Int. J. Fuzzy Syst. (2016) 18(6):999–1007
DOI 10.1007/s40815-016-0220-y

Real-Time Self-Localization of a Mobile Robot by Vision and Motion System

Shu-Yin Chiang[1] · Chi-An Wei[1] · Ching-Yi Chen[1]

Received: 25 January 2016 / Revised: 18 May 2016 / Accepted: 23 June 2016 / Published online: 5 July 2016

Abstract An autonomous robot with an omni-vision camera and omni-moving platform is designed to satisfy the requirement of the Federation of International Robot-soccer Association and RoboCup robot soccer competitions. To obtain the robot's location on the field, we use a white-line pattern match localization algorithm. However, when the white-line information is incomplete during the matching process, the observed data differ significantly from a pre-built database. Thus, the localization result causes errors. In this study, we introduce an encoder localization algorithm to obtain a robot's moving direction and distance. We propose an algorithm that integrates white-line pattern match localization and encoder localization. In the integration process, we use a fuzzy system to search the surrounding points to localize the robot. The results demonstrate that integration localization outperforms localization with only white-line pattern match localization or encoder localization. With the proposed algorithm, we can obtain the robot's location within 30 ms at an error of less than 10 cm. The integration localization algorithm is compared to other methods to demonstrate its performance.

✉ Shu-Yin Chiang
sychiang@mail.mcu.edu.tw

Chi-An Wei
kevinwei3@gmail.com

Ching-Yi Chen
chingyi@mail.mcu.edu.tw

[1] Department of Information and Telecommunications Engineering, Ming Chuan University, Gui-Shan, Taoyuan 333, Taiwan

Keywords Omni-vision · White-line pattern match localization · Encoder localization · Gyroscope · FIRA · Fuzzy system

1 Introduction

Two famous mid-sized robot soccer competitions, the Federation of International Robot-soccer Association (FIRA) and RoboCup, have certain robot behavior requirements. In these competitions, the robots must make decisions and move autonomously. To meet these requirements, self-localization has become increasingly important recently. Self-localization faces the challenges of the localization times, but also spots on the result's preciseness. In this study, we use the FIRA competition as our design goal. The size of the FIRA RoboSot competition field is 6 m × 4 m (width × length) surrounded by a 75-cm border. The field is marked with 5-cm-wide white lines, as shown in Fig. 1a. The robot, with dimensions of 40 cm × 40 cm × 70 cm (width × length × height), is equipped with an omni-vision camera and a high-performance computer. The robot is also equipped with an omni-moving platform that has four high-power motors, a ball-holding mechanism, a ball-kicking mechanism, and FPGA controllers, as shown in Fig. 1b.

Some studies [1–5] have used the field geometry, such as goal posts or white lines, to calculate the position of a robot. White-line information has been used with the Monte Carlo method [2, 6, 7] and particle filters [8, 9] to localize a robot's position. However, the computation time is greater than 10–200 ms due to the number of iterations. Chiang [3] proposed a white-line pattern match localization method. The white-line data of each image of the soccer field obtained by a robot were matched with simulated

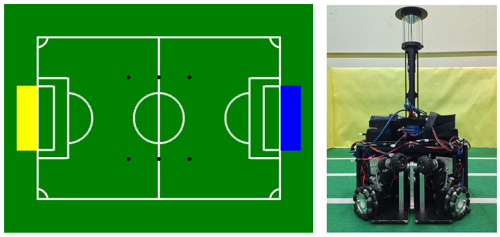

Fig. 1 **a** FIRA Robosot competition field, **b** robot

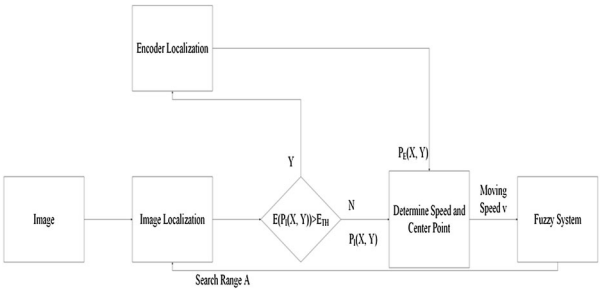

Fig. 2 System diagram

models pre-built in a database to obtain localization results. However, if the white lines on the field are incomplete or there are many points in the database that have the same compared error as that of the white-line pattern extracted from the image, localization will be incorrect. These limitations and poor reliability make robot localization with white-line patterns ineffective in real competition.

