The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Dynamic Path Planning of Mobile Robot Mounted Range Sensors and Single CCD Camera

Satoru Takahashi, *Member, IEEE* and Shunsuke Nara

***Abstract*—This paper treats a trajectory tracking control of a mobile robot by a single CCD camera and range sensors. First, the mobile robot detects the obstacle based on information of both the image and the distance. And then, from the obstacle information, the mobile robot generates a trajectory to avoid the obstacle. Finally, the usefulness of our proposed methods is demonstrated through experiment.**

I. Introduction

In recent years, mobile robots have been under intensive research and development for applications in conveyance and security services [6], [7]. Most mobile robots in these fields are based on the prerequisites that the environmental map information of the area of movement is available in advance and that the environment has been set up. In rescuing, space activities, or car steering, a mobile robot is expected to recognize the environment and autonomously drive itself since it is difficult to obtain environmental information in the vicinity of the mobile robot in advance. In other words, a mobile robot should obtain environmental information in a time series and avoid obstacles autonomously. In here, "obstacle" refers to a fixed object in environment. Since a collision between the mobile robot and the obstacle has the potential to damage both the mobile robot and the obstacle, it is essential to establish methods to allow a mobile robot to avoid obstacles autonomously.

In this paper, we establish the method of obstacle avoidance control for a mobile robot to reach a destination in an unknown environment containing obstacles by detecting the obstacles based on image information [4], [8] from a single CCD camera and distance information from range sensors and by then generating avoidance trajectories.

Image data is effective for detecting obstacles and various methods for recognizing obstacles from images have been proposed [2], [3], [5]. Since it is difficult to obtain depth information from a single CCD camera only, distance information from range sensors is used in combination with image data to enable the mobile robot to determine its positional relationship with an obstacle. We also give control to the mobile robot so that it can generate trajectories in a time series from the obtained obstacle position information and can follow the trajectories to avoid the obstacle safely without the collision.

Section II explains the experimental robot used in this research and Section III describes a method for detecting an obstacle by using a CCD camera and range sensors and presents a method for controlling a mobile robot so that it avoids obstacles. Section IV validates the proposed methods by experiments. Finally, Section V gives a brief conclusion.

S. Takahashi is with the Department of Intelligent Mechanical Systems Engineering, Kagawa University, 2217-20, Hayashi-Cho, Takamatsu-City, 761-0396, Japan (corresponding author to provide phone: +81-87-864-2329; fax:+81-87-864-2329; e-mail: saru@ eng.kagawa-u.ac.jp).

S. Nara is with the Designing Division, Nara Electric Heavy Industries, Ltd., 579, Kami-Cho, Takamatsu, Kagawa 761-8055, Japan (e-mail: n-shun@mail.netwave.or.jp).

II. System Configuration and Kinematics of the Mobile Robot

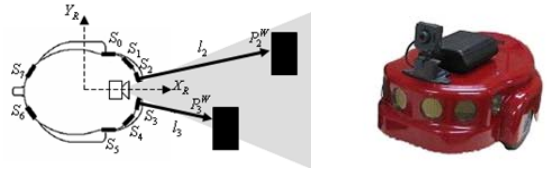
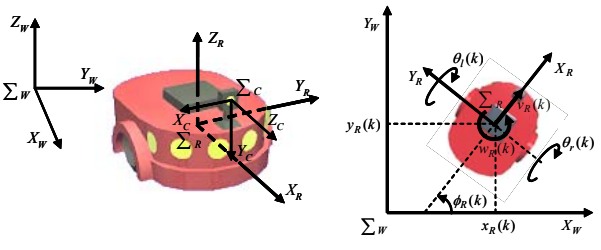


Fig. 1. The positions of range sensors and camera on Amigobot.

In this paper, a mobile robot which is called Amigobot (Active Media Robotics) as shown in Fig. 1 is used. Fig. 1 denotes the arrangement of a single CCD camera and eight range sensors on the robot. The range sensors are denoted as S_0 to S_7. This mobile robot, consisting of a single CCD camera, eight range sensors, and two independent drive wheels with an encoder, is controlled by a notebook computer with Windows-XP through the wireless LAN.



