobile Robot Read Textual Messages [*]

Dominic Létourneau, François Michaud, Jean-Marc Valin, Catherine Proulx
Department of Electrical Engineering and Computer Engineering
Université de Sherbrooke
Sherbrooke, Québec (Canada) J1K 2R1
{Dominic.Letourneau,Francois.Michaud}@USherbrooke.ca

Abstract – *With all of the textual indications, messages and signs we find in urban settings to provide us with all types of information, it is only natural that we try to give to robots reading capabilities of interpreting such information. Equipped with optical character recognition algorithms, a mobile robot has to face the challenge of controlling its position in the world and its Pan-Tilt-Zoom camera to find the textual message to capture, try to compensate for its viewpoint of the message, and use limited processing capabilities to decode the message. The robot also has to deal with non-uniform illumination and changing conditions in the world. In this work, we address the different aspects of the character recognition process to be incorporated into the higher level intelligence modules of a mobile robotic platform.*

Keywords: message tracking, image segmentation, character recognition, neural networks, camera control, mobile robot.

1 Introduction

Giving the ability to read textual messages is highly desirable for increased autonomy of mobile robots operating in the real world. Providing a map of the environment surely can help the robot localize itself in the world but like humans, even if we may use maps, we also exploit a lot of written signs and symbols to help us navigate in our cities, office buildings and so on. Just think about road signs, street names, room numbers, exit signs, arrows to give directions, etc. Messages already incorporated by humans in the environment could be of great value for the overall knowledge of the world if the mobile robot can interpret them. The process seems fairly simple: give a robot the ability to acquire an image of a message to read, extract the symbols and recognize them.

With all the research that has been going on in the area of optical character recognition [8], it is surely possible to make a mobile robot read symbols and signs. But contrarily to conventional character recognition system, a mobile robot has to find a textual message to capture, try to compensate for its viewpoint of the message, and use limited processing capabilities to decode the message. The robot also has to deal with non-uniform illumination and changing conditions in the world.

In this project, our goal is to address the different aspects required in making an autonomous robot recognize textual messages 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. The objective of our work is to provide a mobile robot with the capability of reading one symbol at a time, and extend this to strings of characters. Our approach integrates techniques for: 1) perceiving symbols using color segmentation, 2) positioning and capturing an image with sufficient resolution using behavior-producing modules and a PID controller for a Pan-Tilt-Zoom (PTZ) camera, 3) exploiting simple heuristics to select image regions that could contain symbols, and 4) recognizing symbols using a neural network.

Dulimarta and Jain [4] also address the problem of making a robot recognize messages. However, their approach is based on template matching of the messages to recognize (i.e., door number plate detection), with a non-moving camera fixed at a pre-determined height, while our approach aims at reading any messages at different positions in the environment written using the appropriate font. We outline the constraints under which the approach works, and present results obtained using a Pioneer 2 robot equipped with a Pentium 233 MHz and a Sony EVI-D30 PTZ camera.

2 Control architecture

In our previous work we demonstrated that a mobile robot is capable of reading symbols and deriving useful information that can affect its decision-making process [6]. Our approach allows to recognize one symbol at a time (i.e., by having only one symbol on a sheet of paper), with each symbol made of one segment (all connected pixels). Also, each symbol was placed perpendicular to the floor

[*] **0-7803-7952-7/03/$17.00 © 2003 IEEE.**

on flat surfaces, at the same height of the robot. Figure 1 shows typical symbols recognized using this approach.

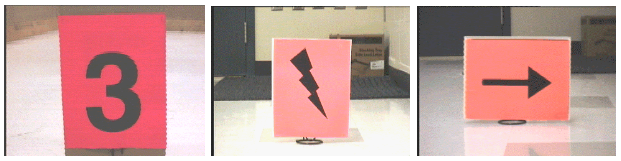

Figure 1. Typical symbols placed perpendicular to the ground

The symbol recognition technique was done in three steps:

1. Symbol perception: using an algorithm for color segmentation using a lookup table in the RGB color format, symbol perception was done by looking for a black blob completely surrounded by an colored (orange, pink or blue) background.

