Received: 28 February 2018 | Revised: 10 May 2018 | Accepted: 27 May 2018

DOI: 10.4218/etrij.2018-0109

SPECIAL ISSUE

WILEY ETRI Journal

Optimization-based humanoid robot navigation using monocular camera within indoor environment

Young-Joong Han[1] | **In-Seok Kim**[2] | **Young-Dae Hong**[3]

[1]TmaxSoft, Suwon, Rep. of Korea.

[2]Hyundai Robotics, Suwon, Rep. of Korea.

[3]Department of Electrical and Computer Engineering, Ajou University, Suwon, Rep. of Korea.

Correspondence
Young-Dae Hong, Department of Electrical and Computer Engineering, Ajou University, Suwon, Rep. of Korea.
Email: ydhong@ajou.ac.kr

Funding information
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1C1B1006691).

Robot navigation allows robot mobility. Therefore, mobility is an area of robotics that has been actively investigated since robots were first developed. In recent years, interest in personal service robots for homes and public facilities has increased. As a result, robot navigation within the home environment, which is an indoor environment, is being actively investigated. However, the problem with conventional navigation algorithms is that they require a large computation time for their building mapping and path planning processes. This problem makes it difficult to cope with an environment that changes in real-time. Therefore, we propose a humanoid robot navigation algorithm consisting of an image processing and optimization algorithm. This algorithm realizes navigation with less computation time than conventional navigation algorithms using map building and path planning processes, and can cope with an environment that changes in real-time.

KEYWORDS
bipedal humanoid robot navigation, indoor environment navigation, particle swarm optimization

1 | INTRODUCTION

In robotics, navigation provides a robot with mobility. Navigation does not only allow for robots to move flexibly within various environments, but also allows them to broaden their range of activities. Therefore, navigation is applied to various types of robots, such as mobile robots, humanoid robots, and quadcopters. Moreover, research on robot navigation has been actively conducted [1–3].

Owing to the specificity of hardware, bipedal walking humanoid robots have been investigated within the field of navigation. Footstep planning is a research area in the field of navigation and has been investigated to create a path for the robot through path planning, and to determine the footsteps required to follow that path [4,5]. A problem encountered by these studies is that the global map must be initially known for the path to be generated. Moreover, because of hardware limitations and robot slippage during movement, it cannot be guaranteed that the given footsteps will be followed accurately. Consequently, this accumulation of errors can cause the robot to collide with obstacles along the path. To compensate for this problem, studies on the localization of robot positions have been conducted [6–8]. However, localization is only one of the navigation elements. For successful navigation, the position and direction of the robot should be determined. Environmental information is required to determine the direction of the robot's motion. Thus, sensors that receive environmental information are required, and various studies have been conducted to perform humanoid navigation using different types of sensors [9–11] such as laser ranges and ultrasonic sensors. In a few cases, multiple sensors have been used as needed. Various studies have generated maps or utilized pre-generated maps by using multiple sensors [12]. However, in such situations, the robot has to stop moving to obtain environmental information. Humanoid navigation studies using cameras within indoor environments have

also been conducted [13–15] and have performed navigation based on corridor lines or markers attached to obstacles. However, such investigations have been limited to static obstacles.

Existing robot navigation studies have encountered problems related to the use of multiple combined sensors and excessive sensor cost. Consequently, such problems have had a hindering effect on the commercialization of the navigation algorithms of various robots. As mentioned previously, the problem with a navigation algorithm using footstep planning is that the map has to be known or obtained in advance, and that a map generation process is required when the global map is not available. Algorithms using maps incorporate path planning for path selection in the used maps. However, this requires a substantial amount of computation time for map building and path planning, which makes it difficult to cope with environmental changes in real-time. Additionally, navigation using footstep planning faces difficulties in determining the position of the robot because it accumulates errors when the robot slips on the floor.

In this paper, we propose an indoor navigation algorithm to solve the problems encountered during the investigation of conventional robot navigation algorithms. In recent years, interest in personal service type robots has increased. Service robots are primarily used indoors, and because the floor of an indoor environment is made of relatively similar colors and patterns, it is easy to identify the areas where the robot can move. In this study, a monocular camera was used to utilize the floor as a navigational reference. The monocular camera is not only inexpensive, but also has very high usability because it can interpret and process information in various ways, in comparison with other sensors. In the proposed algorithm, the computation time was reduced by replacing the map building and path planning processes with the particle swarm optimization (PSO) algorithm, which is an optimization algorithm. This allowed the robot to respond to changes in the environment in real-time.

2 | ALGORITHM FOR HUMANOID NAVIGATION

This section presents the navigation algorithms of humanoid robots.

2.1 | Walking command for humanoid robots

In this study, robot motion was generated through the humanoid robot walking pattern generator using the linear inverted pendulum mode (LIPM) [16].

An example of humanoid robot movement is shown in Figure 1. In this figure, left represents the left foot step, while right represents the right foot step. In the case of a bipedal humanoid robot, movement is realized by inputs to the left and right feet. These inputs refer to the walking command C_w, which is defined as follows:

$$C_w = [S_F, S_L, \theta, T_{SS}, T_{DS}]^T$$

where S_F is the sagittal step length, S_L is the lateral step length, and θ is the walking direction. In the case of a bipedal humanoid, there exists a double support phase wherein both feet are supported on the ground, and a single support phase wherein only one foot is supported on the ground. The time of each state is defined as T_{DS} and T_{SS}, respectively.