Therefore, some studies have employed encoders to localize the mobile robot [10–12]. However, encoder localization may cause errors if the wheels skid on the surface. Therefore, avoiding the effects of light on the vision system and skidding on the surface by the motion system are the goals of this study. In this study, we look for a reliable algorithm to accommodate white-line pattern match localization [3] and avoid significant error when white-line data in the field interfere with the environment. Both vision and motion can be integrated into an algorithm to support the effective localization of the robot. How to integrate vision and motion system, the fuzzy system is applied in the decision making system. The fuzzy system is a rule-based system that can rely on the practical experience and it has been applied in mobile robot to perform navigation [13, 14], target tracking [15], and trajectory tracking [16], etc. Thus, we propose a self-localization system that uses vision and motion information and a fuzzy system to search surrounding points to localize the robot in the integration process. The proposed scheme can operate within 30 ms with an accuracy of 6- to 10-cm deviation in a 6 m × 4 m field. The remainder of this paper is organized as follows. The algorithm is proposed in Sect. 2, experimental results are presented in Sect. 3, and conclusions are given in Sect. 4.

2 The Proposed Algorithm

In this section, we introduce the self-localization system that uses vision and motion information and a fuzzy system to search surrounding points to localize the robot in the integration process. The system diagram is shown in Fig. 2, the system use the image localization method to find the location of the robot. If the position error is greater than the threshold value, then the encoder localization method is employed. Both method provide the location for robot, the fuzzy system uses the moving speed of the robot to output the search range for image localization. More details of the system will be discussed in the following section.

2.1 White-Line Pattern Match Localization

The field in Fig. 1a is divided into a 75 × 55 grid points. On each grid in the system model, we calculate the distances between the first four white-line pixels from all 360-degree directions and the center of the grid point. For example, in Fig. 3, 360 scan lines were measured to calculate the distance (pixel) of the first four white pixels. The feature vector of this grid point is defined in (1):

$$P_{\mathrm{M}}(X_i, Y_j) = P_{i,j}(D_0, D_1, D_2, \ldots, D_{359}), \tag{1}$$

where P_{M} is the position from the simulation system model, is the ith column and jth row of the grid point, and $[D_0, D_1, D_2, \ldots, D_{359}]$ are the vector distances of the first four white pixels from 0, 1, …, 359° to the center of the grid, respectively.

$$D_k = \left[D_k^1, D_k^2, D_k^3, D_k^4\right], \quad k = 0, \ldots, 359. \tag{2}$$

Here, $D_k^n, n = 1, \ldots, 4$ are the first, second, third, and fourth white pixels in line D_k.

In the next step, we obtain an omni-directional image of the soccer field and attempt to match it to the simulated models in the database. Here, assume that the image of the field from the omni-directional camera is taken at location P_O. Then, the distances between the center of the image and the first four white-line pixels in all directions are expressed as follows (3):

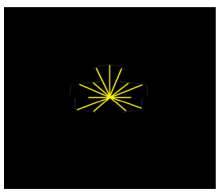

Fig. 3 First four *white-line* pixels vector

$$P_O(X,Y) = P(d_0,\ d_1,\ d_2, \ldots, d_{359}), \tag{3}$$

where d_0, d_1,..., d_{359} are the distances (pixel) of the first four white pixels from 0, 1, ⋯, 359 degrees to the center of the robot and defined in (4).

$$d_k = [d_k^1, d_k^2, d_k^3, d_k^4], \quad k = 0, \ldots, 359, \tag{4}$$

where $d_k^n, n = 1, \cdots, 4$ are the first, second, third, and fourth white pixels in line d_k. The omni-vision of the robot is shown in Fig. 4a, the corresponding scan lines to find the first four white pixels of the robot are shown in Fig. 4b, and the simulated system model pre-built in the database is shown in Fig. 4c. Then, the error between image positions and $P_M(X_i, Y_j)$ for all grid point (X_i, Y_j) is defined by (5).

$$\begin{aligned} E_{i,j} &= \left|P_{i,j}(D_0,\ D_1,\ D_2, \ldots, D_{359}) - P(d_0, d_1, d_2, \ldots, d_{359})\right| \\ &= \left(\sum_{k=0}^{359}\sum_{n=1}^{4}\left|D_k^n - d_k^n\right|\right). \end{aligned} \tag{5}$$

The white-line data of each image of the soccer field obtained by the robot are extracted and matched to the simulated models in the database to obtain localization results. The location with minimum error $E_{i,j}$ between the simulated model and the true field markings is the image location result $P_I(X, Y)$ in (6).

$$P_I(X,Y) = \min E_{i,j}, \quad i \in A, \quad j \in A, \tag{6}$$

where i and j are the ith column and jth row of the grid point in the search range A, respectively (Figs. 3, 4).

