Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

Mobile Robot Navigation in Dynamic Environments using Omnidirectional Stereo

Hiroshi Koyasu, Jun Miura, and Yoshiaki Shirai
Department 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 mobile robot navigation method in dynamic environments. The method uses a real-time omnidirectional stereo which can obtain panoramic range information of 360 degrees. From this panoramic range information, the robot first estimates its ego-motion by comparing the current and the previous observations in order to integrate observations obtained at different positions. The uncertainty in the estimation is also calculated. Next, the robot recognizes and tracks moving obstacles. Finally, the robot plans a collision free path by a heuristic planner in space-time considering the velocity uncertainty of observed obstacles. Experimental results show the effectiveness of our method.

1 Introduction

Avoiding collision with moving obstacles is one of the important functions of mobile robots operating in dynamic environments. To avoid collision, the robot needs two abilities; one is recognizing dynamic environments, and the other is planning a collision free path.

Recognition of obstacles usually requires temporal integration of sensing data to cope with uncertainties of sensor data and changes of an environment. To integrate sensing data which is obtained from a moving observer, a reliable ego-motion estimation is indispensable. Since dead reckoning suffers from accumulated errors, an ego-motion estimation based on external sensors such as vision is needed.

For the ego-motion estimation or localization problem, many works use a feature-based matching (e.g., [1], [3]). Such methods depend on the existence of features in environments. Moreover, finding matches and solving a minimization problem require much computation.

Methods using featureless matching are also proposed. Lu et al. [8] proposed two ego-motion estimation methods using a laser range finder. Both methods are based on the correspondence between 2D contours obtained from the current and the previous range information. Laser range finders which scan a 2D plane have a drawback that objects at a specific height can only be detected. Moreover, since the methods compare only two scanned data, it may not be well applicable to the case where the uncertainty of range data is relatively large. Kidono et al. [6] proposed a method for estimating the best ego-motion which minimizes the difference between two consecutive range data obtained by stereo. Our previous paper [7] applied this method to an omnidirectional stereo to solve the problem of narrow field of view.

These featureless matching-based ego-motion estimation methods still have several problems. They only estimate the most probable ego-motion; this sometimes causes false range data matches in subsequent observations. Moreover, they seem sensitive to noise in range data because they use only two range data. This paper, therefore, proposes a new ego-motion estimation method which uses a sequence of range data for ego-motion estimation. The method also estimates the uncertainty of ego-motion. By estimating the uncertainty, the method can evaluate the reliability of each range data matching, thereby excluding unreliable matchings caused by moving obstacles or false stereo matches.

Once an ego-motion is estimated, the robot updates the free space map and detects moving obstacles. The candidates are then tracked by the Kalman filter. We use our previous method [7] for obstacle detection and tracking.

To plan a collision free path in dynamic environments, the robot must consider moving obstacles. Fiotini et. al.[4] proposed a velocity obstacle model, in which a collision possibility cone is calculated for the robot velocity; the method, however, does not consider the uncertainty of obstacle velocity. This paper employs a heuristic path planner which uses a space-time model with uncertainty [2].

We conducted navigation experiments with avoiding moving obstacle using our mobile robot (see Fig. 1). The robot is equipped with a omnidirectional stereo system. Fig. 2 shows an example panoramic disparity image, whose size is 720×100 and the disparity range is 80. The system can generate a disparity image in 0.2 [s] using a PC of dual-Athlon MP 2200+. Refer to [7] for the detail.

0-7803-7736-2/03/$17.00 ©2003 IEEE

Figure 2: Omnidirectional disparity image. Brighter pixels indicate larger disparities.

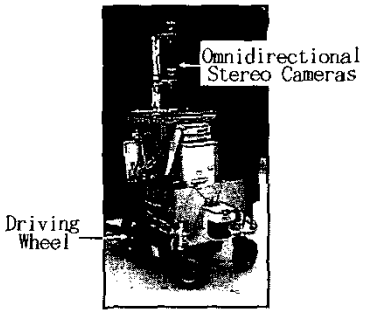


Figure 1: Our mobile robot.

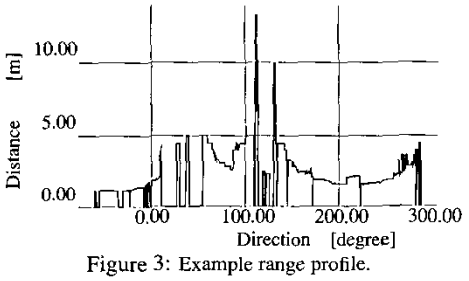


Figure 3: Example range profile.