2.1.1 | Flow of proposed algorithm

Figure 2 shows the proposed algorithm's flowchart. Image information is received through a camera mounted on the robot. In this study, two different images were used to estimate the movement of dynamic obstacles. The time to receive these two images is T_{SS} and T_{PSS}, respectively. Here, T_{SS} is the time at which the single support phase ends, while T_{PSS} is the time at which ΔT is subtracted from T_{SS} before the single support phase ends. Evidently, ΔT should be set to a value smaller than T_{SS}. We define the current image I_C as the image received at T_{SS}, and the

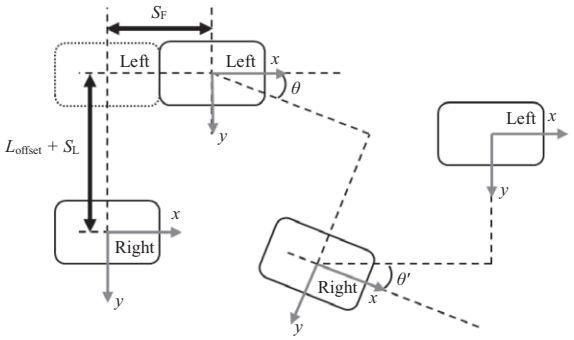

FIGURE 1 Bipedal robot walking command concept

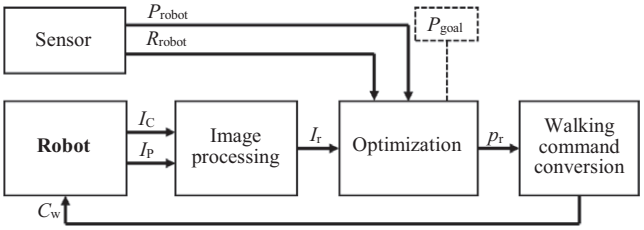

FIGURE 2 Proposed algorithm flowchart

previous image I_P as the image received at time T_{PSS}. Through image processing, the ground of the image is identified and the motion of the obstacles is estimated. Then, these are used to generate I_r, which is a processed image. The optimization algorithm is performed by inputting the I_r, robot position P_{robot}, robot orientation R_{robot}, and goal position P_{goal}. The inputs P_{robot} and R_{robot} can be obtained by estimating the marker on the robot with Vicon or a camera system within the indoor environment [17–19]. Through the optimization algorithm, the robot identifies the point closest to the goal point in the currently viewed image. Thereby, it avoids a collision with the obstacle. The optimization result is one of the pixels in the image and is defined as p_r. Because the pixel position in the image is not in scale with the actual world position, it is necessary to match the optimization result with the actual location scale. Additionally, because of hardware limitations, bipedal humanoids cannot instantaneously change direction and generate motion if large changes in direction occur. Therefore, it is necessary to convert p_r into a walking command such that the robot can move. Through this process, the robot navigation input C_w is generated. Accordingly, humanoid navigation is performed as described above. To generate robot motion in a single support phase, the input C_w for each double support phase is needed. Therefore, a series of processes to generate C_w should be performed within time T_{DS}, and the sum of the time for image processing, optimization, and walking command conversion computation must be less than T_{DS}, respectively.

3 | IMAGE PROCESSING

Contrary to infrared sensors, lidar, and so on, camera sensors are sensor systems that utilize image information instead of distance information. Moreover, a camera sensor system is considerably similar to the human eye. Recently, various studies using cameras, such as object identification studies using artificial intelligence, have been conducted. Therefore, it is expected that the use of cameras will continue to diversify. Thus, in this study, a camera was selected as the sensor system to receive environmental information. By using a camera as the sensor, an image that identifies the region where the robot can move is created. Dynamic obstacles in the image are considered such that they can be used as a non-transportable region for robot navigation. The image processing procedure to create a navigation image is shown in Figure 3.

3.1 | Ground image (I_g)

Indoor environments are different to outdoor environments because they typically have a consistent color or pattern on the floor. Image processing studies have been conducted to distinguish the floor or ground by using color and pattern recognition [20–22]. In this study, the ground was divided into areas by using the color range. Accordingly, images were generated based only on the ground.

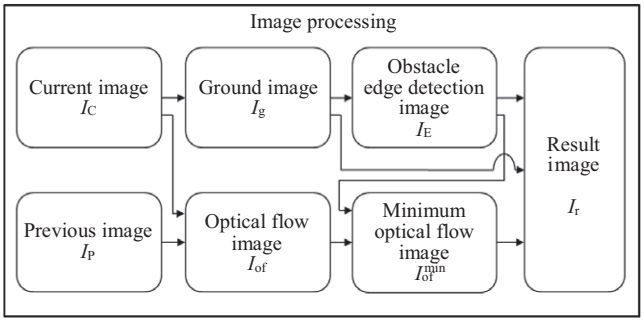

FIGURE 3 Image processing flowchart

3.2 | Obstacle edge detection image (I_E)

If all obstacles are in contact with the ground, the edge of the obstacle can be identified through the point where the ground and the obstacle meet. A pixel, with the current pixel as the obstacle and the pixel below it as the ground, is extracted by using the $[2 \times 1]$ mask in I_g. Through this technique, the edge of the obstacle can be obtained.

3.3 | Optical flow image (I_{of})

The optical flow is the pattern of the apparent motion of an obstacle image between two consecutive frames and is caused by the movement of the obstacle or camera. Moreover, it is a 2D vector field, where each vector is a displacement vector showing the movement of points from the first to the second frame [23]. The optical flows are divided into dense and sparse optical flows. Although a dense optical flow has high accuracy, it is disadvantaged by requiring extensive computation time to calculate all of the image pixels. Conversely, a sparse optical flow computes only a few pixels of the entire image, and although it is less accurate than a dense optical flow, its computation time is shorter. In this study, a sparse optical flow was used because the computation time was required to be smaller than T_{DS}. Specifically, we used the pyramidal Lucas-Kanade optical flow method, wherein the accuracy increases because the pyramidal method changes the image size in a step-by-step manner. Moreover, only a few reference pixels are required to obtain the sparse optical flow. These reference pixels are obtained by implementing the corner detection technique. Consequently, the movement of an obstacle in the image is tracked.