(a) (b)

Fig. 2. Relationship between the mobile robot and the world coordinates system. (a) The world coordinate and the camera coordinate. (b) The world coordinate and the robot coordinate.

Fig. 2 shows the coordinate system of the robot. The symbol $\sum_W$ denotes the world coordinate system that is defined relative to the floor on which the robot moves. The floor is generally free of irregularities and the X_WY_W-plane of the world coordinate system $\sum_W$ is assumed to be the floor. As shown in Fig. 2(b), a coordinate system with the wheel center of the robot at the origin is defined as the robot coordinate

system $\sum_R$. The robot advances in the X_R-axis direction. The $X_W Y_W$-plane of the world coordinate system $\sum_W$ and the $X_R Y_R$-plane of the robot coordinate system $\sum_R$ are parallel to each other. At the initial position of the robot, the origins of the world coordinate system $\sum_W$ and the robot coordinate system $\sum_R$ are identical. In here, the component of the Z_W-axis is not considered. Fig. 2(a) shows the camera coordinate system $\sum_C$ with the center of the CCD camera lens at the origin and the Z_C-axis is identical to the optical axis of the camera.

The method of the robot localization is based on kinematics as follows. This method is general for the right-and-left independent wheel system. The rotational angles (deg) of the right and left wheels which obtained from the decoder are denoted as $\theta_l(k)$ and $\theta_r(k)$, respectively, see Fig. 2(b). If the translational velocity of the robot is denoted as $v_R(k)$ and the turning velocity as $w_R(k)$, their relationship can be written by

$$\begin{bmatrix} v_R(k+1) \\ w_R(k+1) \end{bmatrix} = \begin{bmatrix} \frac{a}{2} & \frac{a}{2} \\ \frac{a}{b} & -\frac{a}{b} \end{bmatrix} \begin{bmatrix} \hat{\theta}_r(k+1) \\ \hat{\theta}_l(k+1) \end{bmatrix} \tag{1}$$

$$\hat{\theta}_r(k+1) = \theta_r(k+1) - \theta_r(k), \quad \hat{\theta}_l(k+1) = \theta_l(k+1) - \theta_l(k)$$

where a denotes the wheel of the radius (mm) and b denotes the distance between the right and left wheels (mm). Let k present discrete time. When time k is 0, the initial rotational angles of the right and left wheels and the initial translational velocity and turning velocity of the robot are both 0.

The self-position of the robot is calculated by integrating the position and rotation of the robot. If the coordinate position of the robot in a discrete time is $(x_R(k), y_R(k))$ and the directional angle (deg) is $\phi_R(k)$, their values in discrete time can be derived using the following (2). The angle $\phi_R(k)$ is formed by the world coordinate system $\sum_W$ and the robot coordinate system $\sum_R$ shown in Fig. 2(b).

$$\begin{aligned} \phi_R(k+1) &= \phi_R(k) + w_R(k+1) \\ x_R(k+1) &= x_R(k) + v_R(k+1) \cdot \cos\phi_R(k+1) \\ y_R(k+1) &= y_R(k) + v_R(k+1) \cdot \sin\phi_R(k+1) \end{aligned} \tag{2}$$

where the initial directional position of the robot is determined as $(x_R(0), y_R(0)) = (0, 0)$, and the initial directional angle $\phi_R(0)$ is 0.

III. Method of Dynamic Path Planning

This section proposes an obstacle avoidance method for the robot based on image data from the CCD camera and distance information from the range sensors.

A. Obstacle Detection

First, we show a method for detecting the candidate areas of obstacles. Through this paper, suppose that the environment of the robot has only fixed obstacles, that is, there are no moving obstacles in the environment. Then, optical flow is applied to detect obstacles. However, for the optical flow detection, various methods have been proposed [2], [3]. In this paper, we use the block matching method. The reason why we choose the block matching method is that this method can calculate the displacement of each small area on the image using the template matching. Therefore, we can manage the processing time to obtain optical flow. Further, by considering the detection accuracy, the block matching method based on normalized correlation is applied.

