J. of Robotics and Mechatronics, Vol. 14, No. 2,
pp. 147-156, 2002.

Recognizing Moving Obstacles for Robot Navigation using Real-time Omnidirectional Stereo Vision

Hiroshi Koyasu, Jun Miura, and Yoshiaki Shirai
Dept. of Computer-Controlled Mechanical Systems, Osaka University

We describe a method to recognize moving obstacles in a wide view and in real-time. Such an ability is required when a mobile robot moves in a dynamic environment. Our method uses an omnidirectional stereo vision which is composed of a pair of vertically-aligned omnidirectional cameras and a PC cluster to obtain panoramic range information of 360 degrees in real-time. From this range information, the robot on-line generates a free space map of the surrounding environment, and extracts objects in the free space as candidates for moving obstacles. The robot makes temporal correspondence of the candidates and estimates their position and velocity using the Kalman filter. To reduce the effect of odometry error to map generation, the ego-motion is estimated by comparing the current and previous range data. We demonstrate the effectiveness of our method by on-line experiments.

1 Introduction

Detection of obstacles and free spaces is an essential function of the sensing system for mobile robots. Even if a robot is given a map of the environment, this function is indispensable to cope with unknown obstacles or error of the map. Especially in dynamic environment, robot must recognize the position and the velocity of moving obstacles to avoid collision.

Many works (e.g., [10]) use laser range finders to detect obstacles. Laser range finders which scan a 2D plane have a drawback that object 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, 12]). Although this method could obtain a high-resolution image, it takes a long time to get an image and is not suitable for robot 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 real-time nature is suitable for mobile robot applications.

Prassler et al. [10] proposed a method of tracking moving objects using a laser range finder. The method detects moving obstacles by calculating the difference between the current and the previous obstacle positions on a grid map. The method does not consider the uncertainty in range data. Moreover, since they estimate the robot position only by dead reckoning, the accumulated positional error of the robot may degrade the obstacle map, thereby, detecting static obstacles as moving ones. The method by Lindström et al. [8] may have a similar problem. Lu et al. [9] proposed an ego-motion estimation using a laser range finder to compensate for the error of dead reckoning; however, their ego-motion estimation does not seem to be applicable to the case when the uncertainty of range data is relatively large.

This paper describes a method to recognize moving obstacles in a wide view and in real-time using a real-time omnidirectional stereo system. The system uses a pair of omnidirectional cameras aligned to a vertical line. The input images are converted to panoramic images, in which epipolar lines become vertical and in parallel. Therefore we can apply efficient stereo matching algorithm for conventional stereo images. The stereo matching is performed by a PC cluster system to realize a real-time range calculation for a relatively large image size. Panoramic range information is obtained from this stereo image. For obstacle detection, a map of static obstacles is first generated from the range information. 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. The 2D range information is also used for an on-line ego-motion estimation to reduce the effect of odometry error on map generation.

The rest of the paper is organized as follows. Section 2 describes the real-time omnidirectional stereo system. Section 3 describes obtaining a 2D range information method and the ego-motion estimation. Section 4 describes a method of generating the free space map. Section 5 describes a method of detection and tracking of dynamic obstacles. Section 6 describes experimental results. Section 7 concludes the paper.

2 Real-time Omnidirectional Stereo

2.1 Configuration of Omnidirectional Camera Pair

We use multiple HyperOmni Visions (HOVIs) [11] 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.

To satisfy the condition on epipolar lines, we align two HOVIs vertically; this configuration is the same as the one proposed by Gluckman et al. [3]. Each omnidirectional image is converted to a panoramic image on a cylindrical image plane

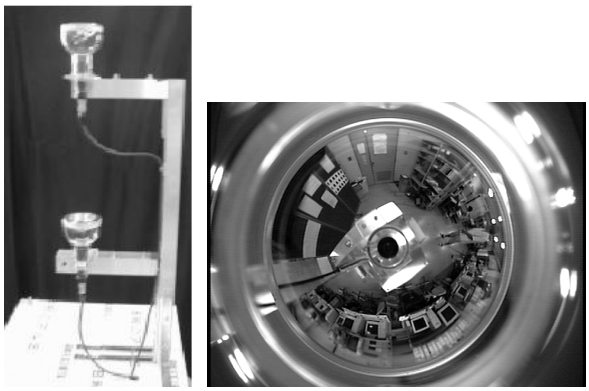


Fig. 1: Omnidirectional stereo setup and an example input image.