2. Positioning and image capture: behavior-producing modules were 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 back-propagation neural network with 15 hidden neurons was trained and used to recognize 25 symbols using Times font. The approach had a recognition performance of 91%, with 5.4% unrecognized symbol of and 3.6% of false recognition, under high and low illumination conditions. The resolution of the symbols was approximately 130 pixels.

The extended approach is basically the same as the one described previously, but with some modifications to the *Character-Tracking* behavior and the *Message Processing Module*. Generalizing those two modules allows us to interpret whole messages (words and sentences) printed on the same sheet of paper instead of a single symbol. To do so, we still assume that the messages will be composed of characters using a pre-specified font, printed black on a 8.5 x 11 sheet of paper (colored or white, specified as a parameter in the algorithm). We expanded the character set to 26 capital letters (A to Z) and 10 digits (0 to 9).

The approach consists of making the robot move autonomously in the world, stop when a potential message is detected based on color, acquire an image with sufficient resolution for identification, one character at a time starting from left to right and top to bottom. To do so, four behavior-producing modules arbitrated using Subsumption [1] are used. These behaviors control the velocity and the heading of the robot, and also generate the Pan-Tilt-Zoom (PTZ) commands to the camera. These behaviors implemented are: *Safe-Velocity* to make the robot move forward without colliding with an object (using sonars); *Character-Tracking* to track a message composed of black regions (characters) over a colored or white background; *Direct-Commands* changes the position of the robot according to specific commands generated by the *Message Processing Module*; and *Avoid*, the behavior with the highest priority, moves the robot away from nearby obstacles based on front sonar readings.

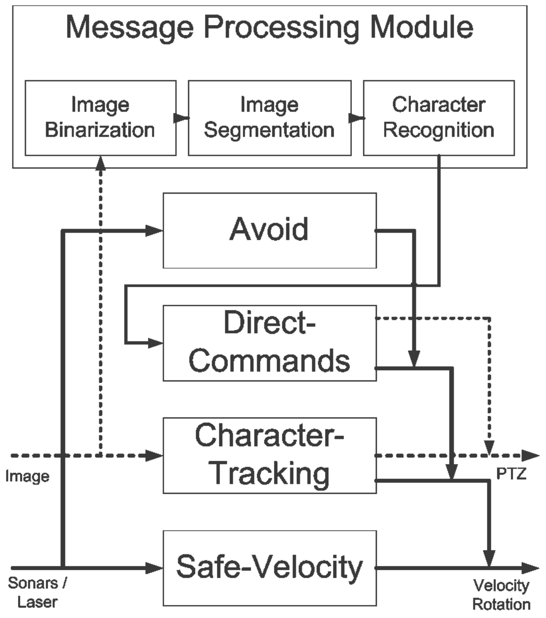

Figure 2. Camera and robot control architecture

The *Character-Tracking* behavior is an important element of the approach because it provides the appropriate Pan-Tilt-Zoom commands to get the maximum resolution of the message to identify. Using the RGB15 color format, our algorithm stores in a 2^{15} x 32 lookup table the membership of a pixel to one of 32 color channels. Using a lookup table instead of thresholds on each of the RGB components to define colors have several advantages:

1. Colors that would require multiple thresholds to define them in the RGB format (multiple cubic like volumes) are automatically stored in the lookup table.

2. Using a single lookup table is faster than using multiple *if-then* conditions for the thresholds.

3. No transformation is required to index the lookup table, the RGB15 pixel value from the captured image is used directly.

4. Membership to a color can be stored in a single bit (0 or 1). Using 32 bits values in the lookup table enables us to get the membership of 32 colors simultaneously in a single lookup.