For the optical flow, two continuous images ρ_k and ρ_{k+1} are used. The collation area matching the template area around the pixel (i, j) on the image ρ_k is calculated for the image ρ_{k+1}. The template area and the collation area are $(2L+1)\times(2L+1)$ (pixel×pixel) in size. L is determined appropriately. Then, the correlation $E(i', j')$ between the template area in the image ρ_k and the collation area with the pixel component (i', j') at the center of the image ρ_{k+1} is calculated using (3) and (4) below:

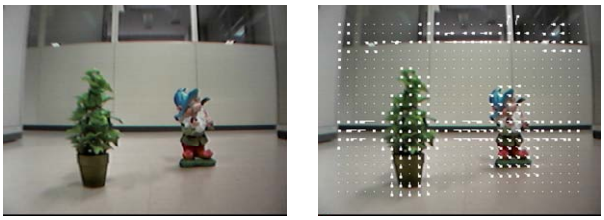
$$E(i', j') = \frac{\sum_{y=-L}^{y=L} \sum_{y=-L}^{y=L} (A_{xy} \cdot B_{xy})}{\sqrt{\sum_{y=-L}^{y=L} \sum_{y=-L}^{y=L} A_{xy}^2 \cdot \sum_{y=-L}^{y=L} \sum_{y=-L}^{y=L} B_{xy}^2}} \tag{3}$$

$$\begin{aligned} A_{xy} &= \rho_k(i+x, j+y) - \overline{\rho}_k \\ B_{xy} &= \rho_{k+1}(i'+x, j'+y) - \overline{\rho}_{k+1} \end{aligned} \tag{4}$$

where $\overline{\rho}_k$ and $\overline{\rho}_{k+1}$ denote the average intensity in each area. The optical flow vector (u_{ij}, v_{ij}) at matching is obtained by

$$u_{ij} = i' - i, \quad v_{ij} = j' - j \,. \tag{5}$$

Fig. 3 denotes the results of detecting the optical flow that is obtained when the robot advances. Fig. 3(a) shows the input image. The dotted lines in Fig. 3(b) indicate the pixel areas of the detected optical flows.



(a) (b)

Fig. 3. Optical Flow. (a) Input image. (b) Detected optical flow.

Then, the obstacle areas can be detected by using the optical flows. Considering that the optical flows are oriented in the same direction, the obstacle areas are classified by the directional components of these flows. The directional component ψ_{ij} and the strength component of the optical flow vector r_{ij} obtained by (5) are calculated by (6). From the

values obtained, the candidate areas of obstacles are decided.

$$\psi_{ij} = \tan\frac{v_{ij}}{u_{ij}}, \quad r_{ij} = \sqrt{u_{ij}^2 + v_{ij}^2} \tag{6}$$

where ψ_{ij} is between 0 and 2π.

For the area classification by the directional component of the optical flow, an image is divided into blocks of $(2H+1)\times(2H+1)$ (pixel×pixel). H may be determined appropriately. Then, the standard deviation σ_{ij} of the directional component in each block area is calculated in order to do the area classification by the directional component, clearly.

$$\sigma_{ij} = \frac{1}{H^2}\sum_{y=-H}^{y=H}\sum_{y=-H}^{y=H}(\psi_{ij} - \overline{\psi}_{ij})^2 \tag{7}$$

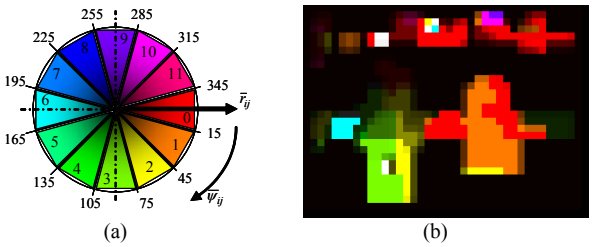
where $\overline{\psi}_{ij}$ denotes the average directional component in each block area that is calculated by (8).

$$\overline{\psi}_{ij} = \frac{1}{H^2}\sum_{y=-H}^{y=H}\sum_{y=-H}^{y=H}\psi_{ij} \tag{8}$$

If the standard deviation σ_{ij} is smaller than the threshold Th_σ in a block area, the optical flows in the area are regarded as generally in the same direction and the average direction obtained by (8) is given to the area. To regard differences in fine directions as identical, the directional component of 0 to 2π is quantized by the number of divisions to calculate c_{ij}. The quantized value c_{ij} is given by the following (9).

$$c_{ij} = \left[\frac{\psi_{ij}}{\Delta\psi}\right] \tag{9}$$

where c_{ij} is an integral value from 0 to D-1 and $\Delta\psi$ denotes the quantized width ($\Delta\psi = 2\pi / D$). If the standard deviation σ_{ij} is greater than the threshold Th_σ in a block area, the optical flows in the area are considered to be diverging and data is given to D that there is no optical flow direction in the area. As shown in Fig. 4(a), the direction of 360 degrees is divided and colored into 12 directional components. When the directional coloring is performed to Fig. 3(b), the image of Fig. 4(b) is obtained.