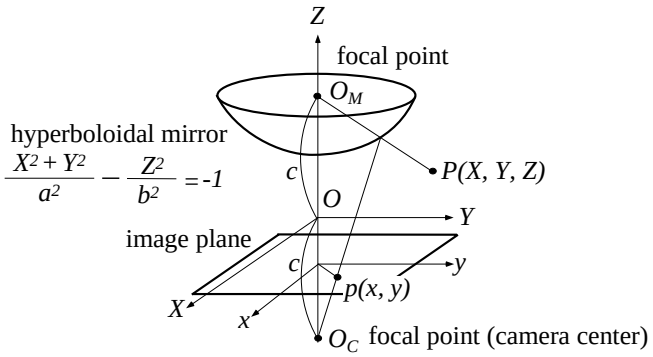


Fig. 2: Geometry of hyperboloidal projection[11].

whose axis is aligned to those of the HOVIs. By this conversion, all epipolar lines become vertical in the panoramic images. Fig. 1 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. 2, 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. 3 shows the panoramic image converted from the omnidirectional input image shown in Fig. 1.

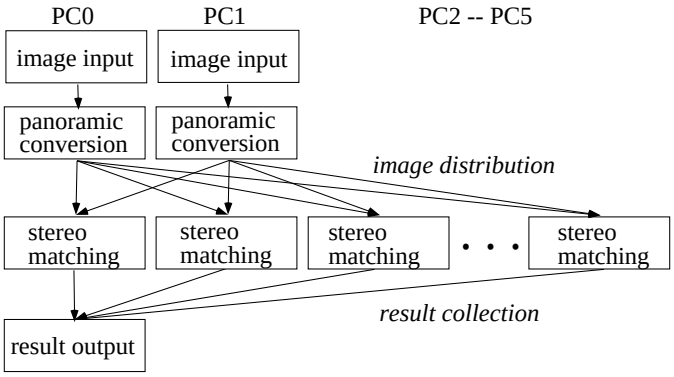


Fig. 5: PC cluster implementation.

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. 4 shows a result of matching, in which larger disparities (nearer points) are drawn in brighter color. This stereo system can measure up to about 40 meter.

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) real-time range data acquisition, we parallelize the third step using a 6-PC (Pentium III, 600MHz) cluster system as shown in Fig. 5. 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.

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

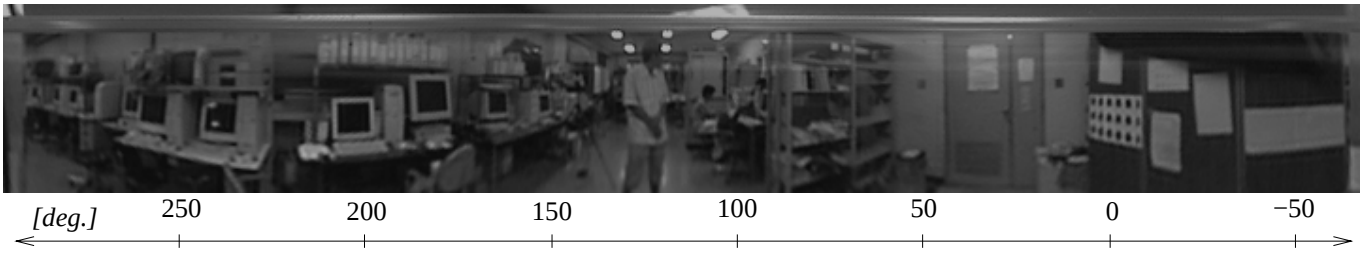


Fig. 3: Panoramic image obtained from the input image shown in Fig. 1.

