Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 – Nov. 03, 2001

Realtime Omnidirectional Stereo for Obstacle Detection and Tracking in Dynamic Environments

Hiroshi Koyasu, Jun Miura, and Yoshiaki Shirai
Dept. of Computer-Controlled Mechanical Systems,
Osaka University, Suita, Osaka 565-0871, Japan
{koyasu,jun,shirai}@cv.mech.eng.osaka-u.ac.jp

Abstract

This paper describes a realtime omnidirectional stereo system and its application to obstacle detection and tracking for a mobile robot. The stereo system uses two omnidirectional cameras aligned vertically. The images from the cameras are converted into panoramic images, which are then examined for stereo matching along vertical epipolar lines. A PC cluster system composed of 6 PCs can generate omnidirectional range data of 720x100 pixels with disparity range of 80 about 5 frames per second. For obstacle detection, a map of static obstacles is first generated. Then candidates for moving obstacles are extracted by comparing the current observation with the map. The temporal correspondence between the candidates are established based on their estimated position and velocity which are calculated using a Kalman filter-based tracking. Experimental results for a real scene are described.

1 Introduction

Detection of obstacles and free spaces is an essential function of the vision system for mobile robots. Even if a robot is given a map of the environment, this function is indispensable to cope with unknown, possibly dynamic, obstacles or errors of the map. Many works (e.g., [8]) use laser range finders to detect obstacles. Laser range finders which scan a 2D plane have a drawback that objects at a specific height can only be detected. Stereo vision is also used widely (e.g., [4]). Conventional stereo systems, however, suffer from a narrow field of view because they usually use a pair of ordinary cameras.

In the case of mobile robots, it is sometimes necessary to obtain panoramic range information of 360 degrees because obstacles may approach from various directions. There are two methods to obtain panoramic images. One is to capture a sequence of images with rotating a camera and then to integrate the images into one panoramic image (e.g., [5, 10]). Although this method could obtain a high-resolution image, it takes a long time to get an image and is not suitable for robots in dynamic environments. The other method is to use a special lens or mirror to obtain an omnidirectional image at once. Although the image resolution is low, its realtime nature is suitable for mobile robot applications.

This paper describes a realtime omnidirectional stereo system and its application to detection of static and dynamic obstacles. The system uses a pair of vertically-aligned omnidirectional cameras. The input images are converted to panoramic images, in which epipolar lines become vertical and in parallel. The stereo matching is performed by a PC cluster system to realize a realtime range calculation for a relatively large image size. For obstacle detection, a map of static obstacles is first generated. Then candidates for moving obstacles are extracted by comparing the current observation with the map. The temporal correspondence between the candidates are examined based on their estimated position and velocity which are calculated using a Kalman filter-based tracking.

2 Realtime Omnidirectional Stereo

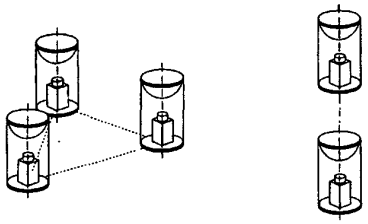
2.1 Configuration of Omnidirectional Cameras

We use multiple HyperOmni Visions (HOVIs) [9] for omnidirectional stereo. A HOVI uses a hyperboloidal projection, which has an advantage that the input image can easily be converted to an arbitrary image with a single effective viewpoint at the focal point of the hyperboloidal mirror.

In the conventional stereo configuration where two normal cameras are aligned in parallel, all epipolar lines are horizontal; this leads to an efficient stereo matching algorithm [2]. Since we are interested in obtaining dense range images, it is desirable that epipolar lines are in parallel so that such an efficient algorithm can be applied.

There are basically two ways of configuring multiple HOVIs for omnidirectional stereo which satisfy the above condition concerning epipolar lines. One is to align them horizontally (see Fig. 1(a)). To make epipolar lines in parallel in this configuration, we convert a pair of omnidirectional images to a pair of normal perspective-projected images to be obtained by virtual, parallel-aligned cameras placed on the optical axes of HOVIs. This configuration

0-7803-6612-3/01/$10.00©2001 IEEE



(a) horizontal alignment (b) vertical alignment

Fig. 1: Two stereo configurations.