3.4 | Minimum optical flow image (I_{of}^{min})

The reference pixel selection utilizes the corner detection when calculating the optical flow. Therefore, if there are several reference pixels in one obstacle, multiple optical flows will be generated in one obstacle. However, only one optical flow must be selected to determine the moving speed of the obstacle.

The optical flow in the image is different from the positional change in the world coordinate system. Therefore, it is necessary to convert the image pixel to the position of the world coordinate system. An image projection model is used to represent the pixel change in the image as a position change in the world coordinates. Figure 4 shows an image projection model indicating the pixel point P_i, where the position P_w of the world coordinate system $\{w\}$ is located in the image plane. Moreover, the pixel coordinate system is used to represent the x_{pixel} and y_{pixel} points in the image plane. The image projection model is defined as follows:

$$P_i = K[R|t]P_w \tag{1}$$

where

$$P_i = \begin{bmatrix} x_{pixel} \\ y_{pixel} \\ 1 \end{bmatrix}, P_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, K = \begin{bmatrix} f_c & 0 & c_x \\ 0 & f_c & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

$$[R|t] = {}_w^cT = {}_c^wT^{-1}.$$

Here, P_i is the pixel position in the image plane, P_w is the position in the world coordinate system, K represents the camera internal parameter matrix that transforms the normalized image coordinates into image coordinates, f_c is the focal length of the camera, (c_x, c_y) are the pixels where the optical axis meets the image plane, and $[R|t]$ is a camera external parameter, which is a transformation matrix that transforms the world coordinate system $\{w\}$ into the camera coordinate system $\{c\}$. The camera coordinate system $\{c\}$ is the coordinate of the camera mounted on the robot. The camera coordinate system $\{c\}$ is defined as follows:

$$ {}_c^wT = R_Z(\alpha)R_Y(\beta)R_X(\gamma)T(P_c) \tag{2}$$

where

$$R_X(\gamma) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_Y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_Z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$P_c = \begin{bmatrix} x_{camera} \\ y_{camera} \\ z_{camera} \end{bmatrix}.$$

In the world coordinate system, the camera angles α, β, and γ, and the camera position P_c can be obtained from P_{robot} and R_{robot}.

In this study, it was assumed that the obstacle was in contact with the ground surface. Therefore, z_w became 0. In this case, if K and $[R|t]$ are known, P_w can be calculated by P_i. With this method, the position change ΔP in the world coordinate system can be determined from the pixel change of the optical flow. Based on the optical flow obtained from the two images during time ΔT, the velocity v_{of} of the optical flow can be obtained by dividing ΔP by ΔT.

To solve the optical flow noise problem and determine one optical flow per obstacle, the optical flow of the smallest v_{of} among the various optical flows in one obstacle is selected. The optical flow with the smallest v_{of} value is displayed to generate I_{of}^{min}.

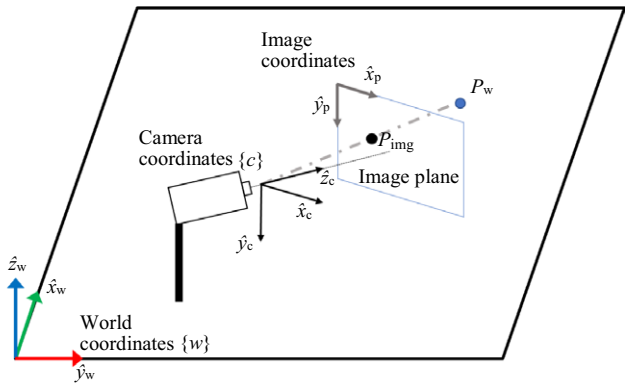

FIGURE 4 Image projection model

3.5 | Resulting image (I_r)

The objective of image processing is to create an image that represents an area wherein the robot can move by considering dynamic obstacles.

The velocity of the optical flow v_{of}, which was obtained previously, is the obstacle movement during ΔT. However, the obstacle moves in the next time step $T_{DS} + T_{SS}$. Therefore, the obstacle region should be expressed as a rate obtained by multiplying v_{of} by $T_{DS} + T_{SS}$. The image projection model is used to express the expected position of

the obstacle in the robot's next step. This indicates a region wherein the robot cannot move. The resulting image I_r is the region wherein the predicted position of the obstacle is removed from I_g. Hence, only the region where the robot can move is displayed in the image.

4 | OPTIMAL POINT SELECTION USING PARTICLE SWARM OPTIMIZATION ALGORITHM

Particle swarm optimization is an algorithm that was proposed by Russel Ebenhart and James Kennedy. It was developed based on the regularity of animal social behavioral patterns [24]. Based on individual cluster theory, movement is determined based on the experiences of the individual and the experiences of the swarm. The PSO algorithm is similar to the genetic algorithm (GA) because it also uses a population, which makes it possible to search a large area. Because the operation of the PSO algorithm is not as complicated as the evolutionary operation of the GA, the computation time of the PSO algorithm can be reduced in comparison with that of the GA.

This study was limited by the fact that the sum of the computation time for the image processing, optimization, and walking command change were required to be smaller than T_{DS}. This problem is similar to a global search problem in two-dimensional space because it requires the identification of an optimal point in the image. The solution to the optimization problem was attempted by using the PSO algorithm, which is suitable under the abovementioned limitation. In this section, the optimization problem is defined as the selection of points where the robot is able to move steadily while approaching the goal in the currently viewed image.