Cubic thresholds in the HSV (Hue, Saturation, Value) color representation work well to represent colors [2]. At the color training phase, conversions from the HSV representation with standard thresholds to the RGB lookup table are easy to do. Figure 3 shows a representation of the black, blue, pink and orange colors in the RGB color space as it is stored in the lookup table. It also demonstrates that using a lookup table enables us to store color membership that would require multiple thresholds in the RGB color space.

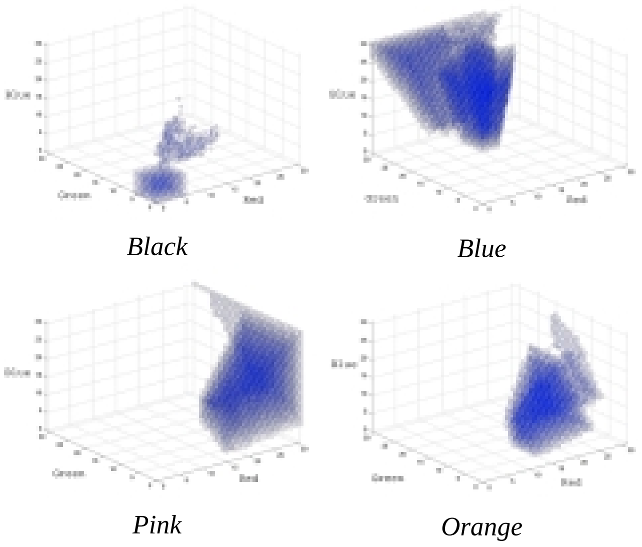

Figure 3 Color membership representation in the RGB color space

In order to facilitate the training of colors, a Graphical User Interface (GUI) has been created in the RobotFlow[1] programming environment. The interface is shown in Figure 4. The GUI provides an easy way to select colors directly on the source image for a desired color channel and stores the selected membership pixel values in the color lookup table. The GUI also provides an easy way to visualize the color perception of the robot for all the trained color channels.

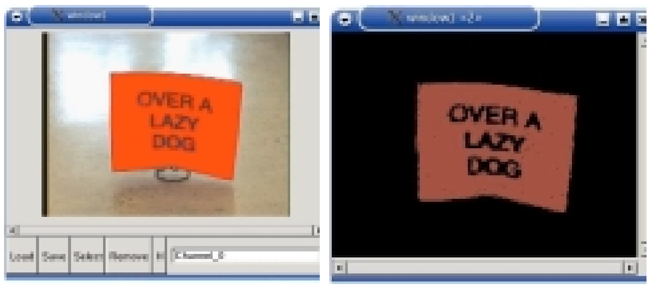

Figure 4. Color training graphical user interface (left) and color perception from the mobile robot (right)

The robot is programmed to move in the environment until it sees with its camera a black region surrounded by a colored or white area. The *Character-Tracking* behavior makes the robot stop and tries to center the black regions in the image (more specifically the center of gravity of the black regions in the image) as it zooms in to get the image with maximum resolution. PID controllers are used for the pan and the tilt commands. More details on our color segmentation algorithm are provided in [6]. A simple heuristic is used to position the zoom of the camera to maximize the resolution of the characters in the message. The algorithm allows to keep in the middle of the image the center of gravity of all of the black areas (i.e., the characters), and zooms in until the edges of the black regions of the image are within 15 pixels of the borders. If the resolution is not good enough, the *Message Processing Module* gives command to the *Direct-Commands* behavior to make the robot move closer to the message, and the process is repeated. It takes approximately from 5 to 16 seconds over a range of 2 to 10 feet for the robot to position itself and take an image with sufficient resolution for message recognition. Most of the processing time of the *Character-Tracking* behavior is taken to send small incremental zoom commands to the Sony EVI-D30 camera to insure the stability of the algorithm. Performances can be improved with a different camera with quicker response to the Pan-Tilt-Zoom commands.