2.2 Encoder Feedback Circuit

In this study, we use a quad two-channel motor encoder with 500 counts per turn equipped at the back of each motor. When the motor rotates, the wheel inside the encoder turns and generates two digital pulses, as shown in Fig. 5. We employ a feedback circuit on an FPGA development board (myRio, National Instruments). The FPGA has four feedback circuits, each with a frequency divider, trigger module,

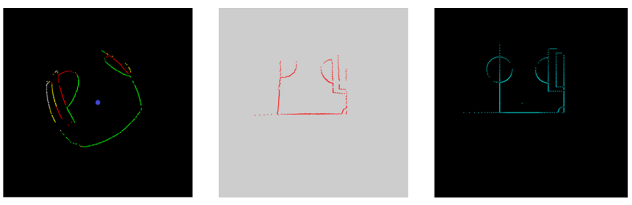

Fig. 4 **a** *White line* of the field captured by camera; **b** *white-line* pattern transferred from pixels to distance; **c** *white-line* pattern of a point in pre-built database

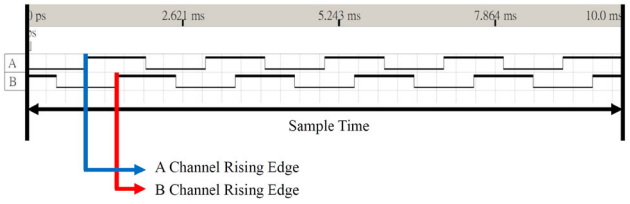

Fig. 5 Encoder signal chart

the encoder's two-channel counter, encoder direction detection, and a data register. The frequency divider reduces the myRio oscillator from 40 MHz to 100 Hz. The trigger module works on this 100-Hz clock to generate enable and clear signals to the two-channel counter and data registers in a period of 10 ms. The two-channel counters of the encoder are calculated at the end of the period.

2.3 Mathematical Model Review

According to the mathematical model of the four-wheel omni-moving platform, the speed of each wheel is defined by (7) and is shown in Fig. 6. Here, V_i is the rotation speed of the ith wheel, r is the radius of the wheel, and ω_i is angular velocity.

$$V_i = r\omega_i, \quad i = 1, \ldots 4, \tag{7}$$

From Figs. 7 and 8, we decompose the speed of wheel 2, V_2, to the X-axis and Y-axis, as expressed by (8):

$$V_2 = r\omega_2 = -\sin(\theta_m)V_x + \cos(\theta_m)V_y + R\omega_R, \tag{8}$$

where V_x and V_y represent two axes of the robot's center to the field moving velocity, ω_R is the angular velocity of the robot, θ_m is the angle of the motor equipped on the robot's base board (45° on our robot), and R is the distance between the robot's center and the wheel. The velocities of wheels 1–4 are expressed by (9).

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} r\omega_1 \\ r\omega_2 \\ r\omega_3 \\ r\omega_4 \end{bmatrix} = \begin{bmatrix} -\sin(\theta_m) & -\cos(\theta_m) & R \\ -\sin(\theta_m) & \cos(\theta_m) & R \\ \sin(\theta_m) & -\cos(\theta_m) & R \\ \sin(\theta_m) & \cos(\theta_m) & R \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega_R \end{bmatrix}. \tag{9}$$

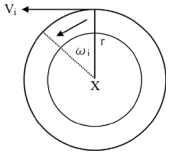

Fig. 6 Wheel variables

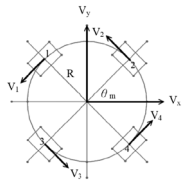

Fig. 7 Velocity of four-wheel omni-moving platform

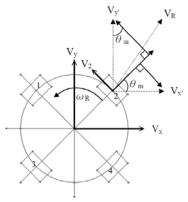

Fig. 8 Vector analysis of wheel 2

2.4 Localization Calculation

The FPGA circuit returns each motor encoder's signal count C_i in a sample period T seconds. Thus, the rotation distance of each wheel D_i can be calculated by (10) using the ratio of count C_i to the resolution of the encoder res. Then, it multiplies the circumference $2\pi r$ and time interval T.

$$D_i(T) = \frac{1}{N} \times \frac{C_i}{\text{res} \times 0.01} \times (2\pi r T), \quad i = 1, \ldots, 4, \tag{10}$$

where N is the reduction ratio defined in the motor specifications. This constant shows the difference between the motor feedback and the real wheel rotation speed. To obtain the direction of the robot, we use a gyroscope to calculate the angle difference between times T_j and T_{j-1}. Therefore, the angular velocity vector distance can be represented by the arc length in (11).

$$d = \theta_g \times \frac{\pi}{180} \times \frac{R}{2}, \tag{11}$$

where d is the angular velocity-distance or arc length we wish to obtain and θ_g is the difference between the current and previous gyroscope reported yaw, as shown in Fig. 9.

Next, we subtract the angular velocity-distance from the wheel's distance calculated by the motor rotation speed to obtain linear velocity-distance S_i in (12).