4.1 | Objective function

Because an optimal point in the image is being sought, each particle represents a pixel. The particle is defined as follows:

$$P_{\text{pixel}} = \left(x_{\text{particle}}, y_{\text{particle}}\right).$$

The position of this particle in the world coordinate system is defined as P_{ptc}. To evaluate the particles, the following objective functions are set.

4.2 | Cost function

The cost function evaluates how close the position of P_{pixel} is to the goal point in the world coordinate system. If the obstacle is in contact with the ground, the z-axis values in the world coordinate system are all zero. By using the image projection model, the P_{ptc} of the world coordinate system $\{w\}$ of the P_{pixel} can be obtained. The cost through the distance between P_{ptc} and P_{goal} in the world coordinate system $\{w\}$ can be evaluated as follows:

$$C = \sqrt{\left(P_{\text{ptc}} - P_{\text{goal}}\right)^T \left(P_{\text{ptc}} - P_{\text{goal}}\right)} \tag{3}$$

where

$$P_{\text{ptc}} = \begin{bmatrix} P_{\text{ptc},x} \\ P_{\text{ptc},y} \\ 0 \end{bmatrix}, P_{\text{goal}} = \begin{bmatrix} P_{\text{goal},x} \\ P_{\text{goal},y} \\ 0 \end{bmatrix}.$$

4.3 | Penalty function 1: Avoid collision at P_{ptc}

Penalty function 1 determines whether the robot will collide with an obstacle when it reaches P_{ptc}. When the P_{ptc} position of the world coordinate system is reached, the size of the robot is obtained from the image plane by reflecting the actual size of the robot. This region is defined as $\mathcal{R}_{\text{robot}}$. In I_r, $\mathcal{R}_{\text{ground}}$ and $\mathcal{R}_{\text{obs}}$ are defined as regions wherein the robot can and cannot move, respectively.

$$p_1 = \begin{cases} \infty & \left(\mathcal{R}_{\text{robot}} \cap \mathcal{R}_{\text{obs}} \neq \varnothing\right) \\ 0 & \left(\mathcal{R}_{\text{robot}} \cap \mathcal{R}_{\text{obs}} = \varnothing\right) \end{cases}, \tag{4}$$

Equation (4) expresses penalty function 1. If there is a pixel in $\mathcal{R}_{\text{robot}}$, where the robot cannot move among the pixels, this means that the robot collided with an obstacle upon reaching P_{ptc}.

4.4 | Penalty function 2: Avoid collision while moving

Penalty function 2 determines whether the robot will collide with an obstacle while moving towards P_{ptc}.

The image and image plane obtained through the robot's camera are shown in Figure 5. In the image plane, the large value of the y_{pixel} is closer to the robot. The pixels points closest to the robot are defined as $\text{pixel}_{\text{nearest}}^{\text{left}}$ and

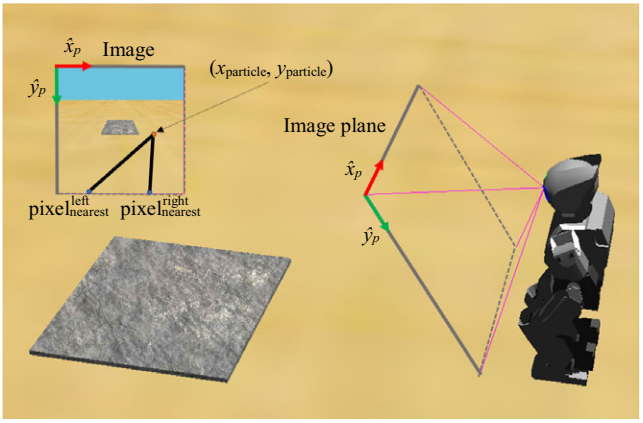

FIGURE 5 Robot camera image concept

$\text{pixel}_{\text{nearest}}^{\text{right}}$, and reflect the width of the robot. L_{left} is defined as the pixel set of the line connecting P_{pixel} and $\text{pixel}_{\text{nearest}}^{\text{left}}$, while L_{r} is defined as the pixel set of the line connecting P_{pixel} and $\text{pixel}_{\text{nearest}}^{\text{right}}$. Moreover, L_{l} represents the left boundary when the robot moves to P_{ptc}, while L_{r} represents the right boundary.

$$p_2 = \begin{cases} \infty, & L_l \cap \mathscr{R}_{\text{obs}} \neq \varnothing \text{ or } L_r \cap \mathscr{R}_{\text{obs}} \neq \varnothing \\ 0 \end{cases}. \quad (5)$$

Equation (5) expresses penalty function 2. If there are pixels in the region where the robot cannot move in the pixel sets L_{l} and L_{r}, this means that the robot collided with an obstacle while moving.

4.5 | Penalty function 3: Blind spot solution

Penalty function 3 was introduced to avoid a collision caused by the robot's viewing angle. The robot's camera was mounted at a certain height and did not have a full view. Owing to the camera's field of view, there existed an area that could not be seen in the image. This area is defined as the blind spot. The nearest area that the robot could see was the lower part of the image, while the closest area was the blind spot. In this regard, two pixels, namely, pixel_1 and pixel_2, at the edge closest to the image are defined in the image coordinate system as follows:

$$\text{pixel}_1 = (0, y_{\max}), \text{pixel}_2 = (x_{\max}, y_{\max})$$

where $y_{\max}$ is the largest y-pixel value, and $x_{\max}$ is the largest x-pixel value in the image plane. If an obstacle is detected in pixel_1, the robot will collide with the obstacle when it moves towards the left direction. To prevent this collision, the robot is given an S_{m} if x_{particle} is less than $x_{\max}/2$. For the same reason, when an obstacle is detected in pixel_2, the robot must refrain from proceeding towards the right direction. If x_{particle} is greater than $x_{\max}/2$, the robot is given an S_{m}. The number of S_{m} is equal to the invisible distance of the robot camera divided by S_{F}. If there are no obstacles in pixel_1 and pixel_2 in the next step, S_{m} is reduced by 1.

$$p_3 = \begin{cases} \infty & (S_{\text{m}} \neq \varnothing). \\ 0 \end{cases} \quad (6)$$

Equation (6) above expresses penalty function 3 and restricts the direction change to the direction of the obstacle, when S_{m} is required.