2 Ego-Motion Estimation

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 data for each candidate pair of the position and the orientation. Finally, we determine the current position and orientation with their uncertainties by a weighted least square-based estimation.

2.1 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. From this data set, a 2D contour (called *range profile*) of the current free space centered at the robot position is obtained. Fig. 3 shows the range profile obtained from the disparity data shown in Fig. 2. In Fig. 3, the horizontal axis represents the viewing direction from the robot and the vertical axis represents distance to obstacles. 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.

2.2 Uncertainty Model of Robot Motion

The positional uncertainty increases as the robot moves due to slippage of wheels or a quantization error of odometry. We model the uncertainty by a three-dimensional normal distribution; the so-called 3σ ellipsoid obtained from the covariance matrix $\Sigma_{\boldsymbol{X}_t}$ represents the uncertainty region, where $\boldsymbol{X} = (x, y, \theta)$ is the robot's state. The positional uncertainty on (x, y) is calculated by projecting the ellipsoid on the x-y plane and the orientational uncertainty is calculated as its marginal distribution on θ. These uncertainties are used for predicting possible robot positions and orientations in ego-motion estimation.

2.3 Comparing Range Profiles

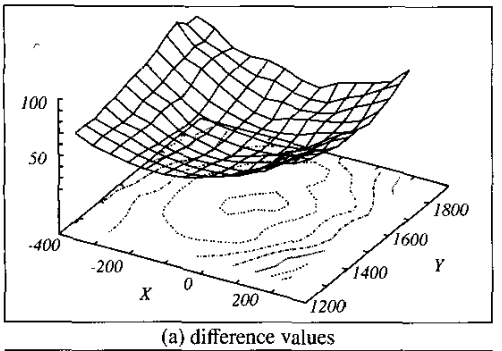
We sample at least nine candidate positions inside the predicted uncertainty region for the weighted least square-based estimation. If the distance between the neighboring candidates is larger than a threshold (currently, 10[*cm*]), the number of candidates is increased. Candidates for the robot orientation are also generated by discretizing the range of the orientational uncertainty with the angular resolution of the range profile.

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

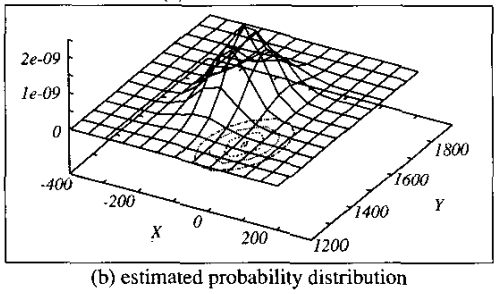
In a candidate position and orientation (x, y, ϕ), a difference of a disparity of direction θ between the current and the ith previous observation is calculated by:

$$d(x, y, \phi, i, \theta) = \frac{(D_t(\theta) - D_{t-i}^{(x,y)}(\theta - \phi))^2}{\sigma^2_{D_{t-i}^{(x,y)}(\theta-\phi)} + \sigma_o^2}, \quad (1)$$

where $D_t(\theta)$ represents the disparity in direction θ at time t; $\sigma^2_{D_{t-i}^{(x,y)}(\theta-\phi)}$ is the variance of the disparity $D_{t-i}^{(x,y)}(\theta - \phi)$; σ_o^2 is the variance of the observed disparity (currently, 1). $\sigma^2_{D_{t-i}^{(x,y)}(\theta-\phi)}$ is calculated by the positional error of time



(a) difference values



(b) estimated probability distribution

Figure 4: Estimation of the positional distribution.

$t-i$, the uncertainty of the disparity of the observation at $t-i$, and the motion uncertainty of the current frame. $d(x, y, \phi, i, \theta)$ represents the Mahalanobis distance; if $D_t(\theta)$ and $D_{t-i}^{(x,y)}(\theta - \phi)$ are from the same obstacle, d is assumed to follow a χ^2 distribution. Therefore, when d is larger than a certain threshold determined from the χ^2 distribution, or when $D_t(\theta)$ or $D_{t-i}^{(x,y)}(\theta - \phi)$ is not obtained, we do not take the corresponding direction into consideration. By this way, the effect of false matches in stereo and that of the moving obstacles can be reduced.