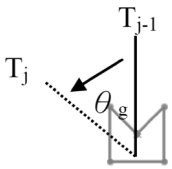

Fig. 9 Yaw rotation of the robot

$$S_i(T) = D_i(T) - d, \quad i = 1, .., 4. \tag{12}$$

Then, we can obtain each motor's two axes distance using trigonometric function operations in (13). This is illustrated in Fig. 10.

$$\begin{aligned}
S_{1x}(T) &= -\cos(\theta_m) S_1(T) \\
S_{1y}(T) &= -\sin(\theta_m) S_1(T) \\
S_{2x}(T) &= -\cos(\theta_m) S_2(T) \\
S_{2y}(T) &= -\sin(\theta_m) S_2(T) \\
S_{3x}(T) &= -\cos(\theta_m) S_3(T) \\
S_{3y}(T) &= -\sin(\theta_m) S_3(T) \\
S_{4x}(T) &= -\cos(\theta_m) S_4(T) \\
S_{4y}(T) &= -\sin(\theta_m) S_4(T)
\end{aligned} \tag{13}$$

Finally, by combining the four wheel's two axes distances in (14), we can obtain the moving distance of the robot in sample time T for both X- and Y-axes. Thus, the total moving distance of the robot within sample duration t is summarized in (15). Here, P_x and P_y are the localization result, and θ_e indicates the moving direction of the robot obtained by encoder localization.

$$\begin{aligned}
S_x &= \frac{1}{2} \sum_{i=1}^{4} S_{ix}(T) \\
S_y &= \frac{1}{2} \sum_{i=1}^{4} S_{iy}(T).
\end{aligned} \tag{14}$$

$$\begin{aligned}
\theta_e &= \tan^{-1}\left(\frac{S_y(T_j)}{S_x(T_j)}\right) + \theta_g \\
P_x &= \sum_{j=0}^{t} \left[\sqrt{S_x(T_j)^2 + S_y(T_j)^2} \times \cos(\theta_e)\right] \\
P_y &= \sum_{j=0}^{t} \left[\sqrt{S_x(T_j)^2 + S_y(T_j)^2} \times \sin(\theta_e)\right]
\end{aligned} \tag{15}$$

2.5 Integration Algorithm with Fuzzy System

In this section, we introduce the integration algorithm that combines white-line pattern match localization and encoder localization. Initially, the system uses white-line pattern match localization to search the pre-built database for the

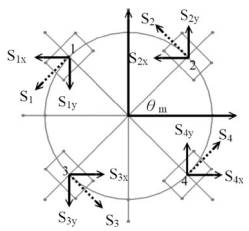

Fig. 10 Vector analysis of wheels distance

current robot location on the field to obtain the image localization result $P_I(X, Y)$ that minimizes the matching error.

After obtaining the result with error $E(P_I(X, Y))$ less than E_{TH} (assume E_{TH} is 20), the system will set this location as the point of origin for the encoder localization system and will accumulate the robot's moving distance. In this process, the encoder localization system will return results for each period to the white-line pattern match localization system. If $E(P_I(X, Y))$ is greater than E_{TH}, the system will adjust the robot's location given by the encoder localization system and limit the local search area in the next period (Fig. 11).

Next, we define a fuzzy system for local search to localize the robot's position. The fuzzy system uses the robot's moving speed as input and the search range as output as shown in Fig. 11. The system's membership function for moving speed is illustrated in Fig. 12, where VL, L, H, and VH denote very low, low, high, and very high, respectively. The membership function of search range A is shown in Fig. 13. From the if-then rule, we can obtain the output, search range A. The rules are as follows:

Rule1 If v is VL, then A is VL
Rule2 If v is L, then A is L
Rule3 If v is H, then A is H
Rule4 If v is VH, then A is VH

Hence, it is evident that there are four search ranges (0, 10, 20, and 30). The corresponding ranges and search points are plotted in Fig. 14. The black circle indicates the localization result and a red square in the database for those points marked in the white circle are the areas to calculate the white-line pattern match. The result is the location with the smallest error in the range. The computation times for each search area are summarized in Fig. 14. The fuzzy system can determine the local search area and obtain a search result in 0.1–1.2 ms. These results are used to compare the efficiency of the proposed system to other methods (Sect. 3).

However, the encoder's feedback has accumulated error and the encoder localization system will reset its point of origin in 125 ms. The integration algorithm is shown in Fig. 15.