(a) (b)

Fig. 4. Classification by direction. (a) Displayed color. (b) Direction image.

For an image colored as shown in Fig. 4(b), areas having the same directional component are labeled. Among the labeled areas, ones that have more pixels or greater areas or are equivalent to the threshold Th_a are detected as candidate areas where obstacles exist. Fig. 5 shows the image of Fig. 4(b) after the threshold processing. The square areas indicate the candidate areas of obstacles.

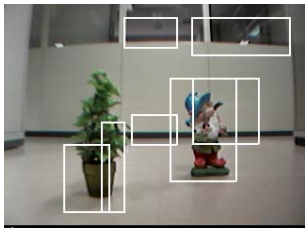


Fig. 5. Candidate areas of obstacles.

It follows from Fig. 5 that even the floor and high objects are detected as the candidate areas of obstacles. Therefore, it says that it is difficult to know the positional relationship between the robot and the obstacles from the image information, only. Considering this difficulty, we identify the true obstacle areas ahead of the robot from the candidate areas of obstacles by adding the distance information obtained by the range sensors. Namely, the image information and the distance information are combined for more accurate obstacle detection. For obstacle detection in this paper, we use the range sensors S_2 and S_3 installed on the front of the robot, as illustrated in Fig. 6, to obtain distance information within the field of vision of the CCD camera. If the distance information $l_n(k)$ from the range sensors is applied, the reference point in the robot coordinate system $\sum_R$ can be presented as (10).

$$\begin{bmatrix} {}^R x_n^p(k) \\ {}^R y_n^p(k) \end{bmatrix} = \begin{bmatrix} x_n^s \\ y_n^s \end{bmatrix} + l_n(k)\cdot\begin{bmatrix} \cos\phi_n^s \\ \sin\phi_n^s \end{bmatrix} \tag{10}$$

where x_n^s, y_n^s, and ϕ_n^s denote the position coordinates and the directional angle of each sensor in the robot coordinate system $\sum_R$, where n=2, 3. The range sensors have directivity, shown in Fig. 6, and the distance to an obstacle point overlapping in the sensitivity areas of the range sensors is measured.

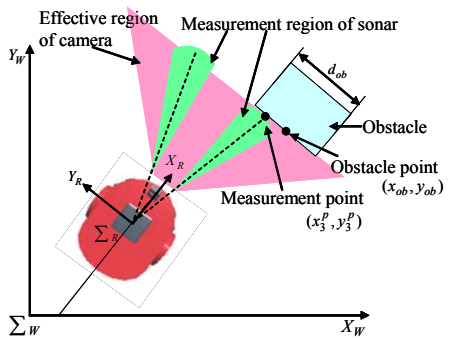


Fig. 6. Measurement position by range sensor.

Image information and distance information are integrated for the position detection of obstacle. For this integration, distance information is set so that it corresponds to image information obtained in advance. Further, the position information at the measurement point derived by (10) is converted into the camera coordinate system based on (11).

$$\begin{bmatrix} {}^{C}x_n^p(k) \\ {}^{C}y_n^p(k) \\ {}^{C}z_n^p(k) \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi_y\cos\phi_z & \cos\phi_y\sin\phi_z & \sin\phi_y & x_c \\ \sin\phi_z & \cos\phi_z & 0 & y_c \\ -\sin\phi_y\cos\phi_z & \sin\phi_y\sin\phi_z & \cos\phi_z & z_c \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} {}^{R}x_n^p(k) \\ {}^{R}y_n^p(k) \\ {}^{R}z_n^p(k) \\ 1 \end{bmatrix} \quad (11)$$

where let (x_c, y_c, z_c) indicate the CCD camera position in the robot coordinate system $\sum_R$, then $\phi_y = \pi/4$ (deg) and $\phi_z = -\pi/4$ (deg). The depth-wise distance ${}^{C}z_n^p(k)$ to the obstacle is obtained by (11) and the width x_n^C and the height y_n^C of the image plane, where the measurement point of the range sensor is derived by the following (12).

