計測自動制御学会東北支部 第184回研究集会 (1999.11.26)
資料番号 184-4

画面分割による移動ロボットの実時間障害物回避経路の探索

Visual Navigation using Segmentation to Localize Obstacle-free Path

○Joe Mari Maja, 高橋隆行, 王志東, 中野栄二

○Joe Mari Maja, Takayuki Takahashi, Zhi Dong Wang, Eiji Nakano

仙台市青葉区荒巻字青葉０１東北大学大学院情報科学研究科 中野研究室

Graduate School of Information Sciences, Tohoku University
Aramaki Aza Aoba 01, Aoba-ku, Sendai City 980-8579

キーワード： Obstacle Avoidance, Segmentation, Real-time, Visual Navigation, Mobile Robot

連絡先： 〒980-8579 仙台市青葉区荒巻字青葉０１東北大学大学院情報科学研究科 中野研究室
Tel.: (022)217-7025, Fax.: (022)217-7023, E-mail: maja@robotics.is.tohoku.ac.jp

1. Introduction

Segmenting in general is to produce a segmentation where there is a high correlation between the entities of the real world and the regions of the segmentation [10]. An algorithm that can segment the background and object can be used as an obstacle avoidance module. This can be used by limiting the working area of the robot to the segmented background area. This type of segmentation does not constitute to the general definition but was used to meet the objective of isolating the path.

This paper describes an obstacle avoidance algorithm that uses segmentation for visual navigation. The segmentation algorithm described and used attempts to build regions in the image based on localizing the path from other entities. In most of the indoor images sampled, a relationship of foreground-background (figure-ground) exists, which a single threshold [10] will be able to detect most of the object boundaries.

Multilevel thresholding is generally less reliable than single thresholding because of the difficulty in establishing which threshold level isolates regions of interest [15].

The best threshold value which is referred in this paper does mean that most of the ground plane was segmented as ground, since there are no accepted criteria for defining a correct segmentation [10].

2. Related Works

The system presented here is most closely related to Lorigo's [4] and Horswill [2] researches. Polly (Horswill's robot) used a minimal frame 64 × 48 and Pebbles (Lorigo's robot) used a frame size of 64 × 64. Lorigo used three different mod-

ules fused together to generate an information for obstacles. Horswill visual principle are currently being incorporated into commercial applications.

Related work in visually guided robotics has used a simplification of optical flow for obstacle avoidance [8]. Optical flow is the use of temporal variations in image intensity pattern to obtain information about the relative motion. Beebot (Coombs' robot) used the technique of equalizing normal flow between two divergent cameras. Normal flow is the flow in the direction of the normal to the image intensity gradient. A related monocular visual system implementation compares the expected flow of a flat ground plane to the perceived flow in the images [7]. If inconsistencies exist, these are considered as obstacles as it lie outside the ground plane.

The proposed system in this research is to used only segmentation technique to generate the obstacle-free plot. Due to this method, the selection of threshold value is the most important process in here. The system is depicted in Fig. 1.

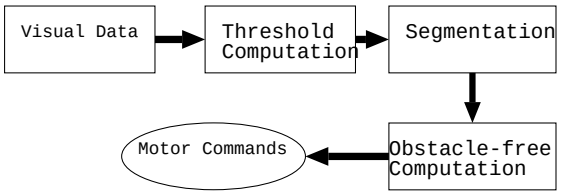


Fig. 1 Obstacle Avoidance System

3. Assumption

Assumptions has been made to simplify the acquisition of data. The assumption is that all obstacle lies on the ground and that the robot is working on a flat surface, the ground-plane constraint [3]. This means that object closer to the robot is represented at the lower level of the frame, while objects far from the robot are those found at the top level of the frame (Fig.2). Objects found nearer, which height covers the upper portion of the frame is considered as a near object due to its attachment to the ground. This constraint-transformation pair which was used by Horswill [2], and Santos [7] is also used in this research.