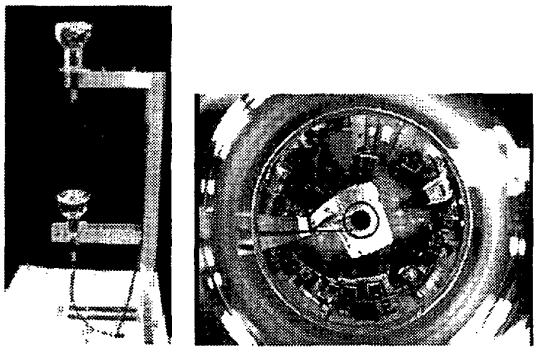


Fig. 2: Stereo setup and an example input image.

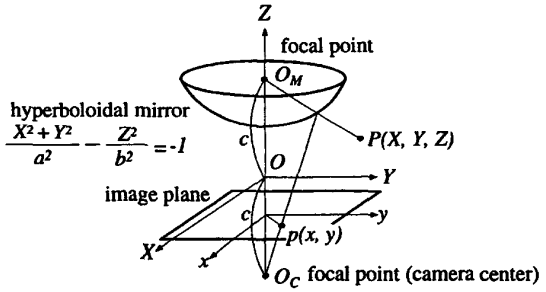


Fig. 3: Geometry of hyperboloidal projection[9].

has two drawbacks: limited field of view and large deformation in the peripheral areas of the converted images. In addition, to obtain a 360-degree range image, we have to use at least 3 HOVIs placed, for example, on the vertices of a regular triangle.

The other configuration is to align HOVIs vertically (see Fig. 1(b)). Each omnidirectional image is converted to a panoramic image on a cylindrical image plane whose axis is aligned to those of the HOVIs. By this conversion, all epipolar lines become vertical in the panoramic images.

Based on the above consideration, we selected the latter configuration, which is the same as the one proposed by Gluckman et al. [3]. Fig. 2 shows a pair of vertically-aligned omnidirectional cameras. In the current setup, since each omnidirectional camera is supported by a cantilever attached to a vertical bar, there is a blind spot in the omnidirectional images.

2.2 Image Conversion

From the geometry of HOVI shown in Fig. 3, we obtain the following relationship between scene position (X, Y, Z) and image position (x, y):

$$x = \frac{Xf(b^2 - c^2)}{(b^2 + c^2) \cdot (Z - c) - 2bc\sqrt{(Z - c)^2 + X^2 + Y^2}},$$

$$y = \frac{Yf(b^2 - c^2)}{(b^2 + c^2) \cdot (Z - c) - 2bc\sqrt{(Z - c)^2 + X^2 + Y^2}},$$

where a and b are the parameters of the mirror shape; c is the half of the distance between the focal points of the mirror and the camera; f is the focal length of the camera.

To generate a panoramic image, we first set a virtual cylindrical image plane around the vertical axis. For the cylindrical image plane, we currently use the following parameters: 10 degrees and 30 degrees for the viewing angles above and below the focal point of the mirror, respectively; 720 and 100 for the horizontal and the vertical resolution. From these parameters, we can determine (X, Y, Z) for each pixel, thereby, calculating the corresponding image position. Fig. 4 shows the panoramic image converted from the omnidirectional input image shown in Fig. 2.

2.3 Stereo Matching Algorithm

A SAD-based matching algorithm is used. For each pixel in the right image, the corresponding point is searched for on the epipolar line within a predetermined disparity range. As a matching criterion, we use SAD (sum of absolute difference) of the intensity value in a window around the points. If the SAD value is small enough, two points are considered to match. We also adopt the above-mentioned efficiency matching algorithm and a consistency checking [4] to increase the efficiency and the reliability. Fig. 5 shows a result of matching, in which larger disparities (nearer points) are drawn in brighter color.

2.4 Implementation on PC Cluster

The steps to obtain omnidirectional range data are: (1) capturing a pair of omnidirectional images, (2) converting the images into panoramic images, and (3) finding matched points between the images. To realize a (near) realtime range data acquisition, we parallelize the third step using a 6-PC (Pentium III, 600MHz) cluster system as shown in Fig. 6. The size of the panoramic image is 720x100 and the disparity range to search is 80. The current throughput is about $0.2[s]$ per frame. Since the current algorithm is not fully optimized (for example, we have not used the MMX instructions), we are expecting to improve the performance in a near future.

Fig. 4: Panoramic image obtained from the input image shown in Fig. 2.