$$x_n^C = {}^{C}z_n^p(k)\cdot\frac{CAM_x}{f}, \quad y_n^C = {}^{C}z_n^p(k)\cdot\frac{CAM_y}{f} \quad (12)$$

where CAM_x and CAM_y denote the width (mm) of the CCD camera imaging element and the height f (mm) denotes the focal distance. Then, the measurement point $u_n^p(k)$, $v_n^p(k)$ of the range sensor can be calculated by (13).

$$u_n^p(k) = X_{SIZE}\cdot\left(\frac{x_n^p(k)}{x_n^C(k)}+\frac{1}{2}\right), \quad v_n^p(k) = Y_{SIZE}\cdot\left(\frac{y_n^p(k)}{y_n^C(k)}+\frac{1}{2}\right) \quad (13)$$

where X_{SIZE} and Y_{SIZE} denote the image size, in here let X_{SIZE} be 320 pixels and Y_{SIZE} be 240 pixels.

By considering the range sensor directive angle θ_{sd} (deg), the measurement region of the range sensors are set corresponding to the next image plane. The range sensor measurement region $W_n^p(k)$ determined from the directive angle can be calculated by (14). Fig. 7(a) shows an experimental image in the measurement regions of the range sensors.

$$W_n^p(k) = \frac{X_{SIZE}}{x_n^C(k)}\cdot l_n(k)\cdot\tan\theta_{sd} \quad (14)$$

(a) (b)

Fig. 7. Detection of obstacles. (a) Measurement regions by range sensors. (b) Detected obstacles.

Finally, the image of obstacles areas in Fig. 5 and the image of measurement regions in Fig. 7(a) are combined to identify and detect the real obstacles. Namely, obstacles are detected where the candidate areas of obstacles and the range sensor measurement regions overlap in the combined image. Fig. 7(b) shows the detected areas of obstacles. Then, the central position (u_{ob}, v_{ob}) and the width im_{ob} of the obstacle on the image are obtained, and the obstacle position $({}^{W}x_{ob}, {}^{W}y_{ob})$ in the world coordinate system $\sum_W$ and the obstacle size d_{ob} are derived by the followings (15) and (16).

$$\begin{bmatrix} {}^{W}x_{ob} \\ {}^{W}y_{ob} \end{bmatrix} = \begin{bmatrix} x_R(k) \\ y_R(k) \end{bmatrix} + \begin{bmatrix} \cos\phi_R(k) & \sin\phi_R(k) \\ -\sin\phi_R(k) & \cos\phi_R(k) \end{bmatrix} \begin{bmatrix} {}^{R}x_{ob} \\ {}^{R}y_{ob} \end{bmatrix} \quad (15)$$

$$d_{ob} = im_{ob}\cdot\frac{x_n^C}{X_{SIZE}} \quad (16)$$

where $({}^{R}x_{ob}, {}^{R}y_{ob})$ in (15) denotes the obstacle position in the robot coordinate system $\sum_R$ that is calculated by (17).

$$\begin{aligned} {}^{R}x_{ob} &= {}^{R}x_n^p(k) \\ {}^{R}y_{ob} &= (u_{ob} - \frac{X_{SIZE}}{2})\cdot\frac{x_n^C}{X_{SIZE}} \end{aligned} \quad (17)$$

The obstacle position $({}^{W}x_{ob}, {}^{W}y_{ob})$ and the size d_{ob} obtained by (15) are applied to the obstacle avoidance control of the robot as the obstacle information.

B. Path Planning

In this section, we describe a method for generating a trajectory from information obtained in Section III-A to enable the robot to safely avoid the obstacle. We propose a method of generating trajectories to avoid obstacles in virtual space and to control a robot according to trajectories [1].