Fig. 2 Obstacles lie on the ground and any objects that is nearer can be found at the lower portion of the frame

4. Visual System

The visual system comprises of IP5000 that is connected to PC-AT machine and a controller board (MC68332). The IP5000 is an image processing board. A MC68332 board controls the motion of the robot (Rabbit), while the processing of image is done on PC-AT machine (Fig. 3).

A bigger image size is used (220×256) in this research. And uses a more simplified algorithm to segregate the floors from the obstacles. This can also be categorized as "lightweight vision" because of its simplicity.

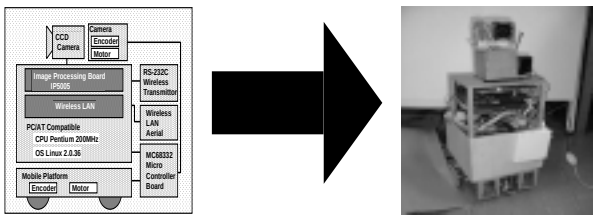


Fig. 3 Robot's component

5. Survey of Segmentation Methods

Segmentation evaluation and comparison are closely related [14], due to its distinction. The purpose of evaluating a different sets of algorithm is to quantitatively recognize its behavior and limitations in treating different various images.

There are many methods of segmenting images [5, 13, 14, 11, 12]. The most used methods will be presented. These are the Mean threshold method, *p-tile* method, Edge-pixel method, Iterative selection method, Grey-level histogram method, Entropy method and its variances, Fuzzy Method and its variances and Minimum Error Thresholding method. These methods are based on threshold selection.

All the images used for the evaluation were real-world indoor images, e.g. building hallways and office environment, since this algorithm will be used in an indoor mobile robot.

Most of the concepts of these methods are based on the distribution of the pixel in the image, e.g., Mean, *p-tile*, Minimum Error Thresholding, and Grey-Level Histogram.

The Mean method is to get the mean value of the different pixels present in the image and used it as the threshold value.

The *p-tile* method locates the first peak in the histogram and the second peak and used the lowest pixel value found in between this peak as the threshold value.

The idea of the Edge pixel method is that an edge pixel must be the boundary of an object and background. This idea was used by Weszka[1974], to produce a thresholding based on the digital laplacian. The threshold is computed by computing the Laplacian of the input image with the mask,

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

A histogram of the original image is made considering only those pixels having large laplacians [5]. The pixels which have a laplacian greater than 85% will have their grey-level appear in the histogram.

The iterative selection method as the word implies it iterates in the computation of the threshold value. The first threshold value is the mean of the image. It computes the mean level below the computed threshold and defined it as T_b, and the mean level greater than or equal to the initial threshold as T_o. The new estimate of the threshold is computed as $(T_b + T_o)/2$. If the value of the computed threshold does not change on the next iteration, the process stops.

In the Method of Grey-Level Histogram, the threshold value is computed by minimizing the ratio of the between-class variance to the total variance [11].

$$n(t) = \frac{\sigma_b^2}{\sigma_t^2} \quad (1)$$

$$\sigma_b^2 = \omega_0 \omega_1 (u_0 u_1)^2 \quad (2)$$

$$\sigma_t^2 = \sum_{i=1}^{256} (i - u_t) * (i - u_t) * p_i \quad (3)$$

$$\omega_0 = \sum_{i=0}^{t} p_i \quad (4)$$

$$\omega_1 = 1 - \omega_0 \quad (5)$$

$$u_0 = \frac{u_t}{\omega_0} \quad (6)$$

$$u_1 = \frac{u_T - u_t}{1 - \omega_0} \quad (7)$$

The value of t that gives the smallest value of n is the threshold value.