4.6 | Penalty function 4: Step angle restriction

Owing to hardware limitations, bipedal walking humanoid robots are not able to change their directions instantaneously or move with large changes in direction. Therefore, penalty function 4 was introduced to limit the angle change. In the world coordinate system, the directional angle θ_{p} is obtained based on the distance from the robot position P_{robot} to P_{ptc}, as follows:

$$\theta_{\text{p}} = \tan^{-1}\left(\frac{P_{1,x}}{P_{1,y}}\right) \quad (7)$$

where

$$P_{1,x} = P_{\text{ptc},x} - P_{\text{robot},x}, P_{1,y} = P_{\text{ptc}} - P_{\text{robot},y},$$

$$P_{\text{robot}} = \begin{bmatrix} P_{\text{robot},x} \\ P_{\text{robot},y} \\ P_{\text{robot},z} \end{bmatrix}.$$

Accordingly, $|\theta_{\text{safety}}|$ is defined to allow the robot to maximize its movement, depending on the robot model. However, because of the robot's field of view, a few additional steps are required to reach the $y_{\max}$ position in the world coordinate system, and this is defined as the N_{step}. Until the robot reaches the $y_{\max}$ point, the robot can change its angle (in degrees) by $|\theta_{\text{safety}}| \times N_{\text{step}}$.

$$p_4 = \begin{cases} \infty, & |\theta_p| > |\theta_{\text{safety}}| \times N_{\text{step}} \\ 0 \end{cases}. \quad (8)$$

Equation (8) expresses penalty function 4, which ensures the generation of stable motion for the robot.

The objective function is determined by the sum of the cost function and the penalty functions. The objective function to evaluate the particles is expressed as follows:

$$O_{\text{value}} = C + \sum_{i=1}^{4} p_i. \quad (9)$$

5 | WALKING COMMAND CONVERSION

For the robot to move during navigation, $S_{\text{F}}, S_{\text{L}}, \theta, T_{\text{SS}}$, and T_{DS} are required. Image processing and optimization should be performed within time T_{DS}. The value of θ can be obtained from the P_{best} position, which is obtained from the optimization result p_{r} in the world coordinate system and the robot's position (P_{robot}). Accordingly, θ is defined as follows:

$$\theta = \tan^{-1}\left(\frac{P_{2,x}}{P_{2,y}}\right) \quad (10)$$

where

$$P_{2,x} = P_{\text{best},x} - P_{\text{robot},x}, P_{2,y} = P_{\text{best},y} - P_{\text{robot},y},$$

$$P_{\text{best}} = \begin{bmatrix} P_{\text{best},x} \\ P_{\text{best},y} \\ P_{\text{best},z} \end{bmatrix}.$$

However, if O_{value}, which is the evaluation value of the p_r objective function that results from optimization, is ∞, there exists uncertainty in the resulting value. Therefore, in this case, movement is restricted. Hence, the robot is left to rotate and receive new environmental information. At that time, the walking command $C_{w,penalty}$ is defined as the motion.

6 | SIMULATIONS

In this section, the implementation of the proposed navigation algorithm in a simulation, and the confirmation of the algorithm's usability through the results of the navigation are discussed.

6.1 | Simulation composition

The proposed navigation algorithm was applied to a DARwIn-OP robot model by using a three-dimensional (3D) dynamics simulator and O*pen*CV. We used the Webots program, which is a 3D robotics simulation software developed by the Cyberbotics company [25]. Because image processing is restricted in the Webots software, the simulation proceeded according to the diagram shown in Figure 6.

First, the image was saved through the robot model of the simulator. Thereafter, it was imported into Visual Studio, where image processing and optimization were performed by using O*pen*CV and the PSO algorithm. Finally, robot navigation was performed by utilizing the optimization results that were input to the robot in the simulator.

6.2 | Configuration of simulation environment

The height, width, and depth of the DARwIn-OP robot were 0.455 m, 0.164 m, and 0.110 m, respectively. Therefore, the indoor environment was appropriately reduced to correspond to the robot's size.

Figure 7 shows the simulation environment. This environment was similar to that of a home, which is essentially