The difference of range profiles is then evaluated by:

$$Diff(x, y, \phi) = \sum_{i=1}^{k} \frac{1}{N(x, y, \phi, i)} \sum_{\theta=\theta_{min}}^{\theta_{max}} d(x, y, \phi, i, \theta), \quad (2)$$

where $[\theta_{min}, \theta_{max}]$ represents the range of possible viewing directions (corresponding to the right and the left end of panoramic image); $N(x, y, \phi, i)$ indicates the number of data for which the difference of disparity is obtained. This equation calculates the sum of the averaged squared difference between range profiles normalized by the uncertainty of the disparities. Notice that we do not compare distances but compare disparities in calculating the difference because the error of disparity is constant while that of distance is larger for a longer distance.

2.4 Estimating Ego-Motion

Our previous method [7] selected the robot position and orientation which minimizes the difference between range profiles. However, such a method often leads to an incorrect estimation of the position and orientation, particularly in an environment which does not have enough objects. For example, in a corridor, the reliability of the estimated position is high for the perpendicular direction to the corridor, while it is low to the direction along the corridor. In such a case, the estimated position along the corridor may be incorrect due to false matches caused by noises and moving objects. Fig. 4(a) shows a distribution of difference values *Diff* around the predicted position. As shown in the figure, the uncertainty distribution of the estimated position is considered to be ellipsoidal; the reliability of the estimation is low along the longer principal axis of the ellipsoid. We, therefore, estimate not only the robot position and orientation but also their uncertainty.

The probability distribution of the position and orientation can be estimated by the difference values around the estimated point. Nickels and Hutchinson [9] solved a similar problem for the SSD-based feature tracking; they estimated the uncertainty of the target localization in the image. To estimate the uncertainty, they first calculate the SSD values around predicted position, then convert them to *response distribution*, which is defined by Singh and Allen [10]. The response distribution calculates the confidence of each estimated position.

In our method, the response distribution is represented by the following:

$$r(x, y, \phi) = \exp(-\kappa Diff(x, y, \phi)), \quad (3)$$

where κ is used as a normalization factor, which is determined so that the following equation holds::

$$\kappa \min(Diff(x, y, \phi)) = c, \quad (4)$$

where c is constant (currently, 5).

Since the response distribution can be interpreted as a probability distribution of the position and orientation, the best position and orientation are determined by a weighted least squares method.

$$\begin{aligned} \hat{x} &= \frac{\sum_{x,y,\phi} r(x, y, \phi)x}{\sum_{x,y,\phi} r(x, y, \phi)} \\ \hat{y} &= \frac{\sum_{x,y,\phi} r(x, y, \phi)y}{\sum_{x,y,\phi} r(x, y, \phi)} \\ \hat{\phi} &= \frac{\sum_{x,y,\phi} r(x, y, \phi)\phi}{\sum_{x,y,\phi} r(x, y, \phi)} \end{aligned} \quad (5)$$

Under the assumption of additive zero mean independent errors, the estimated error covariance matrix is also calculated by:

$$\Sigma_{\boldsymbol{x}_t} = \begin{pmatrix} \frac{\sum_{x,y,\phi} r(x,y,\phi)(x-\hat{x})(x-\hat{x})}{\sum_{x,y,\phi} r(x,y,\phi)} & \frac{\sum_{x,y,\phi} r(x,y,\phi)(x-\hat{x})(y-\hat{y})}{\sum_{x,y,\phi} r(x,y,\phi)} & \frac{\sum_{x,y,\phi} r(x,y,\phi)(x-\hat{x})(\phi-\hat{\phi})}{\sum_{x,y,\phi} r(x,y,\phi)} \\ \frac{\sum_{x,y,\phi} r(x,y,\phi)(x-\hat{x})(y-\hat{y})}{\sum_{x,y,\phi} r(x,y,\phi)} & \frac{\sum_{x,y,\phi} r(x,y,\phi)(y-\hat{y})(y-\hat{y})}{\sum_{x,y,\phi} r(x,y,\phi)} & \frac{\sum_{x,y,\phi} r(x,y,\phi)(y-\hat{y})(\phi-\hat{\phi})}{\sum_{x,y,\phi} r(x,y,\phi)} \\ \frac{\sum_{x,y,\phi} r(x,y,\phi)(x-\hat{x})(\phi-\hat{\phi})}{\sum_{x,y,\phi} r(x,y,\phi)} & \frac{\sum_{x,y,\phi} r(x,y,\phi)(y-\hat{y})(\phi-\hat{\phi})}{\sum_{x,y,\phi} r(x,y,\phi)} & \frac{\sum_{x,y,\phi} r(x,y,\phi)(\phi-\hat{\phi})(\phi-\hat{\phi})}{\sum_{x,y,\phi} r(x,y,\phi)} \end{pmatrix}. \quad (6)$$

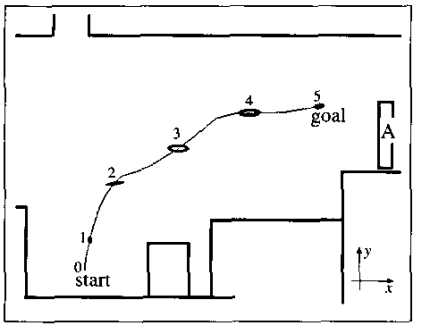


Figure 5: Experimental result of ego-motion estimation.