The Entropy Method used the technique in communication theory, where the image is thought of as the source of symbols. The threshold is

computed by taking t, which maximizes the equation

$$H = H_b + H_w \quad (8)$$

where, H_b is the entropy of the black pixel while H_w is the entropy of the white pixel. The entropy is computed as

$$H(X) = -\sum_{i=1}^{n} p(x_i) log(p(x_i)) \quad (9)$$

Pun[1981] also shows that maximizing H is also the same as maximizing $f(t)$

$$f(t) = X + Y \quad (10)$$

$$X = \frac{H_t}{H_T} \frac{logP_t}{log(max[p_0, p_1, ...p_t])} \quad (11)$$

$$Y = [1 - \frac{H_t}{H_T}] \frac{log(1 - P_t)}{log(max[p_{t+1}, p_{t+2}, ..., p_{255}])} \quad (12)$$

$$H_t = -\sum_{i=0}^{t} p_i log(p_i) \quad (13)$$

$$H_T = -\sum_{i=0}^{255} p_i log(p_i) \quad (14)$$

$$P_T = \sum_{i=0}^{t} p_i \quad (15)$$

H_t is the entropy of the black pixel, H_T is the total entropy and P_T is the commulative probability.

Another variance of the Entropy method was proposed by Johanssen[1982]. It divides the grey-level into two parts so as to minimize the interdependence between them [5]. This is materialize by minimizing $S_b(t)+S_w(t)$, and the corresponding t will be the threshold value.

$$S_b(t) = log(\sum_{i=0}^{t} p_i) + \frac{1}{\sum_{i=0}^{t} p_i}[E(p_t) + E(\sum_{i=0}^{t-1} p_i)] \quad (16)$$

$$S_w(t) = log(\sum_{i=t}^{255} p_i) + \frac{1}{\sum_{i=t}^{255} p_i}[E(p_t) + E(\sum_{i=t+1}^{255} p_i)] \quad (17)$$

Kapur[1985] define two distribution - An object and a background distribution, and computes the entropy of the said distribution by

$$H_b = -\sum_{i=0}^{t} \frac{p_i}{P_t} log[\frac{p_i}{P_t}] \quad (18)$$

$$H_w = -\sum_{i=t+1}^{255} \frac{p_i}{1 - P_t} log[\frac{p_i}{1 - P_t}] \quad (19)$$

The threshold value used is the value in which it maximizes $H_b(t) + H_w(t)$.

In Fuzzy Method, it used the measure of fuzziness [12], which is the distance between the grey-level image and the thresholded image. A segmented image is produced in the minimization of fuzziness. By using the entropy of the fuzzy sets, the fuzziness of the image can be measured (Shannon Function),

$$H_f(x) = -xlog(x) - (1 - x)log(1 - x) \quad (20)$$

And the entropy computation of the entire fuzzy set is

$$E(t) = \frac{1}{MN} \sum_{g} H_f(u_x(g)) h(g) \quad (21)$$

Yager[1979] proposed another measure of fuzziness by taking the degree of a set and its complement indistinction. And this can computed using this equation,

$$D_p(t) = [\sum_{g} |\mu_x(g) - \mu_{x^-}(g)|^p]^{\frac{1}{p}} \quad (22)$$

Minimum Error Thresholding Method is based on the idea of selecting thresholds based on the minimum error criterion [13]. Kittler and Illingworth [1986] created a criterion function to be minimized,

$$J(t) = 1 + 2(P_1(t) log\sigma_1(t) + P_2(t) log\sigma_2(t)) - 2(P_1(t) log P_1(t) + P_2(t) log P_2(t)) \quad (23)$$

The value of t that minimizes $J(t)$ is the threshold value to be used. A more thorough discussion of $J(t)$ can be found in the paper of Zhang, et al [14].

5.1 Sample Images

The rendition of the different threshold methods and their corresponding segmented images are shown in Appendix A. The first row are the original images and the succeding rows are the output of using Mean method, *p-tile* Method, Edge-pixel Method, GLH Method, Entropy(Pun), Entropy(Johanssen), Entropy(Kapur), Fuzzy(Entropy), Fuzzy(Yager) and Minimum Error Thresholding Method.

As shown in Appendix A, which is also true to other images sampled, the output of the different threshold methods are not consistent to different indoor images. A threshold computation that can segregate most of the free space from other entities will be discussed in the next section.