The robot and the obstacle positions at the moment of obstacle detection are set in virtual space with the destination. Depending on the obstacle size, a circle of radius $d_{OA} = d_{safe} + d_{ob}$ is defined as the dangerous area, as shown in Fig. 8. The value of d_{safe} depends on the size of the actual robot, but it is taken to be 300 (mm) in this paper. In virtual space, the translational velocity $v_v(k_v)$ and turning velocity $w_v(k_v)$ are imparted to the robot to reach the destination linearly. The translational velocity $v_v(k_v)$ is fixed and the turning velocity $w_v(k_v)$ is determined by the following (18).

$$w_v(k_v) = \begin{cases} A(d)(\phi_{ob} - \phi_R(k_v) \pm \frac{\pi}{2}) & \text{if} \quad d \le d_{OA} \\ \phi_g(k_v) - \phi_R(k_v) & \text{otherwise} \end{cases} \quad (18)$$

The variables in (18) are given by (19). Further, $A(d)$ is defined as the danger function and is described in detail later.

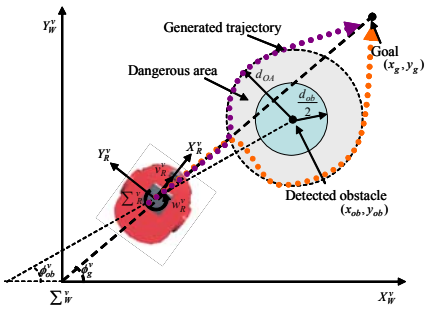


Fig. 8. Trajectory generation for obstacle avoidance.

$$
\begin{aligned}
d &= \sqrt{(x_R(k_v)-x_{ob})^2+(y_R(k_v)-y_{ob})^2} \\
\phi_{ob}(k_v) &= \text{atan2}(y_{ob}-y_R(k_v), x_{ob}-x_R(k_v)) \\
\phi_g(k_v) &= \text{atan2}(y_g-y_R(k_v), x_g-x_R(k_v))
\end{aligned} \tag{19}
$$

In this paper, a trajectory is formed from transit points where the robot passes at the velocity of (18). Further, the robot is controlled to pass each transit point in the general set of transit points ($(x_R^v(0), y_R^v(0))$, $(x_R^v(1), y_R^v(1))$,…, $(x_R^v(k_v), y_R^v(k_v))$, … , $(x_R^v(k_{end}), y_R^v(k_{end}))$.

If the distance between the robot and the obstacle is less than d_{OA} or if the robot is within the dangerous area, a controlled variable is calculated to move the robot away from the obstacle. When the robot is outside the dangerous area, a controlled variable is calculated to move the robot to the destination from its current position. To determine this controlled variable, the danger function $A(d)$ is used. Actually, the turning velocity $w_v(k_v)$ in (18) is regarded as the controlled variable of the robot that is calculated using the danger function. Based on the danger function, a controlled value that is sufficient to move the robot away from the obstacle is available only if the robot approaches the obstacle excessively. However, in (18), the controlled variable depends on the distance d to the obstacle and the obstacle size is not considered. Therefore, a trajectory causing a collision with an obstacle may be generated. Therefore, to reduce the possibility of a collision with an obstacle, a danger function incorporating the obstacle size should be considered for a smooth trajectory. The danger function expressed using the well-known Gaussian distribution is adopted in this paper.

$$
A(d) = \frac{1}{\exp(d^2/\sigma^2)} \tag{20}
$$

In (20), σ is a value variable with the obstacle size. In here, $\sigma=d_{OA}/2$. This produces the obstacle avoidance trajectory for the robot to move slowly away from the obstacle around the perimeter of the dangerous area and to move faster as the distance from the obstacle becomes shorter.

Two kinds of trajectories can be generated by selecting the sign ± in (18). One is for passing on the left side of the obstacle and the other is for passing on the right side. However, a trajectory of smaller load on the robot should be selected. Then, a trajectory that minimizes the evaluation function of (21) should be selected by noting the inclination changes of a fine section on the trajectory.

$$
J_n = \sum_{k_v=0}^{k_{end}} \tan^{-1}\left(\frac{y_R^v(k_v+1)-y_R^v(k_v)}{x_R^v(k_v+1)-x_R^v(k_v)}\right) \tag{21}
$$

For Fig. 8, the sign + reduces the evaluation function of (21). Then, the trajectory is for passing on the left side of the obstacle.