Fig. 4(b) shows the probability distribution calculated by the covariance matrix. The contour of the probability distribution matches well to that of the original difference distribution. This covariance matrix is used to calculate $\sigma_{D_{t-i}^{(x,y)}(\theta-\phi)}$ in Eq. (1).

2.5 Result of Ego-Motion Estimation

Fig. 5 shows a result of ego-motion estimation. In the figure, black lines represent boundaries of static obstacles in the environment, dark gray line represents the estimated trajectory of the robot, and bright gray ellipses represent the estimated uncertainty of each position. In Fig. 5, at the points 2 – 4, the uncertainty along the x axis was large because the robot could not obtain reliable range information for the x direction. Then, at the goal (point 5), the uncertainty along x axis became smaller because wall A was near to the robot.

3 Recognizing Dynamic Environment

This section briefly describes methods of generating a free space map and detecting and tracking moving obstacles. Please refer to [7] for the detail.

3.1 Making a Free Space Map

A free space map is generated by temporal integration of range data. For the range measurement in one direction, the region before the estimated range is interpreted as a *safe region* and the region near the estimated range as an *obstacle region*. Safe regions are used for making a map of static obstacles, while obstacle regions are for detecting moving obstacle candidates.

Each grid of the map holds a counter which indicates how many times the grid has been observed as a safe region. If the counter value of a grid is higher than a certain threshold (currently five), the grid is considered free. The set of free grids constitutes the current free space. Fig. 6 shows an example map.

3.2 Detecting and Tracking Moving Obstacles

If a point in the current 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. We consider a merged group of such points as a candidate for moving obstacle and use their mass center as its observed position. Each candidate is tracked using the Kalman filter [5]; the filter outputs the position and the velocity of each obstacle and their uncertainties.

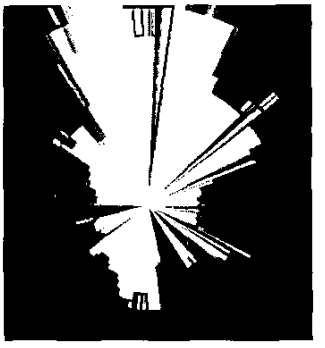


Figure 6: An example map. The white region indicates the free space; gray regions indicate the areas where observation counts are less than the threshold; black regions indicate the area where the observation is never counted.

4 Path Planning

Our path planning method is based on a *space-time* search to cope with moving obstacles. The method considers the velocity uncertainty of obstacles in planning. To determine the path towards a destination, we use a heuristic path planner. If the destination is in the free space of the map, the robot use it for path planning. Otherwise, the robot selects a temporary destination (a via point) which is in the free space and nearest to the given destination, and used it for path planning.

First, the planner generates a circular path which connects the current position and the destination and whose tangent line at the current position is the same as the current orientation of the robot. If the path is judged to lead the robot to a collision with a (static or dynamic) obstacle, the planner searches for a path to avoid it.

Currently our robot moves at a constant speed. This setup simplifies the path planning.

4.1 Avoiding Collision with a Static Obstacle

Fig. 7 illustrates the process of generating a path avoiding a collision with a static obstacle. In the first path candidate (arc $P_0V_0G_0$ in Fig. 7), the planner selects a point which is farthest from the free space (V_0 is selected) and draws a line perpendicular to the tangent line of the circular path there, and selects a via point (G_1) on the line in the free space which is nearest to the point (V_0). For this via point, the planner repeats the same operation until a safe

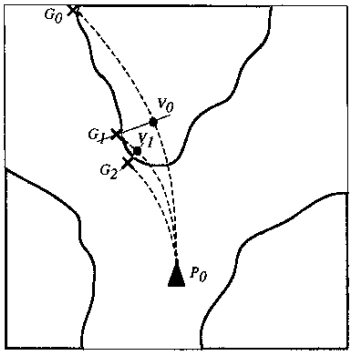


Figure 7: Avoiding static obstacles. Gray regions indicate obstacles.