6. Analysis

The first task is to find a suitable thresholding method to be used in segmenting the background, where the computed value will vary according to the input image.

6.1 Threshold Computation

Most of the histogram plot of the indoor images taken resembles a normal curve (see Fig.4).

By sampling different images in different lighting environments (indoor), the histogram plot of most images taken can be modeled with the plot of a normal distribution. This is even true to some images which are taken outdoor.

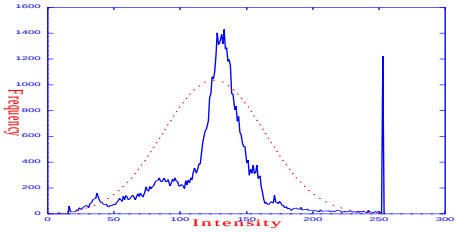


Fig. 4 Histogram Plot

And the pixel value which is one standard deviation (σ) below the mean (μ), represents a cut-off value where if used will segregate most of the floor/ground area.

The original image (Fig. 5) sampled at different threshold value and the free-space plot are shown in Fig. 6 (See Section 6.2 for the Free-space plot computation).

Fig. 5 Original Image

The first row's threshold values are 16, 18, and 20 respectively. This value is far from the $\mu - \sigma$ (located on the far left of the distribution plot). The second row's threshold are 36, 38 and 40 respectively. And the last plot of the second row is the computed best threshold value. The last row's are 68, 70 and 72 respectively. And this is located on the far right of the distribution plot.

By computing the mean (Eq. 25) and the standard deviation (Eq. 24) of the image, we can determine directly the threshold value. The threshold value which will be used as mentioned is located one standard deviation below the mean, Though, as can be seen in Fig. 6, the output of using a threshold in the range of $(\mu - \sigma - 2) \sim (\mu - \sigma + 2)$ is also acceptable, as far as the objective of this research is concerned.

$$\sigma = \sqrt{\frac{(\sum X - \mu)^2}{M}} \quad (24)$$

$$\mu = \frac{\sum X}{M} \quad (25)$$

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(X-\mu)^2}{2\sigma^2}} \quad (26)$$

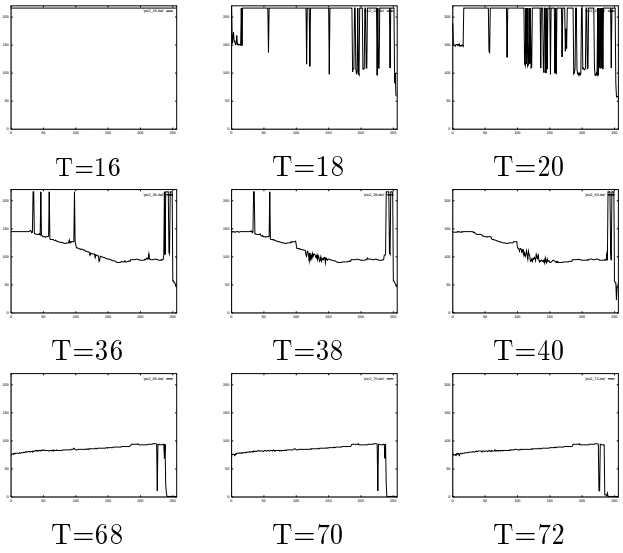


Fig. 6 Image samples of different threshold values

A segmented frame is shown in Appendix B using the $\mu - \sigma$.

6.2 Free-space search

By segmenting the image, the background which is the ground/path can be isolated. This is being done by defining a (0) to the edge of the obstacles and (1) to obstacle-free area directly in front of the robot. Which means that the computation of obstacle-free area begins at the lower portion and ends only after it encounters a (0) on each column of the image. Whatever is the data after the (0) pixel will be discarded. Figure 7 is the original image grabbed and the bi-level format of the original image.