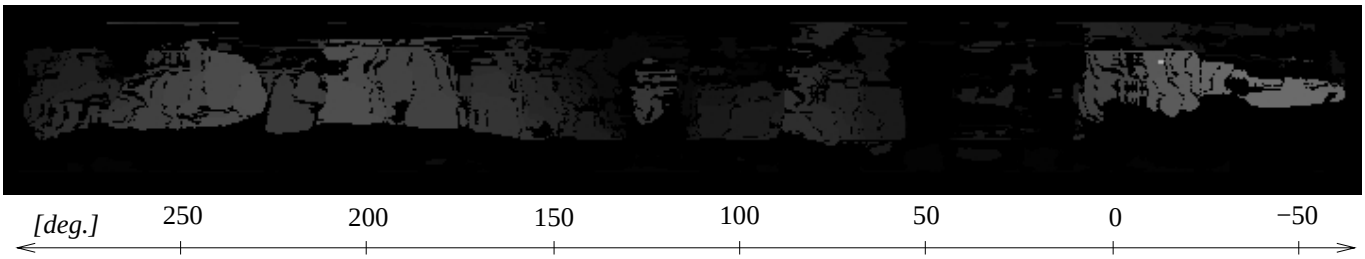


Fig. 4: Panoramic disparity image obtained from the images in Fig. 3. Brighter pixels are nearer.

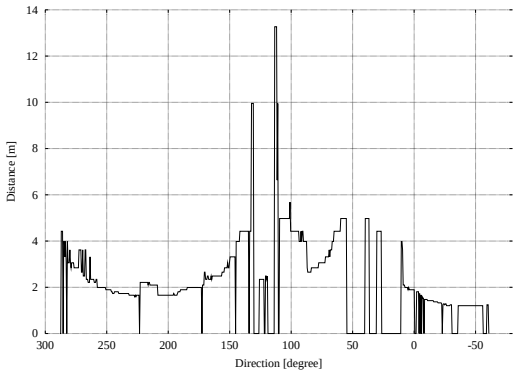


Fig. 6: Example range profile.

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 Obtaining Range Profile and Ego-Motion Estimation

3.1 Obtaining 2D Range Profile

To make a map of static obstacles and to adopt a visual ego-motion estimation method, 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 and extract the connected interval in which the disparity difference between (vertically) adjacent pixels is less than or equal to one and which is nearest among such intervals in the same column. By applying this operation to every column, we obtain a set of distance of 360 degrees. From this data set a 2D contour (called *range profile*) of the current free space centered at the robot position is obtained.

Fig. 6 shows the range profile obtained from the disparity data shown in Fig. 4. In Fig. 6, the horizontal axis represents the viewing direction from the robot and the vertical axis represents distance to obstacles. Actually, the direction ranges from $-60.747°$ to $287.974°$; we cannot obtain the range data for the directions corresponding to the blind spot caused by the pillar supporting the stereo camera pair. The resolution of the direction is about 0.5 degrees. Note that if no range data is obtained for a direction, the distance for the direction is set to zero.

3.2 Ego-Motion Estimation

The positional uncertainty accumulates when the robot position is estimated only by dead reckoning. Therefore we have to estimate the ego-motion of the robot between the viewpoints to integrate observation results at different viewpoints. Thus, we adopt the following vision-based ego-motion estimation method to reduce the uncertainty. This ego-motion estimation method is similar to [7], which is based on the comparison between the current and the previous range profile. In the method, we first compute the uncertainty of the current robot position to determine a set of possible robot positions. Next, we calculate the difference between the view of the current and the previous range profiles for each candidate pair of the position and the orientation. We use range profiles of previous k times for comparison to reduce the effects of moving obstacles and stereo matching errors (currently, k is 3). Finally, the pair of position and orientation which minimize the difference is selected as the current position and orientation.

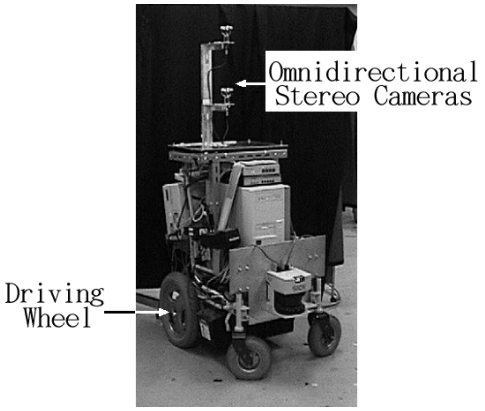


Fig. 7: Our mobile robot.