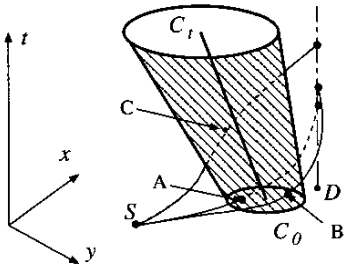


Figure 8: Avoiding moving obstacles.

circular path is found (try arc $P_0V_1G_1$, select G_2, and find P_0G_2). If a path is found but the endpoint of the path is not the original destination, this process is iterated with the selected via point (G_2) being the initial position and the original destination (G_0) as the destination.

4.2 Avoiding Collision with a Dynamic Obstacle

Fig. 4 illustrates the process of generating a path to avoid a moving obstacle. Let $x_0 = (x_0, y_0, \dot{x}_0, \dot{y}_0)$ be the obstacle state (position and velocity) at the current time. We assume every obstacle has a circular section, and expand it by the robot's approximate radius; C_0 in the figure thus indicates the initial (expanded) obstacle (its radius is r), S indicates the position of the (point) robot, and D indicates the destination.

We model the acceleration of a moving obstacle by an isotropic normal distribution. Let u be the so-called 3σ radius of the distribution; then the region where the obstacle may exist at time t is represented by:

$$(x - (x_0 + \dot{x}_0 t))^2 + (y - (y_0 + \dot{y}_0 t))^2 - (r + ut)^2 < 0, \quad (7)$$

which is indicated as C_t in Fig. 4. The boundary of the time-evolving regions form a leaned cone in space-time. The robot must plan a path which never enter the cone. The planner first generates a circular path to the destination (Fig. 4 A). If the path penetrates the cone, the robot plans a collision-avoiding path, which is as near to the first circular path as possible. So the planner calculates two tangent circular path from its initial position on both side of the cone, and one of the tangent points is used as a via point.

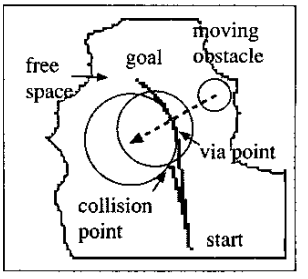


Figure 9: Result of path planning.

Since the robot moves at a constant speed, a robot path in (x, y, t) space is represented by:

$$\begin{cases} x = r\cos(\frac{v}{r}t + \theta_r) + x_r - r\cos\theta_r \\ y = r\sin(\frac{v}{r}t + \theta_r) + y_r - r\sin\theta_r \end{cases}, \quad (8)$$

where x_r, y_r and θ_r represent the current robot position and orientation, respectively, r represents the radius of a circular path, v represents the robot's speed. Our robot cannot change the turning radius continuously but can use only a set of radii due to a hardware limitation. The planner thus examines all of the radii to select the nearest one to the ideal (tangent) circular path. This is done by searching for the radius whose minimum value of the left side of the inequality (7) is positive and smaller than those of other radii on both side of the cone. On each circular path obtained from one of the selected radius, the nearest point to the cone is selected as a via point (points B and C in Fig. 4). Then the planner generates two paths by calculating a further path from each via point to the destination, and selects the shorter one.

4.3 Path Planning Example

Fig. 9 shows the result of a path planning simulation. In the figure, the gray region represents the free space which the robot recognizes, the broken arrow represents the initial position and the moving direction of a moving obstacle, black lines represent calculated paths, circles are projections of collision-possible regions in space-time on the X–Y plane. In the simulation, the robot could not generate a path to go to the goal directly, so the robot found a via point and generated a path through it.

5 Experiment

We performed experiments of navigation in dynamic environments. The total processing time is currently about 0.5 seconds using a dual-Athlon MP 2200+ PC, in which the ego-motion estimation takes about 0.3 seconds.