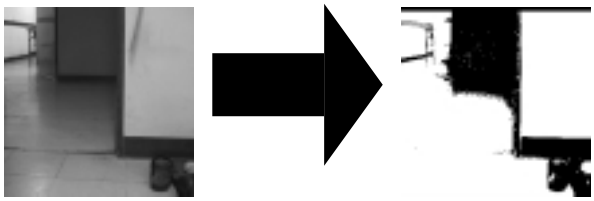


Fig. 7 Grey-Level image and Bi-level image

By plotting the resulting image, an area which is defined as an obstacle-free area is found. This is the AREA I of Fig. 8.

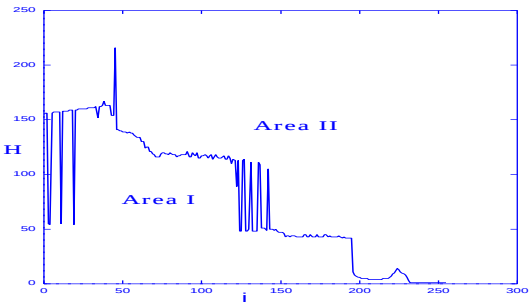


Fig. 8 The plot of the obstacle-free area

This plot is taken by calculating the distance from the bottom until it encounters a black spot on the image, it then records the pixel count and start to count again on its next adjacent pixel. The image frame size is 220 × 256. The calculation stops after it reaches the 256th vertical line.

$\boldsymbol{H}$ is a row vector, which represents the distance of obstacles from the bottom of the image.

$$H(i) = \sum_{j=0}^{Q} I(i,j) \quad (0 < i \leqq 255) \tag{27}$$

$$Q = \begin{cases} min(j), & \text{for } I(i,j) = 0 \\ 219, & \text{for } I(i,j) \neq 0 \end{cases}$$

With this calculation, the obstacle-free area as shown in Figure 9 can be plotted. And the next possible way/path can also be located. This is being done by locating the biggest area on the image with respect to the robot's width.

$\boldsymbol{A}$ is a row vector which represents the area computed from k to $k + Robotwidth$.

$$N = Framewidth - Robotwidth \tag{28}$$

$$A_k = \sum_{i=k}^{k+Robotwidth} min[H(i)] \quad (0 \leqq k \leqq N) \tag{29}$$

$$A_k = \sum_{i=k}^{k+Robotwidth} min[\sum_{j=0}^{219} I(i,j)] \quad (0 \leqq k \leqq N) \tag{30}$$

We can now locate which has the biggest computed area (Eq. 31).

$$max[A_k] \tag{31}$$

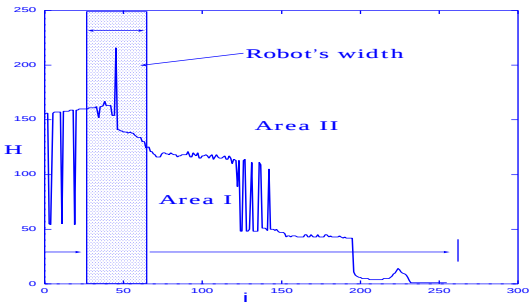


Fig. 9 Sample calculation of the biggest area in the frame

The number of calculated area depends on the frame's width and robot's width. In this research, 256 pixel [count] for the frame width and 25 [cm.] for the robot's width, which corresponds to 20 pixel [count] (Fig. 9).

7. Operations and Processing Time

The processing time is dependent to the computer. In a Pentium MMX 200 MHz computer where IP5000 is connected, it can process the data in 59 - 89 ms. This suggest that it is capable of processing 11 frames per seconds. The frame size as mentioned is 220 × 256. A minimal frame size has also been tested (25% of the current frame size - 64 × 55) and has greatly reduced the processing time to 2.150 - 2.8 [ms.], that is, 86 [ms.] less than the original frame size. With the said procesing time, 357 - 465 frames can be processed in one second. Though, currently, the system in this research used the 256 × 220 frame size, as the minimal frame size wasn't fully tested.