In Fig. 15, the system uses global white-line pattern search to find the position $P_{IG}(X, Y)$. If the pattern match error is less than the threshold E_{TH}, the fuzzy system will perform a local search and obtain the result $P_{IL}(X, Y)$. If the search result error is greater than the threshold E_{TH}, the system will use the localization result from the encoder, i.e., $P_{IL}(X, Y) \leftarrow P_E(X, Y)$. Similarly, if the reset time is up and the error for image localization result is less than the threshold E_{TH}, then the system will use the localization result from the image, i.e., $P_E(X, Y) \leftarrow P_{IL}(X, Y)$. With the integration algorithm, the system can reduce the effect of light on the vision system and the ground effect of the field on the motion system. The fuzzy system provides a dynamic search area relative to the speed of the robot; the higher the speed, the larger the search area and vice versa. The fuzzy system can reduce the computation time from 1.2 to 0.1 ms.

Next, the time required for localization is calculated. Assume the image capture rate is approximately 60 frames per second, and the image processing and white-line pattern match times are 12.9 and 0.4 ms (20 cm search range), respectively. The timeline of the localization process is shown in Fig. 16. If the pattern match error from the image localization is greater than the threshold, then the integration localization will modify the result using encoder localization. For example, in Fig. 16, the image localization result is less than the threshold in the 1st image and 3rd image; therefore, the integration scheme retains the image localization. However, the error in the 5th image is greater than the threshold. Therefore, the position obtained by the encoder is updated by the integration algorithm.

3 Performance Analysis

We chose 19 points from the FIRA field as test points to analyze the performance of integration localization. The test points were divided into two categories, i.e., positions inside the field and special points on the field. Figures 17 and 18 indicate the locations of the test points.

We evaluated the white-line pattern match localization and encoder localization separately. Then, we evaluated the performance of the integration localization algorithm. The results for white-line pattern match localization for general cases (Fig. 17) and boundary cases (Fig. 18) are listed in Tables 1 and 2, respectively. The results for encoder

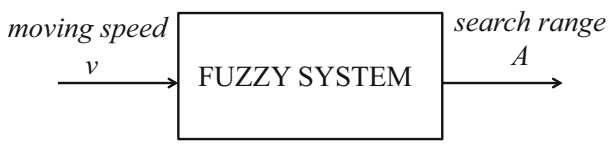

Fig. 11 Fuzzy system block diagram

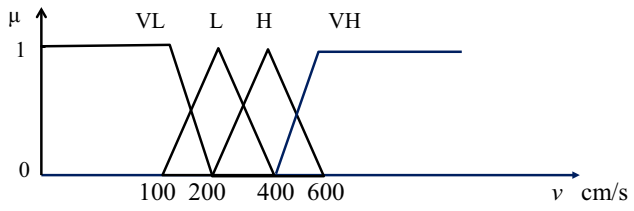

Fig. 12 Membership function for the robot's moving speed

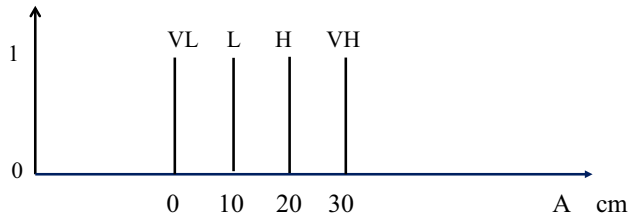

Fig. 13 Membership function for search range *A*

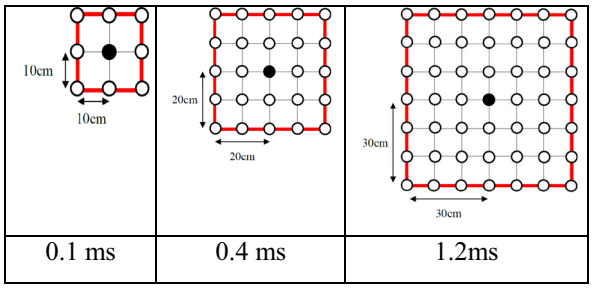

Fig. 14 Comparison range for local search and computation time

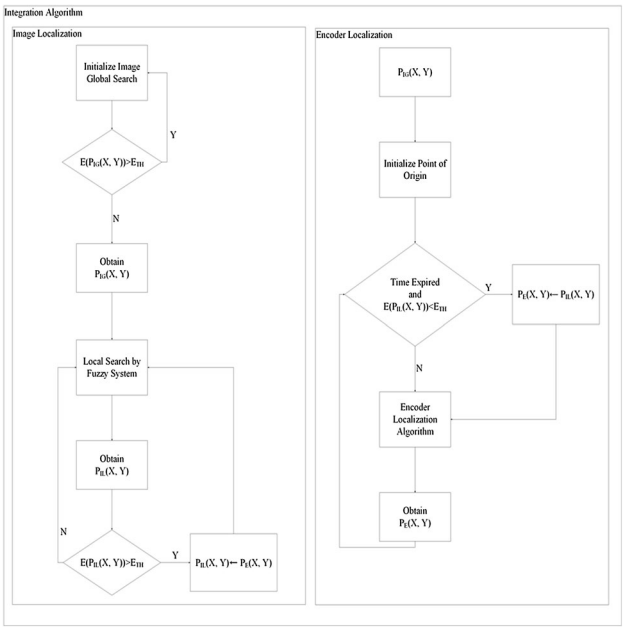

Fig. 15 Integration algorithm flowchart, (*The *white-line* pattern match localization omitted as image localization)