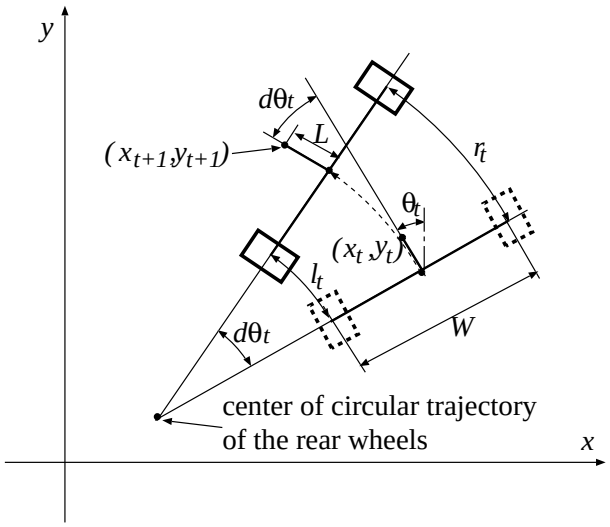


Fig. 8: Motion model of four wheeled mobile robot.

3.2.1 Uncertainty Model of Robot Motion

Fig. 7 shows our mobile robot which is based on a four-wheeled electric wheel chair driven by two rear wheels. The state of robot, $\boldsymbol{X} = (x, y, \theta)^T$, consists of the 2D robot position (x, y) which corresponds to the position of the camera pair, and the orientation of the robot, θ. Fig. 8 shows the motion model of the robot controlled by input $\boldsymbol{U} = (l, r)^T$ which is the moving distance of the left and the right wheels. The state transition of the robot is expressed by the following nonlinear equation:

$$\boldsymbol{X}_{t+1} = \begin{pmatrix} x_t + \frac{W}{2}\frac{l_t+r_t}{l_t-r_t}\left(\cos\theta_t - \cos(\theta_t - \frac{l_t-r_t}{W})\right) + L\left(\sin\theta_t - \sin(\theta_t - \frac{l_t-r_t}{W})\right) \\ y_t + \frac{W}{2}\frac{l_t+r_t}{l_t-r_t}\left(\sin\theta_t - \sin(\theta_t - \frac{l_t-r_t}{W})\right) - L\left(\cos\theta_t - \cos(\theta_t - \frac{l_t-r_t}{W})\right) \\ \theta_t - \frac{l_t-r_t}{W} \end{pmatrix} = \boldsymbol{F}(\boldsymbol{X}_t, \boldsymbol{U}_t), \quad (1)$$

where W is the distance between the two rear wheels; L is the distance between the robot position and the midpoint of the rear wheels.

Linearizing Eq. (1) by the first-order Taylor series expansion around the mean value, $\hat{\boldsymbol{X}}_t$ and $\hat{\boldsymbol{U}}_t$, the covariance matrix of the predicted state error, $\Sigma_{\boldsymbol{X}_{t+1}}$, can be obtained by:

$$\Sigma_{\boldsymbol{X}_{t+1}} = E[(\boldsymbol{X}_{t+1} - \hat{\boldsymbol{X}}_{t+1})(\boldsymbol{X}_{t+1} - \hat{\boldsymbol{X}}_{t+1})^T] = \frac{\partial \boldsymbol{F}(\boldsymbol{X}_t, \boldsymbol{U}_t)}{\partial \boldsymbol{U}_t} \Sigma_{\boldsymbol{X}_t} \frac{\partial \boldsymbol{F}(\boldsymbol{X}_t, \boldsymbol{U}_t)}{\partial \boldsymbol{U}_t}^T + \frac{\partial \boldsymbol{F}(\boldsymbol{X}_t, \boldsymbol{U}_t)}{\partial \boldsymbol{U}_t} \Sigma_{\boldsymbol{U}_t} \frac{\partial \boldsymbol{F}(\boldsymbol{X}_t, \boldsymbol{U}_t)}{\partial U_t}^T, \quad (2)$$

where $\Sigma_{\boldsymbol{U}_t}$ is the covariance matrix of the input $\boldsymbol{U}_t$. We assume that the error $\Sigma_{\boldsymbol{U}_t}$ is caused only by the slippage of wheels. We also assume that the error of the left and the right wheels, ${\sigma_{l_t}}^2$ and ${\sigma_{r_t}}^2$, are Gaussian and independent of each other. Thus, $\Sigma_{\boldsymbol{U}_t}$ is expressed by the following diagonal matrix:

$$\Sigma_{\boldsymbol{U}_t} = \begin{pmatrix} {\sigma_{l_t}}^2 & 0 \\ 0 & {\sigma_{r_t}}^2 \end{pmatrix}. \quad (3)$$

${\sigma_{l_t}}^2$ and ${\sigma_{r_t}}^2$ are considered to be proportional to the moving distance, l_t and r_t; we determine the proportional coefficients experimentally (currently, we use 0.3).