Fig. 10 shows the result of an experiment in which the robot and a person pass each other. Speed of the robot is

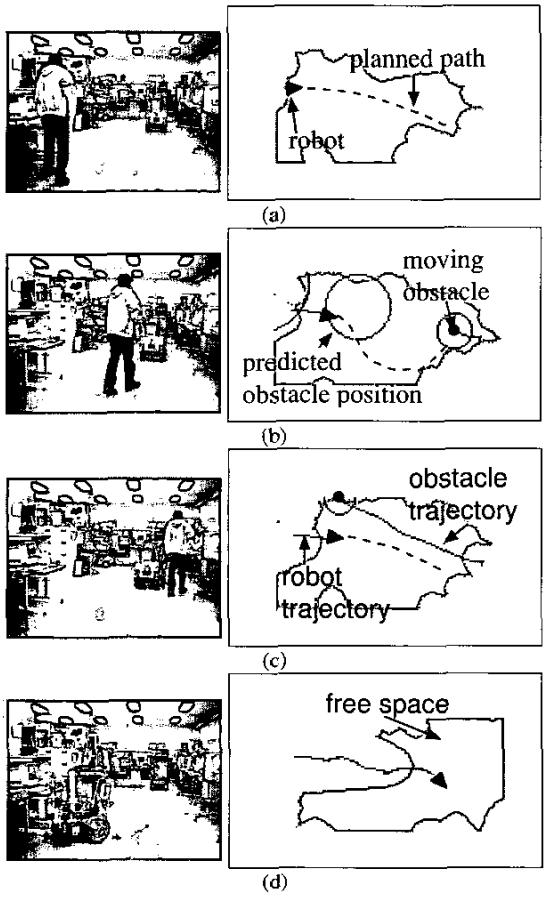


Figure 10: An experimental result.

about $0.3[m/s]$ and that of the person is about $0.8[m/s]$. The left side of the figure shows snapshots of the experiment. The right side shows recognition results of both static and dynamic obstacles and planned path.

In the experiment, first, the robot planned a circular path to the destination (see Fig. 10(a)). Then, the robot detected a person approaching the robot. Since the position of the person was far from the robot and speed of the person was very larger than that of the robot, the uncertainty of the person position was large at the time of possible collision. Thus the robot planned a path to avoid collision with a small turning radius (see Fig. 10(b)). After some time passed, the uncertainty of the person decreased, then the robot replanned a circular path to the destination (see Fig. 10(c)) and finally arrived there (see Fig. 10(d)).

6 Conclusion

We have developed a mobile robot navigation method for dynamic environments using omnidirectional stereo. To recognize environments reliably, the robot employs an ego-motion estimation method which does not depend on any features in environments and also estimates its uncertainty. After recognizing and tracking moving obstacle using estimated ego-motion, the robot plans a collision free path by a heuristic planner in space-time considering the velocity uncertainty of observed obstacles. Experiments of navigation in dynamic environments were performed to show the effectiveness of our method.

Currently, the robot moves at a constant speed. This constraint may lead to an inefficient path. For example, when an obstacle crosses the robot's path, it may be more efficient for the robot to stop and wait for the obstacle passing by than to follow an avoiding path. A future work is to improve the path planner so that it can consider the change of robot speed. Another future work is to reduce the processing time, especially that for ego-motion estimation, in order to increase the reactiveness to moving obstacles.

References

[1] G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A Computationally Efficient Solution to the Simultaneous Localization and Map Building (SLAM) Problem. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 1009–1014, 2000.

[2] A. Elnagar and A. Basu. Local Path Planning in Dynamic Environments with Uncertainty. In *Proc. of IEEE Int. Conf. on Multisencor Fusion and Integration for Intelligent Systems*, pages 183–190, 1994.

[3] M. Etoh, T. Aoki, and K. Hata. Estimation of Structure and Motion Parameters for a Roaming Robot that Scans the Space. In *Proc. of 7th Int. Conf. on Computer Vision*, volume 1, pages 579–584, 1999.

[4] P. Fiorini and Z. Shiller. Motion Planning in Dynamic Environments using Velocity Obstacles. *Int. J. of Robotics Research*, 17(7):760–772, 1998.

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

[6] K. Kidono, J. Miura, and Y. Shirai. Autonomous visual navigation of a mobile robot using a human-guided experience. *Robotics and Autonomous Systems*, 40(2-3):121–130, 2002.

[7] H. Koyasu, J. Miura, and Y. Shirai. Realtime omnidirectional stereo for obstacle detection and tracking in dynamic environments. In *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*, pages 31–36, 2001.

[8] F. Lu and E. Milios. Robot Pose Estimation in Unknown Environments by matching 2D Range Scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1997.

[9] K. Nickels and S. Hutchinson. Estimating uncertainty in SSD-based feature tracking. *Image and Vision Computing*, 20:47–58, 2002.

[10] A. Singh and P. Allen. Image-Flow Computation: An Estimation-Theoretic Framework and a Unified Perspective. *Computer Vision Graphics and Image Processing: Image Understanding*, 56(2):152–177, 1992.