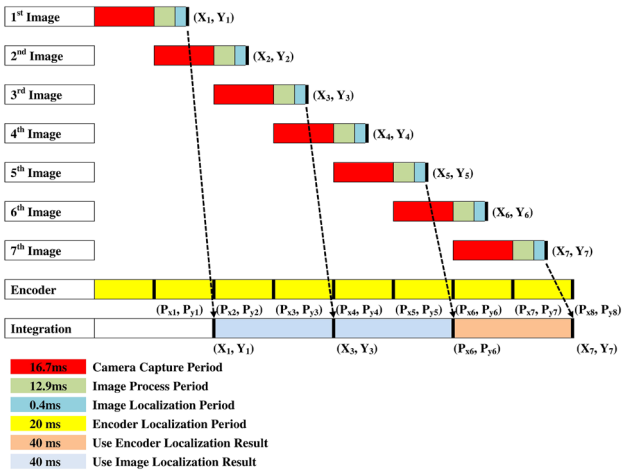

Fig. 16 Integration algorithm timeline

localization for general cases (Fig. 17) and boundary cases (Fig. 18) are listed in Tables 3 and 4, respectively. Finally, the results for integration localization for general cases (Fig. 17) and boundary cases (Fig. 18) are listed in Tables 5 and 6, respectively.

3.1 Image Localization

Initially, the white-line pattern match localization algorithm is used. From Tables 1 and 2, the results show that the average error is approximately 9 cm for general cases and 212 cm for boundary cases.

3.2 Encoder Localization

Next, we tested the encoder localization. For each test point, we moved the robot to the center of the field to reset the point of origin. The results are presented in Tables 3 and 4. The average error is approximately 10 cm for general cases (Fig. 17) and 18 cm for boundary cases. It is evident that encoder localization is better than image localization for test points on the field boundary (Fig. 18).

3.3 Integration Localization with Image and Encoder

In integration localization, the system uses the algorithm shown in Fig. 12 to obtain the robot's position. As shown in Table 5, the average error of integration localization for general cases is 6 cm. The average error of integration

Table 1 Test results with image localization (general cases; Fig. 17)

Test point	Real (X, Y)	Estimated (X, Y)	Error (cm)
1	(225, 173)	(230, 180)	9
2	(375, 188)	(370, 190)	5
3	(521, 170)	(530, 170)	9
4	(156, 273)	(180, 270)	24
5	(375, 280)	(370, 280)	5
6	(594, 281)	(600, 280)	6
7	(225, 366)	(230, 370)	6
8	(375, 382)	(370, 380)	5
9	(513, 375)	(500, 370)	14
Average error (cm)			9

Table 2 Test results with image localization (boundary cases; Fig. 18)

Test point	Real (X, Y)	Estimated (X, Y)	Error (cm)
1	(75, 75)	(70, 80)	7
2	(225, 73)	(220, 80)	9
3	(380, 74)	(530, 320)	288
4	(678, 71)	(540, 100)	141
5	(722, 82)	(140, 260)	609
6	(90, 466)	(590, 270)	537
7	(250, 475)	(240, 470)	11
8	(375, 475)	(570, 240)	305
9	(520, 476)	(520, 480)	4
10	(662, 467)	(140, 270)	558
Average error (cm)			212

Table 3 Test results with encoder localization (general cases; Fig. 17)

Test point	Real (X, Y)	Estimated (X, Y)	Error (cm)
1	(255, 174)	(257, 175)	2
2	(382, 171)	(378, 180)	10
3	(508, 166)	(508, 179)	13
4	(155, 271)	(167, 272)	12
5	(375, 275)	(375, 275)	0
6	(596, 268)	(580, 265)	16
7	(240, 366)	(246, 355)	13
8	(382, 387)	(379, 375)	12
9	(519, 368)	(512, 362)	9
Average error (cm)			10

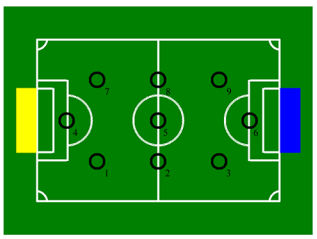

Fig. 17 Nine test points from general cases

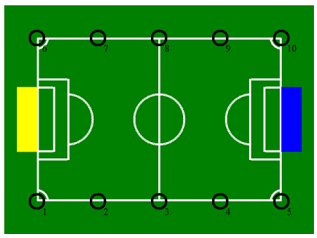

Fig. 18 Ten test points for boundary cases

Table 4 Test results with encoder localization (boundary cases; Fig. 18)