In this paper, we define the uncertainty region as the so-called 3σ ellipsoid obtained from $\Sigma_{\boldsymbol{X}_t}$. The positional uncertainty is represented as an ellipse generated by projecting the ellipsoid on the $X-Y$ plain. The orientational uncertainty is calculated as the marginal distribution of θ.

3.2.2 Comparing Range Profiles for Ego-Motion Estimation

Since we use a 2D grid map as described in Sec. 4, we do not need to consider the positional errors less than the grid size of the map. Therefore, candidates for the current robot position are grid points of the map inside the predicted uncertainty region. Candidates for the robot orientation are generated by discretizing the orientational uncertainty region with the angular resolution.

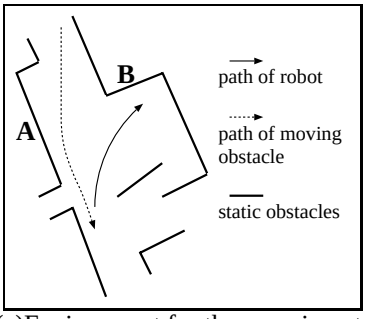
For each pair of candidate position and an orientation, we can compute the views of a previous range profile. By comparing such view of the previous three range profiles with the current range profile, we determine the current position and orientation which minimizes the *difference* between these range profiles.

The difference of range profiles is evaluated by:

$$Diff(i, \phi) = \sum_{j=1}^{k} \frac{1}{N(i, \phi, j)} \sum_{\theta=\theta_{min}}^{\theta_{max}} d(i, \phi, j, \theta), \quad (4)$$

$$d(i, \phi, j, \theta) = \begin{cases} \left| D_t(\theta) - D^i_{t-j}(\theta - \phi) \right| & (D_t \text{ and } D^i_{t-j} \text{ are obtained}) \\ 0 & (\text{otherwise}) \end{cases},$$

where i and ϕ represent the candidate position and orientation of the robot, respectively; $D(\theta)_t$ represents the disparity in direction θ at time t; $[\theta_{min}, \theta_{max}]$ represent the range of possible viewing directions (corresponding to the right and the left side of panoramic image); $N(i, \phi, j)$ indicates the number of data for which the difference of disparity is obtained. This equation calculates the sum of the average of absolute difference between range profiles. Notice thatwe do not compare distances but compare disparities in calculating the difference because the error of disparity is constant while that of distance increases as the distance increases. In dynamic environments, data corresponding to moving obstacles may degrade the ego-motion



(a)Environment for the experiment.


(b) with only dead reckoning.

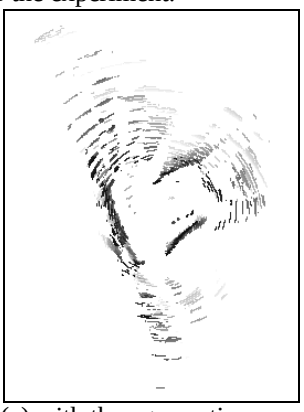

(c) with the ego-motion estimation.

Fig. 9: Effect of visual ego-motion estimation.

estimation. Currently, however, we use all range data in ego-motion estimation by assuming that moving obstacles are small enough in the panoramic image to be neglected in ego-motion estimation.

Using this equation, for each candidate position, the best orientation $\phi^*(i)$ is determined by:

$$\phi^*(i) = \arg\min_{\phi} Diff(i, \phi). \tag{5}$$

Then, the best position i^* is determined by:

$$i^* = \arg\min_{i} Diff(i, \phi^*(i)). \tag{6}$$

After i^* is determined, the positional uncertainty Σ_{X_t} is updated so that the corresponding uncertainty region circumscribe the selected grid. The orientation uncertainty set to the constant value (currently used $3\sigma = 1.5[deg.]$ which is determined experimentally.

3.2.3 Experiment of The Ego-motion Estimation

Fig. 9(a) shows the environment for an experiment of the ego-motion estimation. Fig. 9(b) and (c) show the superimposed position of obstacle regions obtained by 30 consecutive observations. In Fig. 9(b), the robot position is estimated only by dead reckoning, while in Fig. 9(c), the above visual ego-motion estimation method is used. In Fig. 9(c), walls **A** and **B** are clearly seen and the orientations of these walls are correct, although there is a moving obstacle which is seen in the figure. These results show the effectiveness of the ego-motion estimation method even in dynamic environments.