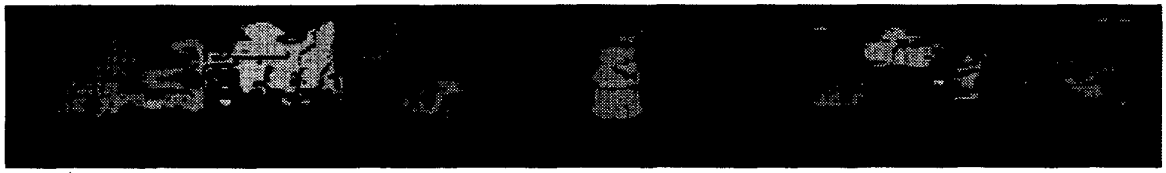


Fig. 5: Panoramic disparity image obtained from the images in Fig. 4. Brighter pixels are nearer.

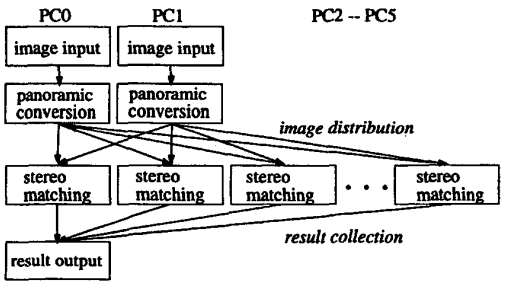


Fig. 6: PC cluster implementation.

2.5 Implementation on a Single PC

The PC cluster-based system may not be suitable for mobile robot applications due to its size. We are, therefore, investigating another system which uses an image merger to capture a pair of omnidirectional images by one frame grabber and a single PC (Pentium III, 850MHz); the current implementation generates the disparity image of 360x50 in size and 40 in disparity range. The current throughput is about $0.18[s]$ per frame without MMX acceleration. One potential problem in applying this system to dynamic obstacle detection and tracking is its low image resolution, i.e., low spatial and disparity resolution in range data may make it difficult to detect obstacles. We are now experimentally examining the problem.

3 Making a Map of Static Obstacles

3.1 Calculating 2D Range Data

To make a map of static obstacles, we first extract the nearest obstacle in each direction. Since the horizontal axis of the panoramic image indicates the horizontal direction, we examine each column of the panoramic disparity image vertically and extract the connected interval in which the disparity difference between (vertically) adjacent pixels is less than or equal to one and which is the nearest among such intervals in the same column. By applying this operation to every column, we obtain a set of distances of 360 degrees. From this data set, the 2D contour (called *range profile*) of the current free space centered at the robot position is obtained.

3.2 Integrating Multiple Observations Considering Vision Uncertainty

3.2.1 Vision Uncertainty

We consider two factors in estimating the uncertainty of range data. One is the quantization error in panoramic images. Let d be the disparity of a matched point and R $\left(=\sqrt{X^2+Y^2}\right)$ be the distance to the point in the scene. We obtain $R = Bf'/d$, where B is the baseline (the distance between the focal points of two mirrors, currently $30[cm]$) and f' is the focal length of the *virtual* camera used for generating panoramic images. Considering the ± 1 pixel uncertainty in d, we can calculate the maximum and the minimum possible distance R_{max} and R_{min} by:

$$R_{max} = \frac{Bf'}{d-1}, \qquad R_{min} = \frac{Bf'}{d+1}. \tag{1}$$

The other factor is the blurring effect of the panoramic conversion; that is, a blurred panoramic image is obtained if the resolution of the panoramic image is higher than that of the original one. This blurring effect varies depending on the vertical position in the panoramic image.

The following equation represents the relationship between r, the distance from the image center in the radial direction in the original omnidirectional image, and y, the vertical position in the panoramic image:

$$r = \frac{f(b^2-c^2)}{(b^2+c^2)(U-Ly/h) - 2bc\sqrt{1+(U-Ly/h)^2}},$$
$$U = \tan\theta_u, \quad L = \tan\theta_u + \tan\theta_l,$$

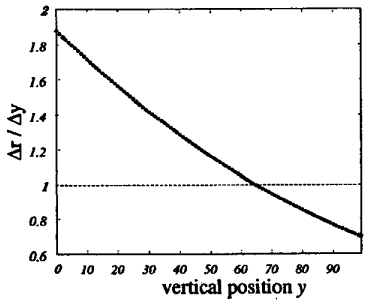


Fig. 7: Effect of image blur.