3 Message Processing Module

With the image obtained by the *Character-Tracking* behavior, the *Message Processing Module* must find the lines, the words and the symbols in the message. This process is done in three steps: Image Binarization, Image Segmentation and Image Character Recognition.

3.1 Image Binarization

This first step consists of converting the image into black and white values (0,1) based on its greyscale representation. Binarization must be done carefully using proper thresholding to avoid removing too much information from the textual message. Figure 5 shows the effect of different thresholds for the binarization of the same image. In the first version of our binarization algorithm, thresholds were hard-coded and results were unsatisfactory since the algorithm did not take into consideration variations in the lighting conditions.

[1] http://robotflow.sourceforge.net

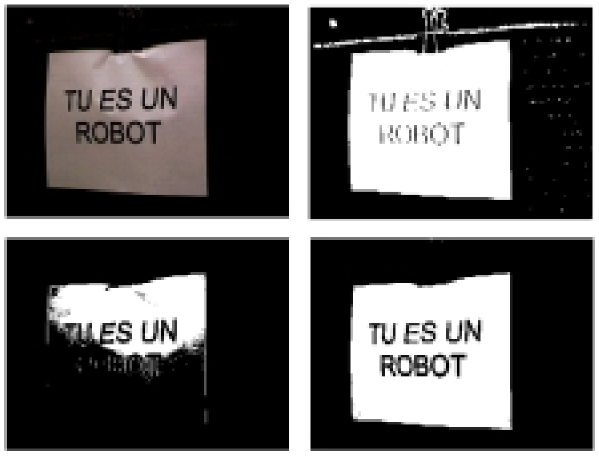

Figure 5. Effects of thresholds on binarization: (top left) original image; (top right) large threshold; (bottom left) small threshold; (bottom right) proper threshold.

We changed our algorithm to adapt the threshold automatically using the following procedure:

a) Intensity of each pixel of the image is calculated using the average intensity of the three color channels (red, green, blue). Intensity is then transformed in the [0,1] greyscale range, 0 representing completely black and 1 representing completely white.

b) Randomly selected pixel intensities in the image (empirically set to 1% of the image pixels) are used to compute the desired threshold. Minimum and maximum image intensities are found using these pixels. We experimentally found that the threshold should be set at 2/3 of the difference between the maximum and the minimum intensity of the randomly selected pixels for good results. Using only 1% of the pixels to compute the threshold offers good performances without requiring too much calculations.

c) Binarization is then performed on the whole image converting pixels into binary values. Pixels with intensity higher or equal than the threshold are set to 1 (white) while the others are set to 0 (black).

3.2 Image Segmentation

Once the image is binarized, black areas are extracted using standard segmentation methods [2][6]. The process works by looking, pixel by pixel (from top to bottom and left to right), if the pixel and some of its eight neighbors are black. Regions of black pixel connected with each other are then delimited by a rectangular bounding boxes. Each box is characterized by the positions of all pixels forming the region, the center of gravity of the region (x_c,y_c), the area of the region, and the upper left and lower right coordinates of the bounding box. Figure 6 shows the results of this process. To avoid characters being separated with many segments (caused by noise or bad color separation during the binarization process), which would cause bad interpretation of those characters, the segmentation algorithm allows connected pixels to be separated by at most three pixels. This value can be set in the segmentation algorithm and must be small enough to avoid connecting valid characters together.

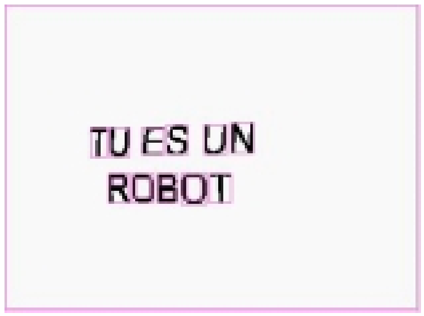

Figure 6. Results of the segmentation of black areas.