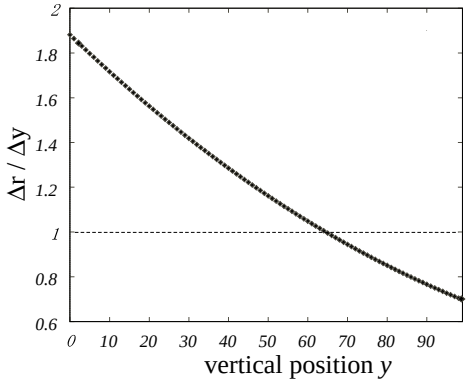


Fig. 10: Effect of image blur.

4 Making a Free Space Map Considering Vision Uncertainty

4.1 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:

$$R_{max} = \frac{Bf'}{d-1}, \qquad R_{min} = \frac{Bf'}{d+1}. \tag{7}$$

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:

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

where θ_u and θ_l are the viewing angle above and below 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. 10 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

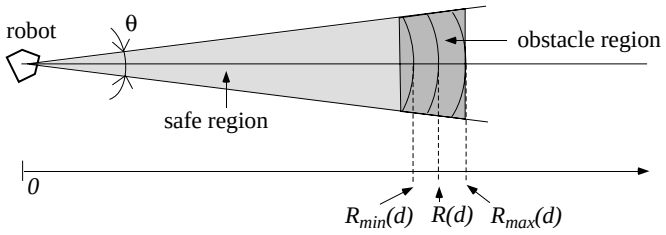


Fig. 11: Safe region and obstacle region.

the distance (see Eq. (7)) is increased accordingly. In the current configuration, for example, the range error at $1[m]$ is about $5[cm]$ and at $5[m]$ is about $1.2[m]$.

Using the above uncertainties, we interpret each point of disparity d in the range profile as follows. In Fig. 11, 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 in front of 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. 5.1).

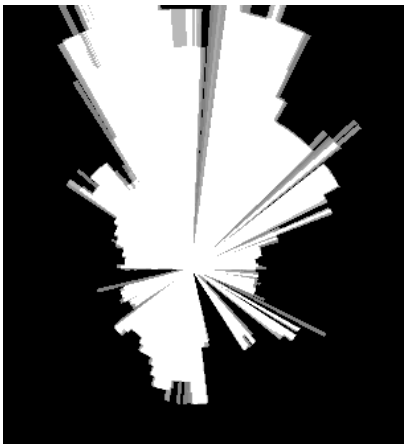


Fig. 12: An example map. White region indicates the free space; gray regions indicate the area where the observation count is less than the threshold.

4.2 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*).

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 calculated by the ego-motion estimation method described above. To reduce the effect of accumulated error when the robot moves by a long distance, we use only twelve latest observations for making the free space map, and then use five as the threshold for the observation counter mentioned above. Fig. 12 shows an example map. The grid size is currently $5[cm] \times 5[cm]$.

5 Detecting and Tracking Moving Obstacles

5.1 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. 11) 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.

5.2 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. 4.1). 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. Letting P, Q, and R be the covariance matrix of $\boldsymbol{x}$, $\boldsymbol{w}_t$, and $\boldsymbol{v}_t$, respectively, the following Kalman filter is derived:

$$
\begin{aligned}
\boldsymbol{x}_{t+1/t} &= F_t \boldsymbol{x}_{t/t}, \\
\boldsymbol{x}_{t/t} &= \boldsymbol{x}_{t/t-1} + K_t \left[\boldsymbol{y}_t - H_t \boldsymbol{x}_{t/t-1}\right], \\
K_t &= P_{t/t-1} H_t^T \left[H_t P_{t/t-1} H_t^T + R_t\right]^{-1}, \\
P_{t+1/t} &= F_t P_t F_t^T + Q_t, \\
P_{t/t} &= \left[I - K_t H_t\right] P_{t/t-1}, \\
\boldsymbol{x}_{0/-1} &= \boldsymbol{x}_0, \\
P_{0/-1} &= P_0,
\end{aligned}
$$

where K_t is the Kalman gain; $\boldsymbol{x}_0$ and P_0 are the initial position and the uncertainty of an obstacle, respectively; P_0 is equal to the vision uncertainty R_0 when the obstacle is first observed.

5.3 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:

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

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. 5.1), 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 20$[cm/s]$, that object is considered to be a part of some static obstacle.