C. *Path Tracking Method*

To avoid obstacles, the robot is controlled to track each trajectory generated in Section III-B passing transit points. Since a trajectory is generated at each sampling time ΔT , it is necessary to know the coordinate points on a trajectory that the robot should track until the next sampling time. Trajectory is searched for the coordinate point $(x_R^v(k_v), y_R^v(k_v))$ closest to the robot position $(x_R(k), y_R(k))$. Then the coordinate point $(x_R^v(k_v+q), y_R^v(k_v+q))$, q points or q time ahead of the closest point searched, is defined as the destination of the robot or the reference point $(x_d(k), y_d(k))$. Here, q is determined experimentally. As the value becomes smaller, trajectory tracking is more accurate but robot movement is less smooth. As q becomes greater, robot movement is smoother but the trajectory tracking is less accurate. In this paper, $q=10$ was experimentally derived both for accurate tracking and smooth movement. For movement to the set reference point $(x_d(k), y_d(k))$, the translational velocity $v_R^{cmd}(k)$ and the turning speed $w_R^{cmd}(k)$ of the robot are calculated by (22).

$$
\begin{aligned}
&v_R^{cmd}(k+1) \\
&\quad = \alpha \cdot \sqrt{(x_R(k+1)-x_d(k+1))^2+(y_R(k+1)-y_d(k+1))^2} \\
&w_R^{cmd}(k+1) = \beta \cdot (\phi_d(k+1)-\phi_R(k+1)) + \hat{\phi}_d(k+1)
\end{aligned} \tag{22}
$$

$$
\begin{aligned}
\phi_d(k+1) &= a\tan 2(y_R(k+1)-y_d(k+1), x_R(k+1)-x_d(k+1)) \\
\hat{\phi}_d(k+1) &= \phi_d(k+1)-\phi_d(k)
\end{aligned} \tag{23}
$$

where α and β are arbitrary constants. To implement the translational velocity $v_R^{cmd}(k)$ and turning velocity $w_R^{cmd}(k)$ obtained by (22), the rotational velocity of each robot wheel is calculated using (24).

$$
\begin{aligned}
\hat{\theta}_r^{cmd}(k+1) &= \frac{1}{2a}\cdot(b\cdot w_R^{cmd}(k+1)+2\cdot v_R^{cmd}(k+1)) \\
\hat{\theta}_l^{cmd}(k+1) &= \frac{1}{2a}\cdot v_R^{cmd}(k+1) - \hat{\theta}_r^{cmd}(k+1)
\end{aligned} \tag{24}
$$

In here, Fig. 9 denotes the relationship between the

above-mentioned reference point $(x_d(k), y_d(k))$, the robot, and the obstacle.

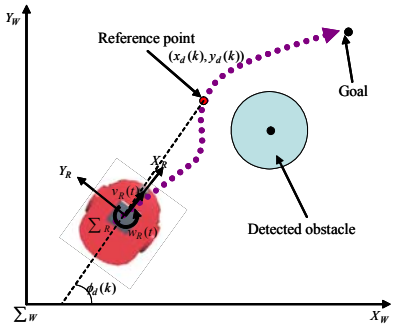


Fig. 9. Trajectory tracking control.

The method described so far may generate an unsatisfactory trajectory in the depth-wise direction. Therefore, the robot may have difficulties in avoiding an obstacle that is outside the field of view of the CCD camera or that is next to the robot. Therefore, Information on the side direction of the robot is incorporated from the range sensor S_n (n=0, 1, 4, 5) shown in Fig. 1 to avoid obstacles more accurately. If the obstacle is detected by the side range sensor, the turning velocity $w_v(k_v)$ satisfying (25) is calculated from (18) and input into the robot as a controlled variable. However, the translational velocity $v_v(k_v)$ is fixed as in (18).

$$
\begin{aligned}
w_v(k_v) &= \varepsilon \cdot A(l_n) \cdot \frac{\pi}{2} \\
A(l_n) &= \frac{1}{\exp(l_n^{\,2} / \sigma^2)} \\
\varepsilon &= \begin{cases} -1 \text{ if } n = 0 \text{ or } n = 1 \\ 1 \text{ if } n = 4 \text{ or } n = 5 \end{cases}
\end{aligned}
\tag{25}
$$

where the measurement distance l_n of the range sensor S_n (n=0, 1, 4, 5) is ≤250 mm, and the controlled variable is calculated by (25). However, the minimum value in each l_n should be selected.