Below are the different operations that occur for each frame shot:

1. Grab Frame
2. Compute the threshold (Mean and Standard Deviation computation)
3. Compute the $H(i)$
4. Compute A_k and $max[A_k]$

After this process, the data extracted from 4 will then be transported to the MC68332 controller board for commands to the motor via serial link. The data is updated after every frame processing ends. This also signify that the system will be of reactive type, where the data comes mainly from whatever is being perceived by the CCD.

Figure 10 are samples of the real-time redirection of the algorithm, it determine where's the biggest free-space plot and draws a line where the robot should go. This frame is taken where the robot tries to search the free-space by turning to either left or right. After it can find it, it will try to make the new location as its center. Figure 11 are sample frames of the robot (Rabbit), in searching the free-space area. It tries to search the bigger space in front of it and redirect itself to that particular area.

Refer to Table 1 for the processing time of each function in the algorithm.

Table 1 Processing time for each frame

IP5000 → Arrays + Mean	50 - 80 ms.
Deviation + Threshold	5 ms.
Greyscale Segmentation	2 ms.
Search for Biggest Area	0 ms.
Distance Tabulation	2 ms.

Fig. 10 Sample of path redirection

Fig. 11 Test run on Rabbit

8. Failure Modes

Though most of the normal surface/floor markings have been discarded by using the $\mu - \sigma$ threshold computation, a strong color mark on the floor with a width of 1 [cm] will not be discarded by this algorithm. This will be treated as an object, though it does not possess a substantial heights that will classify the object as an obstacle. With this, the robot will obviously avoid the area, thus a false positive.

And if there is a high correlation of object with respect to the ground then, the object will be classified as ground.