Test point	Real (X, Y)	Estimated (X, Y)	Error (cm)
1	(82, 72)	(95, 79)	15
2	(251, 70)	(255, 81)	12
3	(379, 71)	(374, 87)	17
4	(507, 72)	(493, 76)	15
5	(670, 65)	(660, 89)	26
6	(83, 478)	(92, 453)	27
7	(237, 480)	(237, 463)	17
8	(385, 475)	(377, 464)	14
9	(520, 486)	(518, 465)	21
10	(667, 482)	(661, 468)	15
Average error (cm)			18

Table 5 Test Results with integration localization (general cases; Fig. 17)

Test point	Real (X, Y)	Estimated (X, Y)	Error (cm)
1	(225, 170)	(220, 170)	5
2	(375, 173)	(370, 170)	6
3	(517, 166)	(520, 170)	5
4	(157, 278)	(150, 280)	7
5	(375, 275)	(370, 270)	7
6	(595, 274)	(600, 270)	6
7	(225, 385)	(230, 380)	7
8	(368, 371)	(370, 370)	2
9	(507, 380)	(520, 380)	13
Average error (cm)			6

Table 6 Test results with integration localization (boundary cases; Fig. 18)

Test point	Real (X, Y)	Estimated (X, Y)	Error (cm)
1	(81, 82)	(76, 83)	5
2	(225, 73)	(230, 70)	6
3	(386, 78)	(370, 82)	16
4	(525, 80)	(515, 100)	22
5	(661, 76)	(660, 80)	4
6	(89, 479)	(73, 468)	19
7	(245, 479)	(230, 470)	17
8	(386, 479)	(370, 476)	16
9	(509, 475)	(520, 470)	12
10	(649, 479)	(662, 480)	13
Average error (cm)			13

localization in Table 6 for boundary cases is 4–22 cm, and the average error is 13 cm. Thus, it is evident that integration localization can eliminate the effect of light on the

Table 7 Comparison of errors of proposed scheme with other methods

Error	Method				
	Map and MCL [6]	Adaptive MCL [7]	NM-EPF [8]	Match pattern [5]	Proposed scheme
(cm)	30	19.36	88.13	20	6

Table 8 Comparison of efficiency of proposed scheme with other methods

Time	Method				
	Map and MCL [6]	Adaptive MCL [7]	Match pattern [4]	Match pattern [9]	Proposed scheme
(ms)	195	1.76	3.2	4.2	0.4

vision system and does not accumulate errors due to encoder distance calculations.

3.4 Comparison with Other Work

To demonstrate that the proposed scheme has higher accuracy and higher efficiency than other systems, we compared the proposed algorithm to adaptive Monte-Carle localization (MCL) [6, 7], localization with enhanced particle filter (EPF) [8], and pattern matching algorithms [4, 5, 9]. For fair comparison, we neglected the frame rate and image processing time, and defined the computation time as the pattern match time using the fuzzy system. In Monte-Carle localization and localization with enhanced particle filter, the system chooses two hundred sample points in the simulation. The results are shown in Tables 7 and 8. The mean error of the proposed scheme was approximately 6 cm; however, the errors for the pattern matching and MCL algorithms were 20–88 cm. Similarly, we also compared the computation time for localization of the proposed algorithm to the MCL and pattern matching algorithms. The computation time for the proposed scheme was approximately 0.4 ms; however, the computation time for the pattern matching and MCL algorithms was 1.76–195 ms. Thus, the proposed integration localization has higher accuracy and efficiency than MCL or pattern matching schemes.

4 Conclusions

In this study, we have proposed a localization algorithm that can integrate the vision and motion systems of a robot. In the vision system, white-line pattern match localization is used to determine the initial location of the robot. Then, the distance and moving direction are calculated by encoder localization. Integration localization using a fuzzy system is proposed to reduce the effect of light on the vision system and the ground effect of the field on the motion system. Our experimental results show that a more reliable and precise location can be obtained by combining vision localization and motion localization. The results indicate that performance errors are less than 10 cm with 30-ms positioning time. The main contribution of this study is real-time computation of the localization scheme and accurate results. The proposed scheme can localize the robot and help design strategies for the robot in mid-sized FIRA and RoboCup competitions.