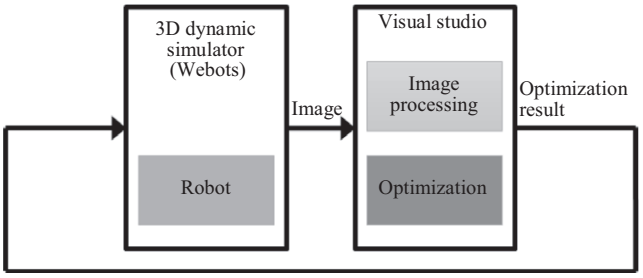

FIGURE 6 Simulation configuration diagram

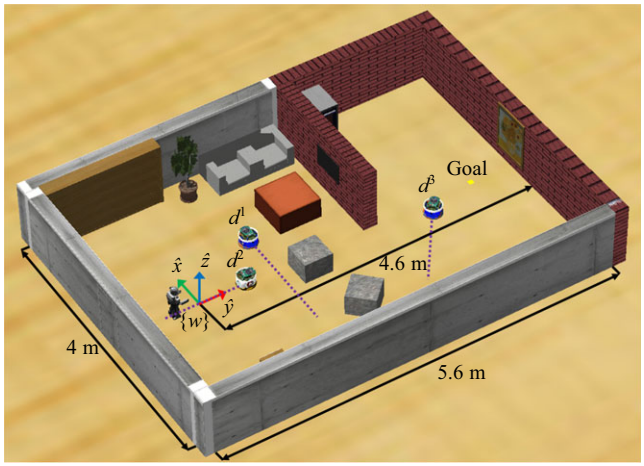

FIGURE 7 Simulation environment

an indoor environment. The coordinate system $\{w\}$ represents the world coordinate system in the simulation. This world coordinate system is a reference coordinate system to represent any position in the image. The indoor environment spanned 4 m in the x-axis and 5.6 m in the y-axis. The initial position of the robot was located at (0, −0.3), which means that it was 0 m in the x-axis and −0.3 m in the y-axis. The position of the robot was received in real-time by placing a global positioning system (GPS) on the robot during the simulation. To obtain the rotational angle of the robot's camera and the world coordinate system, the GPS was attached along the axis direction of the robot's camera. The rotational angle of each axis in the world coordinate system was obtained by the roll-pitch-yaw fixed angles, which rotated about each of the three rotation directions in the reference coordinate system. The roll, pitch, and yaw were equal to α, γ, and β, respectively, as discussed in Section 3.4. If the camera's field of view was narrow, the nearest position from the robot was distant. The pitch angle indicated the degree of bowing for the robot's head. In this study, the pitch angle was set to 30°. This allowed the robot to see closer areas. The camera mounted on the robot had a resolution of 720 × 720 pixels and a field of view of 90°. Because the robot was capable of bowing its head, it was also capable of seeing far points at a distance of approximately 0.16 m. Additionally, based on the hardware of the DARwIn-OP robot model used in this study, S_F was set to 0.04 m. Therefore, S_m and N_{step} were set to 4.

When the robot moved, if S_F and S_L were considerably distant, the stability of the robot could not be assured. Throughout the experiment, the parameter values were set to a range wherein the robot could walk stably by setting S_F and S_L to 0.04 m and 0 m, respectively. Moreover, the maximum walking direction angle $|\theta_{safety}|$ was 14°. Consequently, the angle at which the robot could rotate was

limited to 14°. A case where the optimization result's cost O_{value} was ∞ has been described above. At this instance, the robot had to rotate, and the walking command was set to $C_{\text{w, penalty}}$, which is defined as follows:

$$C_{\text{w, penalty}} = [0, 0, 14, 0.3, 0.35]^T.$$

If time T_{SS} was long, the navigation time was prolonged. Additionally, if ΔT was considerably short, it was difficult to evaluate the obstacle's motion. Conversely, if ΔT was substantially long, an unreliable optical flow could be obtained. Accordingly, ΔT was set to 0.2 s, and T_{SS} was determined as 0.3 s. Moreover, T_{DS} had to be longer than the image processing and optimization time. In this study, the average time and the sum of the image processing and optimization time were approximately 0.3 s. Therefore, T_{DS} was set to 0.35 s.

TABLE 1 Computing environments

Property	Details
CPU	intel® i7-7600
CPU speed	4.5 GHz
Memory	8 GB
Actual used memory	200,000 KB
Operating system	MS windows 7
Programming language	C++

The goal point of the robot was initially located at (0, 4). In Figure 7, d^1, d^2, and d^3 constitute the e-puck mobile robot model, which represented a dynamic obstacle in the simulation. The initial position of d^1 was (0.6, 1.1), and the velocity, which changed with a time interval of 25 s, was ±0.06 m/s in the x-axis direction. The initial positions of d^2 and d^3 were (0, 0.7) and (−0.2, 3.3), respectively. The velocities of the dynamic obstacles d^2 and d^3 with time intervals of 18 s changed by ±0.06 m/s in the y-axis direction and by ±0.06 m/s in the x-axis $-\frac{\sqrt{2}}{2}$ and y-axis $-\frac{\sqrt{2}}{2}$ unit vector directions, respectively.

Table 1 shows the computation environment used in the simulation. The simulation computer was equipped with Intel's i7 CPU. The actual amount of memory used to perform the algorithm was approximately 200 MB and varied depending on the resolution of the image obtained through the camera.

6.3 | Image processing result

Figure 8 shows the robot's image processing scenes during navigation. Specifically, Figure 8A shows the current image I_{C}, while Figure 8B shows the previous image I_{P}. Figure 8C is the ground image I_{g} of the current image I_{C}. Figure 8D shows the optical flow (I_{of}) of the previous image (I_{P}) and the current image (I_{C}). Figure 8E shows only the boundary between the ground and the obstacle in