where θ_u and θ_l are the viewing angle above and the below of the focal point of the mirror; h is the height of the panoramic image. From this equation, we can calculate how many pixels in the original omnidirectional image corresponds to one pixel in the panoramic image at a specific vertical position, denoted as $\Delta r/\Delta y$. Fig. 7 shows the calculation result of $\Delta r/\Delta y$ at all vertical positions; the lower part (larger y) of the panoramic image is degraded because that part corresponds to the central part of the original omnidirectional image, where the resolution is low compared with the peripheral areas. For the positions where $\Delta r/\Delta y$ is less than one, the uncertainty in disparity should be considered larger than ± 1. In that case, the possible range of the distance (see eq. (1)) is increased accordingly.

Using the above uncertainties, we interpret each point of disparity d in the range profile as follows. In Fig. 8, the angle θ indicates the angular resolution of the panoramic image (currently, 0.5 degrees); $R(d)$, $R_{max}(d)$, and $R_{min}(d)$ are the distance of the point from the robot and their maximum and minimum values mentioned above. We use the dark gray trapezoid in the figure to approximate the region where an obstacle may exist. We call this region an *obstacle region*. The area before the obstacle region approximated by the light gray triangle, called a *safe region*, is the region where an obstacle never exists as long as the stereo matching is correct. Safe regions are used for making a map of static obstacles, while obstacle regions are used for detecting moving obstacles (see Sec. 4.1).

3.2.2 Generation of Free Space Map

We use a grid representation for the map. To cope with false matches in stereo, we accumulate multiple observations to obtain reliable map information. Each grid of the map holds the counter which indicates how many times the grid has been observed to be safe. At each observation, the counter of each grid inside the safe regions is incremented. If the counter value of a grid is higher than a certain threshold, the grid is considered safe. The set of safe grids constitutes the current free spaces (called a *free space map*).

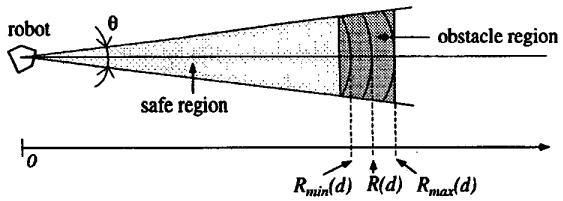


Fig. 8: Safe region and obstacle region.

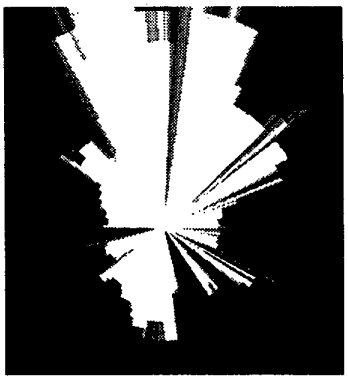
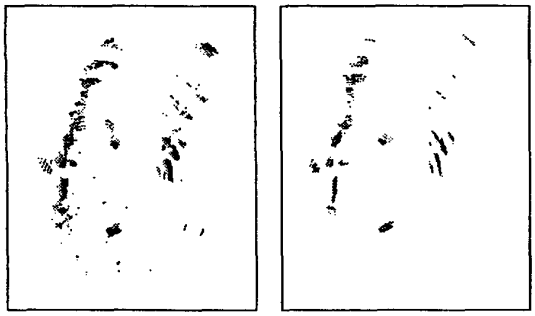


Fig. 9: An example map. White region indicates the free space; gray regions indicate the area where the observation count is less than the threshold.

Since the robot makes a map while it moves, we first transform the observed data using the current position (including orientation) of the robot with respect to some fixed world coordinates, and then integrate the transformed data to the map. The robot position is estimated from the odometry and visual information. The localization method will be described later. To reduce the effect of accumulated error when the robot moves by a long distance, we use only ten latest observations for making the free space map, and then use five as the threshold for the observation counter mentioned above. Fig. 9 shows an example map.

3.2.3 Vision-Based Localization

We adopt a visual localization method similar to [7], which is based on the comparison between the current and the previous range profile. The method first predicts the uncertainty region for the robot position using the uncertainty model of robot motion. Candidates for robot position are grid points inside the uncertainty region. The range of possible robot orientations is also determined from the uncertainty model. Then, for each pair of candidate position and orientation, the view of the previous range profile is calculated and compared with the current one. Only relatively large continuous segments in the range profiles are used for comparison. Finally, the best pair is selected which maximizes the similarity of two range profiles. Fig.