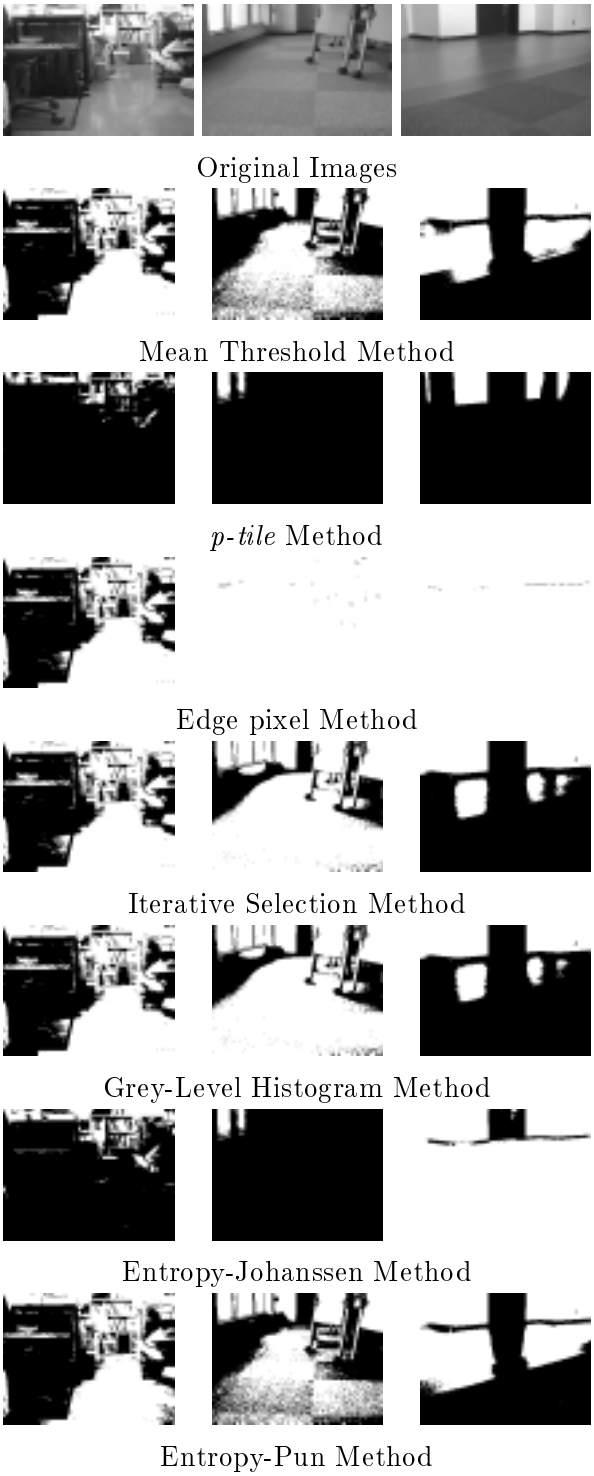
9. Conclusion

An obstacle avoidance algorithm was presented which is based mainly on segmentation module. This module segregates a passable and non-passable path. The salient point of this research is its minimal computation, which yields a minimal processing time, for indoor environment with various lighting situation. This provides the basic technique in visual navigation and a possible combination of other visual cues in the future will produce a more robust algorithm.

Also presented was a solution on how to determine the suitable threshold value for used in this research. This $\mu - \sigma$ were based on the different frames that were sampled at different lighting environment.

The processes was minimized by excluding an edge-detection module, since the presence of an edge does not really constitute an obstacle. A good example of this one is the surface/floor markings.

10. Appendix A



Entropy-Kapur Method

Fuzz-Entropy Method

Fuzz-Yager Method

Minimum Error Thresholding Method

Figure 12. Different Methods and segmented frames

11. Appendix B

Original Images

$\mu - \sigma$ Method

Fig. 13 Using the $\mu - \sigma$ as threshold

References

1) Mitsueda, S.: Real-time obstacle avoidance using single camera, *Master's Thesis*, Tohoku University, (1990)

2) Horswill, I.D.: Specialization of Perceptual Processes, *PhD Theses, Massachusett Institute of Technology*, Cambridge, May (1993)

3) Horswill, I.D.: Polly: A vision based artificial agent, *11th National Conference on AI*, pp. 824-829, (1993)

4) Lorigo, L.M., Brooks, R.A., and Grimson, W.E.L.: Visually-guided obstacle avoidance in unstructured environments, *IROS*, (1997)

5) Parker, J.R.: Algorithms for Image Processing and Computer Vision, *John Wiley and Sons*, USA, (1997)

6) Broesch, J.D.: Digital Signal Processing Demystified, *High Text Publications*, Solana Beach, CA, (1997)

7) Santos-Victor, J., Sandini, G., Curotto, F. and Garibaldi, S.: Uncalibrated obstacle detection using normal flow, *International Journal of Computer Vision*, pp. 159-177, (1995)

8) Coombs, D., and Roberts K.: BeeBot: Using peripheral optical flow to avoid obstacles, *Intelligent Robots and Computer Vision*, Vol. 1825, SPIE, pp. 714-721, (1992)

9) Ballard D.H and Brown, C.M.: Computer Vision, *Prentice Hall Inc.*, Englewood Cliffs, New Jersey, (1982)

10) Kohler, Ralf: A Segmentation System Based on Thresholding, *Computer Graphics and Image Processing*, 15, pp. 319-337, (1981)

11) Otsu, N.: A Threshold Selection Method from Grey-Level Histogram, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9. 1:377-393, 1979

12) Huang, L.K. and Wang, M-J J.: Image Thresholding by Minimizing the Measures of Fuzziness, *Pattern Recognition*, Vol. 28. 1:41-51, 1995

13) Jiulun, Fan and Winxin, Xie: Minimum Error Thresholding: A note, *Pattern Recognition Letters*, Vol. 18, pp. 705-709, 1997

14) Zhang, Y.J.: Evaluation and Comparison of different segmentation algorithms, *Pattern Recognition Letters*, Vol. 18, pp. 963-974, 1997

15) Fu, K.S., Gonzalez, R.C., and Lee, C.S.G.: ROBOTICS: Control, Sensing, Vision, and Intelligence, *McGraw-Hill International Editions*, pp.357-359, 1987