Once the black areas are identified, they are grouped into lines by using the position of the vertical center of gravity (y_c) and the height of the bounding boxes, which are in fact the characters of the message. To be a part of a line, a character must respect the following criteria :

- The height of a character must respect a preset admissible range. In our experiments, the minimum height is 40 pixels. No maximum height is specified.

- The vertical center of gravity (y_c) must be inside the vertical line boundaries. Line boundaries are found using the following algorithm : The first line, L_1, is created using the upper left character c1. Vertical boundaries for line L_1 are set to $y_{c1} \pm (h_{c1}/ 2 + K)$, with K being a constant empirically set to $h_{c1} / 2$ (creating a range equals to twice its height). For each character c_i, the vertical center of gravity y_{ci} is compared to the line boundaries of line L_j. Characters with y_{ci} inside the boundaries of line L_j are inserted into the line. If a character c_i cannot find a line to fit in, a new line is created with boundaries $y_{ci} \pm (h_{ci}/ 2 + K)$, with K being $h_{ci} / 2$. A high value of K allows to consider characters seen in a diagonal as being part of the same line. Noise can deceive this simple algorithm, but adjusting the noise tolerance can usually overcome this problem.

With the characters localized and grouped into lines, they can be grouped into words by using a similar algorithm but at the vertical instead of at the horizontal:

going from left to right, characters are grouped into a word if the horizontal distance between two characters is under a specified tolerance (set to the average character's width multiplied by a constant set empirically to 0.5). Spaces are inserted between the words found.

3.3 Character Recognition

From the first line to the last, word by word, the part of the image for each character is sent to a neural network for recognition. A feed-forward network with one hidden layer is used, trained with the delta-bar-delta [5] learning law, which adapts the learning rate of the back-propagation learning law. The network input values are in the [–1,1] range (-1 for black pixel and +1 for white pixel). The neuron activation function used is the hyperbolic tangent.

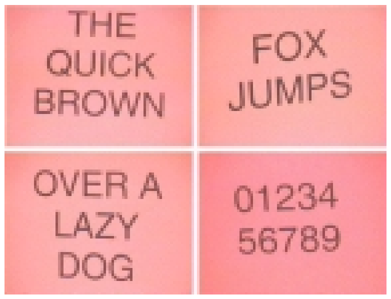

Figure 7. Messages used for training and testing the neural network

Using Arial font, we used four messages to derive our training and testing sets. The messages are shown in Figure 7 and have all of the characters and numbers. Thirty images of these four messages were taken by the robot, allowing to generate a data set of 1290 symbols.

Table 1 Recognition performance of the *Character Recognition Module*

Symbol	% Recog.	% Unrecog.	% False
1	96.7	3.3	0
2	93.3	0	6.7 (I)
3	100	0	0
4	86.7	6.7	6.6 (F,T)
5	83.3	16.7	0
6	60	30	10 (3,C)
7	100	0	0
8	56.7	6.6	36.7 (P)
9	86.7	3.3	10 (3,P)
A	98.3	0	1.7 (F)
B	96.7	3.3	0
C	100	0	0
D	100	0	0
E	98.3	0	1.7
F	100	0	0
G	96.6	3.4	0
H	100	0	0
I	96.7	0	3.3 (V)
J	78.9	18.4	2.6 (G)
K	100	0	0
L	100	0	0
M	94.7	5.2	0
N	100	0	0
O	98.7	1.3	0
P	89.4	7.9	2.6 (R)
Q	100	0	0
R	100	0	0
S	81.6	18.4	0
T	100	0	0
U	89.7	5.9	4.4 (O)
V	96.6	3.4	0
W	100	0	0
X	86.8	5.3	7.8 (F,I,N)
Y	93.1	0	6.9 (6)
Z	100	0	0