6 Experimental Results

We performed experiments of tracking multiple walking persons by the mobile robot in our laboratory as shown in Fig. 13. The total processing time including omnidirectional stereo, ego-motion estimation, update of free space map, and dynamic obstacle detection and tracking is currently about 0.31$[s]$ per frame.

Fig. 13: Snapshot of an experiment.

Figs. 14 and 15 show the result of an experiment. In the experiment, one person walked from the front of the robot, another person walked from the back at same time, and they passed each other near the robot. The robot detected and tracked them as it moved on a pre-determined trajectory. Fig. 14 shows a sequence of the tracking result, in which trajectories of the tracked persons (labeled A and B), trajectory of the robot, moving obstacle candidates, and free space contours are drawn. Fig. 15(a)-(e) show the projection of the estimated person positions onto the panoramic image in the case of Fig. 14(a)-(e), respectively. The figures show that the tracking was correctly performed even when persons were overlapped. In Fig. 15(d), though the robot lost person A due to the blind spot, it successfully restarted tracking later as another person A'(see Fig. 14(e) and Fig. 15(e)). The proposed method is robust in complex environments, where there are various static obstacles such as chairs and desks.

7 Conclusion

We have developed a method of on-line detection and tracking of moving obstacles by a mobile robot using real-time omnidirectional stereo vision system. The method can robustly track multiple moving obstacles under unknown complex environments. It can also cope with the odometry error by an on-line ego-motion estimation method based on the comparison between range data. The experimental results show the validity of the method. A future work is to apply the system to mobile robot navigation. Another future work is to develop a compact on-robot system for experiments in much wider spaces.

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 Correspondnce. *Int. J. of Computer Vision*, 10(1):53–66, 1993.

[2] O. Faugeras et al. Real-Time Correlation-Based Stereo: Algorithm, Implementation and Application. Technical 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*, volume 1, pages 299–303, 1998.

[4] S. Kagami, K. Okada, M. Inaba, and H. Inoue. Design and Implementation of Onbody Real-time Depthmap Generation System. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 1441–1446, 2000.

[5] S.B. Kang and R. Szeliski. 3-D Scene Data Recovery Using Omnidirectional Multibaseline Stereo. *Int. J. of Computer Vision*, 25(2):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. of 6th Int. Conf. on Intelligent Autonomous System*, pages 620–627, 2000.

[8] M. Lindström and J.-O. Eklundh. Detecting and Tracking Moving Objects from a Mobile Platform using a Laser Range Scanner. In *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*, pages 1364–1369, 2001.

[9] F. Lu and E.E Milios. Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. In *IEEE Computer Vision and Pattern Recognition Conf.*, 1994.

[10] E. Prassler and J. Scholz. Tracking Multiple Moving Objects for Real-Time Robot Navigation. *Autonomous Robots*, 8(2):105–116, 2000.

[11] K. Yamazawa, Y. Yagi, and M. Yachida. Omnidirectional Imaging with Hyperboloidal Projection. In *Proc. of 1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1029–1034, 1993.

[12] J.Y. Zheng and S. Tsuji. Panoramic Representation of Scenes for Route Understanding. In *Proc. Int. Conf. on Pattern Recognition*, pages 161–167, 1990.