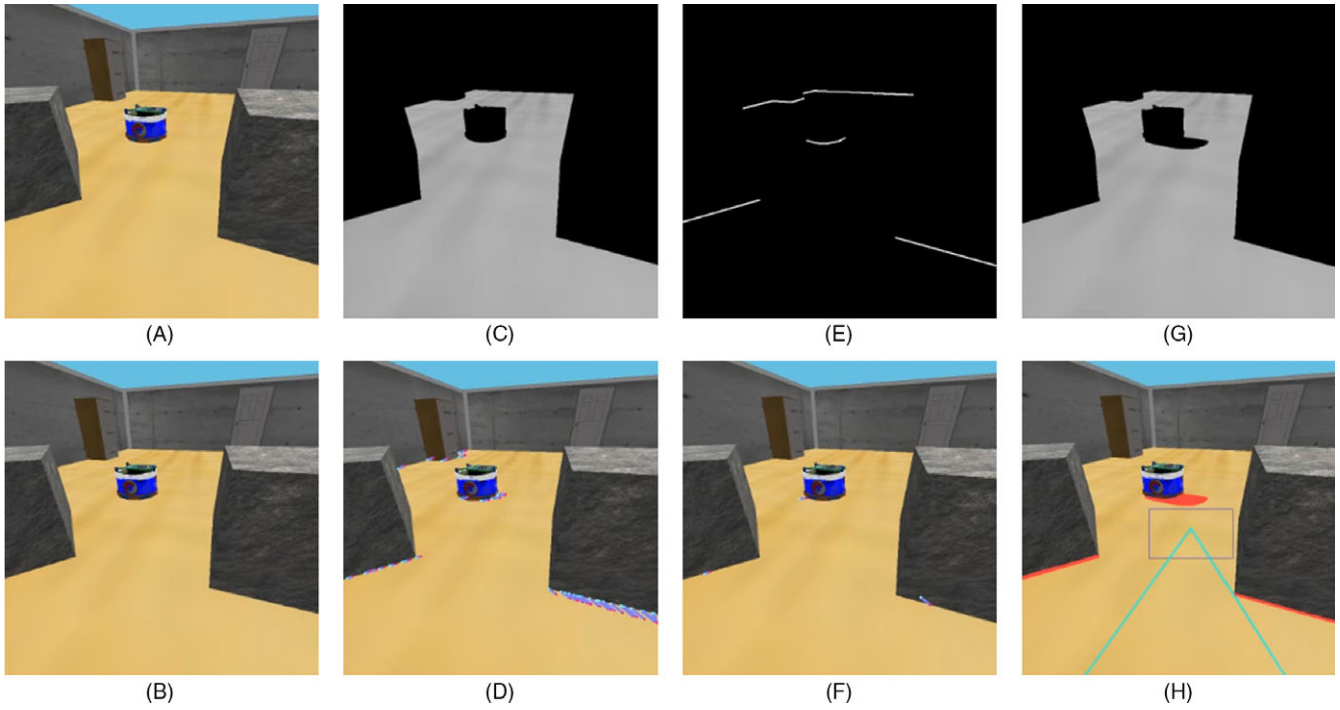

FIGURE 8 Image processing result: (A) current image, (B) previous image, (C) ground image, (D) optical flow, (E) obstacle edge detection, (F) minimum optical flow, (G) result image, and (H) confirmation image

I_g, which made it possible to distinguish the obstacles separately. Figure 8F shows only the output of the smallest optical flows in the image to distinguish the obstacles. For each obstacle in the minimum optical flow image (I_{of}^{min}), the minimum optical flow was converted into a positional change in the world coordinate system. This change was applied to the boundary image and marked by the expected movement of the obstacle in the image. Figure 8G shows the combined movement of the obstacle and ground image (I_g). Thereby, the expected position of the dynamic obstacle in the image indicates the regions where the robot could not move. Figure 8G is the image resulting from image processing. Figure 8H shows the intermediate process during the navigation.

6.4 | Optimization result

The variables used in the PSO algorithm are listed in Table 2. The result of the PSO algorithm was a pixel in the image. A 720 × 720 camera was used in the simulation. Hence, the *x*-axis ranged from 0 to 719. The range of the *y*-axis was set from 120 to 719 because the region where y was less than 120 could not be the ground, owing to the camera's field of view and the degree of pitch of the robot's camera. In the PSO algorithm, the sum of self-confidence and swarm confidence was generally equal to 4. Therefore, each value was set to 2. The setting of the inertia factor determined if the global search or the local search would have to be weighed further. In this study, because the search area was not large, a value of 2 was set, with emphasis on a more detailed search. This was equivalent to moving two pixels according to the inertia factor. The number of particles was changed to 20, 30, and 50, while the number of epochs was changed to 50, 100, and 150. Therefore, the particle and epoch numbers were determined such that the optimization could be completed within time T_{DS}.

Figure 8H shows the optimization results and estimated position of the obstacle in the image. The results of the penalty functions are also shown in the image. The area marked in red is the optical flow and indicates the expected position of the obstacles during the robot's next step. In the case of a static obstacle, the position appeared at the edge of the obstacle, whereas, for a dynamic obstacle, the next expected position was displayed on the ground. The vertex of the green triangle in the image represented the optimal point obtained through the PSO algorithm. The blue rectangle indicated the edge of the robot when it finally reached the optimal point, while the two solid green lines indicated the edge of the robot's width in the movement path. Thereby, it was confirmed that a collision with an obstacle did not occur while the robot was moving. Navigation was performed through the generation of a walking command directed at the optimal point resulting from optimization.

TABLE 2 Particle swarm optimization parameters

Parameter	Range or value
Input1 $x_{particle}$ range	[0, 719]
Input2 $y_{particle}$ range	[120, 719]
Number of particles	50
Number of epochs	100

6.5 | Navigation simulation

6.5.1 | Navigation within indoor environment with dynamic obstacles

The initial position of the goal point was (0, 4). After 70 s, the position of the goal point changed to (−1, −0.2); after 130 s, it changed to (1, 0.1); after 150 s, it changed to (−1, 4); after 200 s, it changed to (1, 0). The robot avoided the obstacle and moved to the goal point. Figure 9 shows a snapshot of the navigation simulation. Starting from the initial position, the robot avoided the dynamic obstacles during its movement. Through optimization, the optimal point was selected as the point closest to the goal among the areas where the robot was similarly capable of avoiding a dynamic obstacle. When the robot encountered a dynamic obstacle, it moved without colliding with that obstacle.