References

1. Merke, A., Welker, S., Riedmiller, M.: Line based robot localization under natural light conditions. In: Workshop on Agents in Dynamic and Real Time Environments (2004)
2. Menegatti, Emanuele, Pretto, Alberto, Scarpa, Alberto, Pagello, Enrico: Omnidirectional vision scan matching for robot localization in dynamic environments. IEEE Trans. Robot. **22**(3), 523–535 (2006)
3. Chiang, S.-Y., Guo, X., Hu, H.-W.: Real time self-localization of omni-vision robot by pattern match system. In: International Conference on Advanced Robotics and Intelligent System, pp. 46–50 (2014)
4. Liu, Jianshen, Yin, Baoyong, Liao, Xinxing: Robot self-localization with optimized error minimizing for soccer contest. J. Comput. **6**(7), 1485–1492 (2011)
5. Liu, B., Fan, J., Zhou, J., Li, K., Xie, Y.: A self-localization method through pose point matching for autonomous soccer robot based on omni-vision. In: The 9th International Conference on Electronic Measurement and Instruments, pp. 246–249 (2009)
6. Kim, H., Oh, T., Lee, D., Myung, H.: Image-based localization using prior map database and monte carlo localization. In: The 11th International Conference on Ubiquitous Robots and Ambient Intelligence, pp. 308–310 (2014)
7. Heinemann, P., Haase, J., Zell, A.: A combined Monte-Carlo localization and tracking algorithm for RoboCup. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1535–1540 (2006)
8. Hsu, C.-C., Wong, C.-C., Teng, H.-C., Li, N.-J., Ho, C.-Y.: Localization of mobile robots via an enhanced particle filter. In:

IEEE on Instrumentation and Measurement Technology Conference, pp. 323–327 (2010)
9. Lauer, M., Lange, S., Riedmiller, M.: Calculating the perfect match: an efficient and accurate approach for robot self-localization. In: RoboCup 2005: Robot Soccer World Cup IX, vol. 4020 of the series Lecture Notes in Computer Science, pp. 142–153
10. Chen, C.-L., Zhou, J.-H., Cheng, A.-C., Huang, S.-H.: Accurate navigation and localization for differential wheeled robots by tracking the embedded encoders. Adv. Inf. Sci. Serv. Sci. **7**(2), 65–79 (2015)
11. Chen, C.-L., Huang, S.-H., Zhou, J.-H., Mobile robot localization by tracking built-in encoders. In: International Symposium on Computer, Consumer and Control, pp. 840–843 (2014)
12. Lee, T.-J., Bahn, W., Jang, B.-M., Song, H.-J., Cho, D.-I.D.: a new localization method for mobile robot by data fusion of vision sensor data and motion sensor data. In: Proceedings of the IEEE International Conference on Robotics and Biomimetics (2012)
13. Lee, M.-F.R., Chiu, F.-H.S., de Silva, C.W., Shih, C.-Y.A.: Intelligent navigation and micro-spectrometer content inspection system for a homecare mobile robot. Int. J. Fuzzy Syst. **16**(3), 389–399 (2014)
14. Min, Byung-Cheol, Kim, Moon-Su, Kim, Donghan: Fuzzy logic path planner and motion controller by evolutionary programming for mobile robots. Int. J. Fuzzy Syst. **11**(3), 154–163 (2009)
15. Chao, C.-H., Hsueh, B.-Y., Hsiao, M.-Y., Tsai, S.-H., Li, T.-H.S.: Fuzzy target tracking and obstacle avoidance of mobile robots with a stereo vision system. Int. J. Fuzzy Syst. **11**(3), 183–191 (2009)
16. Hsiao, M.-Y., Chen, C.-Y., Li, T.-H.S.: Interval type-2 adaptive fuzzy sliding-mode dynamic control design for wheeled mobile robots. Int. J. Fuzzy Syst. **10**(4), 268–275 (2008)

Shu-Yin Chiang received the B.S. degree from the Tatung Institute of Technology, Taipei, Taiwan, R.O.C., in 1990, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1994 and 1999, respectively, all in electrical engineering. She is currently a Chair and Associate Professor in the Department of Information and Telecommunications Engineering, Ming Chuan University, Taiwan, R.O.C. Her research interests include intelligent control, robot applications and wireless sensor networks.

Chi-An Wei received his B.S. degree in Information and Telecommunications Engineering from the Ming Chuan University, Taoyuan, Taiwan, R.O.C., in 2016. He is going to pursuing the M.S. degree in Computer Science and Information Engineering from the Cheng Kung University, Tainan, Taiwan, R.O.C. His current research interests include intelligent control and robot applications.

Ching-Yi Chen received his M.S. and Ph.D. degrees in Electrical Engineering from Tamkang University, New Taipei City, Taiwan, in 1998 and 2006, respectively. He joined the Department of Information and Telecommunications Engineering at Ming Chuan University in 2007 as an Assistant professor. Currently he is an Associate professor at the department. His main research interests include swarm intelligence, pattern recognition, and embedded systems.