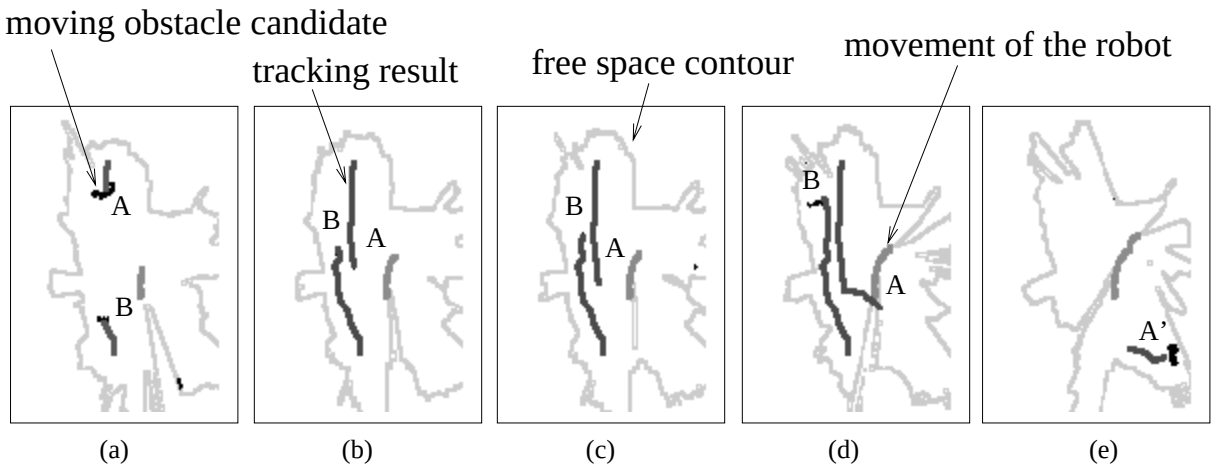


Fig. 14: A tracking result.

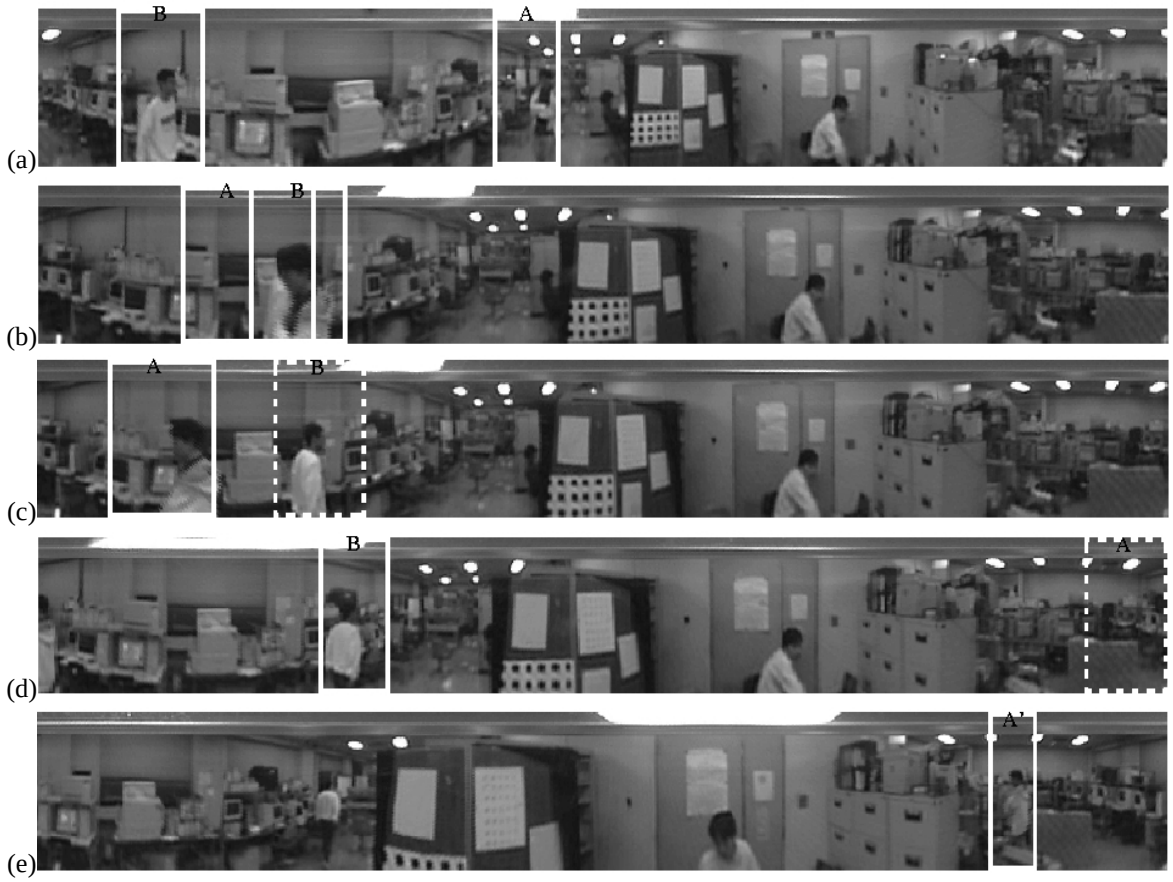


Fig. 15: Projection of the estimated person positions onto the panoramic image. A solid-line box represents an estimated person position when the corresponding observation is available at that frame, while a broken-line box is drawn when the corresponding observation is not available.