IV. Experimental Result

We consider the mobile robot which moves between two rooms where many obstacles, like desks, chairs, and walls, exist. See, Fig. 10. In this experiment, the destination is set at the coordinates (6500mm, -3200mm) ahead of the robot. As shown in Fig. 11, it follows from the experimental result that the robot reached the destination avoiding the obstacles. As a result, a trajectory generated by our proposed method in Section III basically drives the robot to the destination. However, the robot tends to move toward the destination even when there is a wall beside the robot or an obstacle outside the field of view of the CCD camera. Although the side range sensors of the robot help to avoid a collision with the obstacle by (25), it says that it is necessary to detect the space around the robot more accurately for efficient robot control.

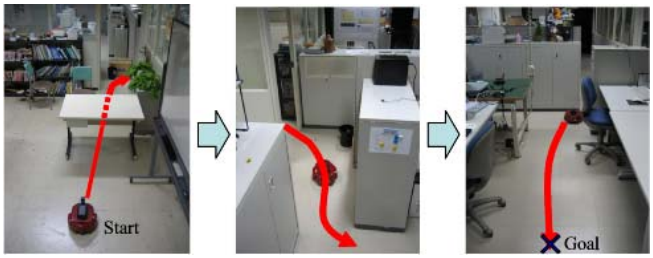


Fig. 10. Setup of experiment.

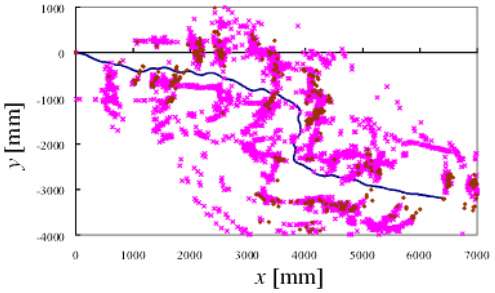


Fig. 11. Result of experiment.

V. Conclusion

This paper proposed an available method of obstacle avoidance control based on vision and range sensors mounted on the mobile robot. First, depending on the change of the brightness on the image that occurs from the moving of the mobile robot, we calculated the optical flow according to the block matching method based on the normalized correlation and detected the obstacle area on the image. And then, to reduce the error of the detection area, combining the distance information by range sensors on the image, we found the position of obstacle with high accuracy. From this position information, we decided the reference points for generating the trajectory. This trajectory is smooth and is generated by minimizing a certain cost function. Finally, usefulness of our proposed method is shown through the experiment.

References

[1] M. Egerstedt and X. Hu, "A hybrid control approach to action coordination for mobile robots", *Automatica*, vol. 38, no. 1, pp. 125-130, 2002.

[2] A. Giachetti, M. Campani, and V. Torre, "The use of optical flow for road navigation", *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 34-48, 1998.

[3] B. K. P. Horn, *Robot Vision*, The MIT Press, 1986.

[4] S. Hutchinson, G. D. Hanger, and P. I. Corke, "A tutorial on visual servo control", *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651-670, 1996.

[5] N. Kobayashi and M. Shibata, "A visual tracking method for a moving object using stereo vision robot", *IEEJ Transactions on Industry Applications*, vol. 127, no. 6, pp. 643-650, 2007. (in Japanese)

[6] K. C. Koh, and H. S. Cho, "A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints", *Journal of Intelligent and Robotics Systems*, vol. 24, pp. 367-385, 1999.

[7] T. Kubota, H. Hashimoto, and F. Harashima, "Navigation of mobile robot based on cooperation of vision and range sensors", *Journal of the Robotics Society of Japan*, vol. 7, no. 4, pp. 275-283, 1989. (in Japanese)

[8] N. Oda and H. Shimizu, "Power assist control of robotic wheelchair based on visual feedback", *IEEJ Transactions on Industry Applications*, vol. 128, no. 1, pp. 41-47, 2008. (in Japanese)