The inputs of the network is a scaled image, 13 pixels by 13 pixels, of the bounding box of the character to process. We performed several tests with different number of hidden units and by adding three additional inputs to the network (the horizontal center of gravity x_c, vertical center of gravity y_c and the height / width ratio). A character is considered recognized when the output neuron associated with this character has the maximum activation value greater to 0.8. Training is done over 5000 epochs. The best results were obtained with the use of the three additional inputs, with 7 hidden units. The network has a overall success rate of 93.1%, with 4.0% of unrecognized character and 2.9% of false recognition. The characters extracted by the *Image Segmentation Module* are about 40 pixels high. Table 1 presents the recognition performance for each of the symbols. Note that using Arial font does not make the recognition task easy for the neural network: all characters have a spherical shape, and the O is identical to the 0. In the *False* column, the characters falsely recognized are presented between parenthesis. Recognition rates are affected by the viewpoint of the robot: when the robot is not directly in front of the message, characters are somewhat distorted. We observed that characters are well recognized in the range ±45°.

4 Experimental setup and results

As shown in Figure 8, the robot used in the experiments is a Pioneer 2 DX robot with a PTZ camera and a Pentium 233 MHz installed on the PC-104 onboard computer with 64 Mb of RAM. The camera is a Sony EVI-D30 with 12X optical zoom, high speed auto-focus lens and a wide angle lens, pan range of ±90° (at a maximum speed of 80°/sec), and a tilt range of ±30° (at a maximum speed of 50°/sec).

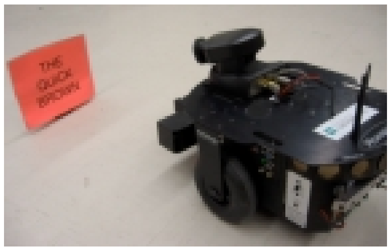

Figure 8. Experimental setup

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 x 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. RobotFlow is the programming environment used. The experiments are performed under normal fluorescent lighting conditions of our laboratory.

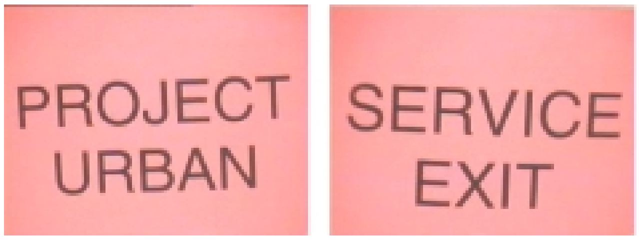

Figure 9. Validation messages

To validate the approach, the experiments consist in making the robot read different messages like the ones shown in Figure 5 and 7, and the ones in Figure 9. These last messages were chosen in order to see how the robot would perform with letters that were difficult to recognize (more specifically J, P, S, U and X). The robot took from 30 to 38 images of these messages, from different angles and ranges.

Table 2 Recognition performance of the Message Processing Module

Word	**Recognized (%)**	**Problems**	**Dictionary (%)**
THE	100	-	100
QUICK	93.3	Either U or C not recognized	100
BROWN	96.7	B not recognized	100
FOX	86.8	X recognized as I or N, or not recognized	97.4
JUMPS	57.9	Either J or P not recognized	97.4
OVER	90	E recognized as F, or R recognized as 4	96.7
A	100	-	100
LAZY	86.7	Y recognized as 6	100
DOG	93.3	G not recognized	100
PROJECT	60	T recognized as 4 ; R recognized as P ; wrong word separation PR_OJECT or PROJEC_T	83.3
URBAN	70	B recognized as R, or not recognized ; N recognized as H	96.7
SERVICE	38.7	V recognized as 6,7 or Y, or not recognized ; C not recognized	100
EXIT	100	-	100
TU	100	-	90
ES	86.7	S not recognized	90
UN	96.7	U not recognized	96.7
ROBOT	73.3	B recognized as R or 8, or not recognized	100

Table 2 shows the recognition performance of the different words recognized by the robot. The average recognition rate is 84.1%. Difficult words to read are SERVICE, PROJECT, and JUMPS because of erroneous recognition or unrecognized characters. With PROJECT however, the most frequent problem observed was caused by wrong word separation, the word being separated into two or more parts.