(a) with dead reckoning. (b) with visual localization.

Fig. 10: Effect of visual localization.

10 shows the effect of our visual localization method. The figure shows the superimposition of obstacle regions obtained by 14 consecutive observations. In Fig. 10(a), the robot position is estimated only by dead reckoning, while in Fig. 10(b), the above visual localization method is used. The method is still a preliminary one and its quantitative evaluation is necessary.

4 Detecting and Tracking Moving Obstacles

4.1 Extracting Moving Obstacle Candidates

Candidates for moving obstacles are detected by comparing the current observation with the free space map. Considering the uncertainty in observation, if the obstacle region (see Fig. 8) of a point in the range profile is completely inside the free space, the point is considered as a part of a moving obstacle. Since the points from the same obstacle may split into several obstacle regions, we merge a set of moving points if their relative distance is less than a certain threshold (currently, $40[cm]$). We consider a merged group of such points as a candidate for moving obstacle and use their mass center as its observed position.

4.2 Tracking Using Kalman Filter

The state $\boldsymbol{x}_t$ of a moving obstacle is represented by:

$$\boldsymbol{x}_t = (x_t, y_t, \dot{x}_t, \dot{y}_t)^T,$$

where (x_t, y_t) and $(\dot{x}_t, \dot{y}_t)$ are the position and the velocity in the world coordinates at time t. The robot obtains the following observation $\boldsymbol{y}_t$ for each moving obstacle:

$$\boldsymbol{y}_t = (x_t^o, y_t^o)^T,$$

where (x_t^o, y_t^o) is the observed position of the obstacle. Supposing a constant velocity of a moving obstacle, we obtain the following state transition equation and observation equation:

$$\begin{aligned} \boldsymbol{x}_{t+1} &= F_t \boldsymbol{x}_t + \boldsymbol{w}_t, \\ \boldsymbol{y}_t &= H_t \boldsymbol{x}_t + \boldsymbol{v}_t, \end{aligned}$$

where $\boldsymbol{w}_t$ and $\boldsymbol{v}_t$ are the error terms. $\boldsymbol{w}_t$ represents the acceleration of the obstacle; the maximum allowed acceleration (i.e., so-called 3σ value) is set to $1[m/s^2]$. $\boldsymbol{v}_t$ represents the observation error (see Sec. 3.2.1). The matrices in the above equations are given by:

$$F_t = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$H_t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix},$$

where Δt is the cycle time.

We use the Kalman filter [6] to track moving obstacles. The filter is initiated when a candidate for moving obstacle is detected at the first time. The initial positional uncertainty is set to the vision uncertainty when first detected.

4.3 Making Correspondence

Using the above motion model, we can predict the position of a moving obstacle and its uncertainty. In order to check the correspondence between an obstacle being tracked and a candidate for moving obstacle detected in the current observation, we calculate the following Mahalanobis distance d_M between the predicted and the observed positions:

$$\begin{aligned} d_M &= d_{\boldsymbol{y}}^T \left(H_t P_{t/t-1} H_t^T\right)^{-1} d_{\boldsymbol{y}}, \\ d_{\boldsymbol{y}} &= \boldsymbol{y}_t - H_t \boldsymbol{x}_{t/t-1}, \end{aligned}$$

where $\boldsymbol{x}_{t/t-1}$ and $P_{t/t-1}$ are the predicted state and its uncertainty, respectively. If d_M is less than a given threshold (currently, 9.21), the correspondence is considered correct.

In some cases, however, the correspondence may not be one-to-one for the following reasons. First, although we merge neighboring points to form a moving obstacle candidate (see Sec. 4.1), it is possible that a single object is observed as two or more separate objects; for example, the left and the right edge of a person may be observed as two separate objects. In addition, there are occasional false objects caused by the error in stereo matching. Therefore we adopt a *track-splitting filter* [1] which considers all possible correspondence and generates a tree of possible tracks. If the corresponding observation is not obtained for a branch of the tree for a certain number of frames (currently, three), the branch is deleted.

Since there is still some uncertainty in localization, parts of static obstacles are sometimes detected as candidates for moving obstacles. As a result, there may be false moving obstacles among the remaining branches in the tracking tree. We then use the estimated velocity to discriminate truly moving obstacles from false ones. If the estimated velocity of a moving obstacle being tracked is less than $60[cm/s]$, that object is considered to be a part of some static obstacle.