Performance can be easily improved by the addition of a dictionary and by changing the way we interpret the outputs of the neural network. Once the activation values are transposed to the [0,1] interval, it can be shown that the neural network outputs are a good approximation of the probabilities $P_{w,k}(w[k])$, the probability of the symbol at position *k* in the word *w*. This is caused by the mean square minimization criterion used during the training of the neural network [3]. For a given word *w* in the dictionary, the probability that X, the observation of characters recognized, for the word *w* is given by the product of the individual probabilities of each symbol in the word :

$$P(X \mid w) = \prod_{k=1}^{N} P_{w,k}(w[k]) \tag{1}$$

The word in the dictionary with the maximum probability is then selected simply by taking best match W using the maximum likelihood criterion :

$$W = \arg\max_{w} \quad P(X \mid w) \qquad (2)$$

For the messages used in our experiments, the overall performance of our method using a dictionary is 97.1%.

5 Conclusions and future work

This work demonstrates that it is possible for mobile robots to read messages autonomously, using characters printed on colored sheet and a neural network trained to identify characters in different conditions to take into consideration the various viewpoints possible. This is surely a nice starting point to increase the reading capabilities of mobile robots. Making mobile robots read textual messages in uncontrolled conditions, without *a priori* information about the world, is surely a challenging task. Our long-term objective is to improve the simple techniques proposed in the work by addressing the following issues :

- **Robot control**. There are several potential improvements to improve the Pan-Tilt-Zoom capabilities (speed, precision) of the camera. These would allow the robot to read messages from top to bottom and from left to right when they do not fit in the field of view of the camera. In the future, we would like to avoid having to stop the robot from moving while reading.

- **Image processing**. Our goal is to allow the robot to read messages on various types of background without the need to specify the background and foreground colors. This would likely require an increase in resolution of the grabbed image, as well as some image filtering to deal with character distortion and illumination variations.

- **Character recognition**. Improvements in the character recognition technique can be accomplished. More specifically, the image segmentation, word extraction and character recognition modules could be adapted to use current state-of-the-art algorithms.

- **Message understanding**. Including natural language processing techniques to extract and understand textual messages meaning would provide useful information about the messages perceived. A language model based on N-grams could help improving sentence recognition rate.

We plan to work on these improvements in continuation of our work on the challenge of making an autonomous mobile robot attend a conference [7]. Integration of all the techniques mentioned above are the key ingredients to have a mobile robot successfully accomplish its tasks in the real world. With the processing power of mobile robots increasing rapidly, this goal is surely attainable.

Acknowledgment

François Michaud holds the Canada Research Chair (CRC) in Mobile Robotics and Autonomous Intelligent Systems. This research is supported financially by the CRC Program, 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. Special thanks to Yannick Brosseau for his help in this work.

References

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

[2] J. Bruce, T. Balch, and M. Veloso (2000). "Fast and inexpensive color image segmentation for interactive robots", in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3:2061-2066.

[3] R.O. Duda, P.E. Hard, and D.G. Stork (2001). *Pattern Clasification*, Wiley-Interscience, Second Edition.

[4] H. S. Dulimarta and A. K. Jain (1997). "Mobile robot localization in indoor environment", in *Pattern Recognition*, 30(1):99-111.

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

[6] F. Michaud and D. Létourneau (2001). "Mobile robot that can read symbols", in *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*.

[7] F. Michaud, J. Audet, D. Létourneau, L. Lussier, C. Théberge-Turmel, and Serge Caron (2001). "Experiences with an autonomous robot attending the AAAI conference", in *IEEE Intelligent Systems*, 16(5):23-29.

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