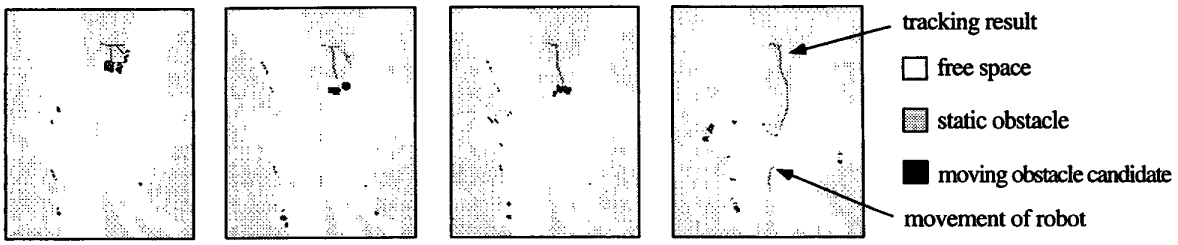


Fig. 11: A tracking result.

Fig. 12: Detected person. The estimated person region is projected onto the panoramic image (black pixels).

5 Experimental Results

The total throughput of omnidirectional stereo, visual localization, update of free space map, and dynamic object detection and tracking is currently about $0.38[s]$ per frame. Fig. 11 shows a tracking result in the case where one person was approaching the robot. In earlier frames (left ones), there are several branches in the tracking tree (i.e., several traces exist). As time elapses (right frames), all branches except the true one are deleted. Fig. 12 shows the projection of the estimated position of the person onto the panoramic images. The figure shows that the tracking was correctly performed.

6 Conclusion

We have developed a realtime omnidirectional stereo system using two omnidirectional cameras and a PC cluster. The system can generate omnidirectional range data of 720x100 pixels with disparity range of 80 about 5 frames per second. We also have developed a method of detecting and tracking moving obstacles from omnidirectional range data obtained by a mobile robot. The experimental results for a real scene show the validity of the method. A future work is to apply the system to mobile robot navigation (e.g., avoiding moving obstacles approaching from various directions). Another future work is to develop a compact on-robot omnidirectional stereo system.

Acknowledgments

The authors would like to thank Prof. Yagi of Osaka University for his useful advice on omnidirectional cameras. This research is supported in part by the Kurata Foundation, Tokyo, Japan.

References

[1] I.J. Cox. A Review of Statistical Data Association Techniques for Motion Correspondence. *Int. J. of Computer Vis.*, Vol. 10, No. 1, pp. 53–66, 1993.

[2] O. Faugeras et al.. Real-Time Correlation-Based Stereo: Algorithm, Implementation and Application. Tech. Report 2013, INRIA Sophia Antipolis, 1993.

[3] J. Gluckman, S.K. Nayar, and K.J. Thoresz. Real-Time Omnidirectional and Panoramic Stereo. In *Proc. of Image Understanding Workshop*, 1998.

[4] S. Kagami, K. Okada, M. Inaba, and H. Inoue. Design and Implementation of Onbody Real-time Depthmap Generation System. In *Proc. ICRA-200*, pp. 1441–1446, 2000.

[5] S.B. Kang and R. Szeliski. 3-D Scene Data Recovery Using Omnidirectional Multibaseline Stereo. *Int. J. of Computer Vis.*, Vol. 25, No. 2, pp. 167–183, 1997.

[6] T. Katayama. *Application of Kalman Filter*. Asakura Shoten, 1983. (in Japanese).

[7] K. Kidono, J. Miura, and Y. Shirai. Autonomous Visual Navigation of a Mobile Robot Using a Human-Guided Experience. In *Proc. IAS-6*, pp. 620–627, 2000.

[8] E. Prassler and J. Scholz. Tracking Multiple Moving Objects for Real-Time Navigation. *Autonomous Robots*, Vol. 8, No. 2, pp. 105–116, 2000.

[9] K. Yamazawa, Y. Yagi, and M. Yachida. Omnidirectional Imaging with Hyperboloidal Projection. In *Proc. IROS-93*, pp. 1029–1034, 1993.

[10] J.Y. Zheng and S. Tsuji. Panoramic Representation of Scenes for Route Understanding. In *Proc. 10th ICPR*, pp. 161–167, 1990.