6.5.2 | Various obstacle shapes and real-time goal change

In this simulation, the navigation was performed in a narrow environment as the robot moved toward the goal point. Figure 10 shows a snapshot corresponding to real-time navigation, even if the goal point changed. Thus, it was demonstrated that the proposed algorithm is capable of coping with a goal changing in real-time. In Figure 10, the obstacles had polygonal and hemispherical shapes. However, because the navigation algorithm only used the ground as the environmental information in this study, it was observed that the navigation was not influenced by the obstacle's shape. The obstacles were avoided and the movement of the robot was directed to the goal point. When a dynamic obstacle was encountered, the robot moved without colliding with that obstacle.

Figure 11 shows the computation time of the algorithm during the simulation. In this simulation, the robot took a

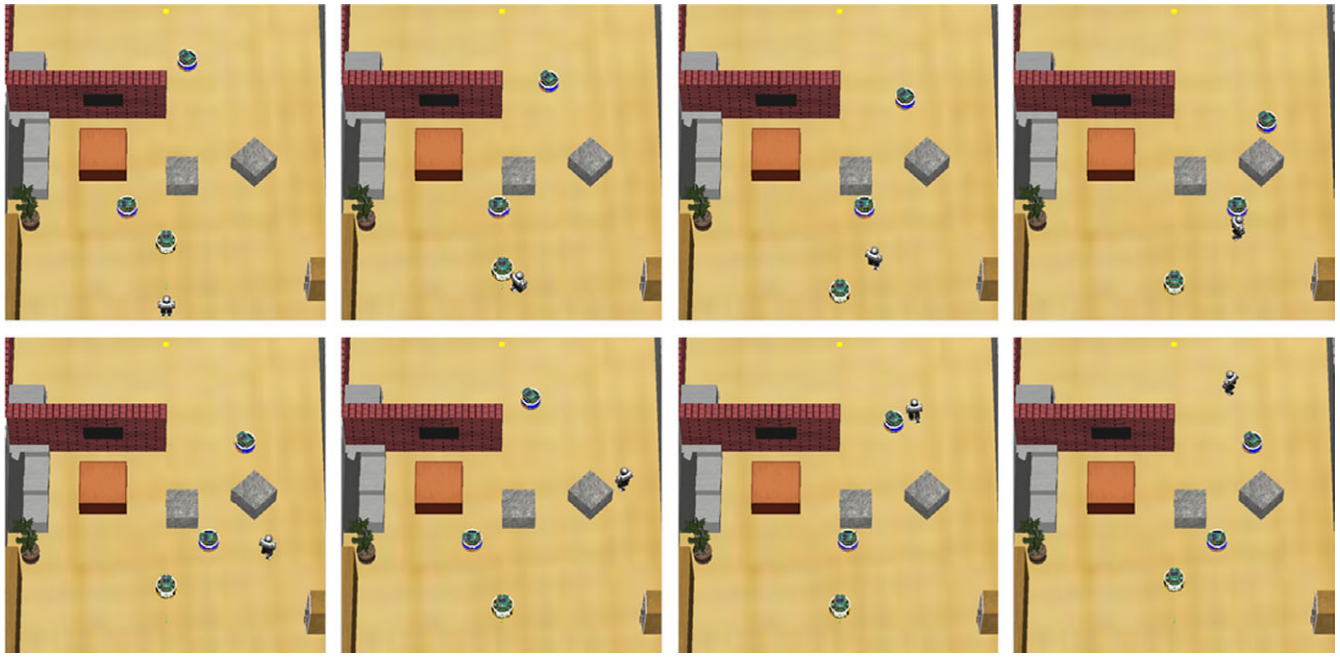

FIGURE 9 Simulation within indoor environment with dynamic obstacles (from left to right, and top to bottom)

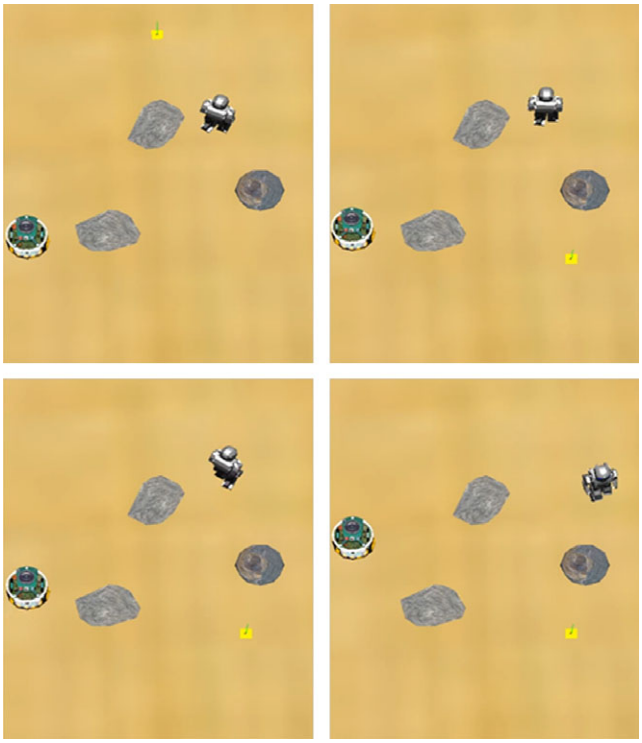

FIGURE 10 Real-time target point change simulation (from left to right, and top to bottom)

total of 124 steps. The total computation time, which is the combined time of image processing and optimization during the simulation, was less than T_{DS}. We confirmed that the defined period of 0.35 s satisfied the computation time required for navigation.

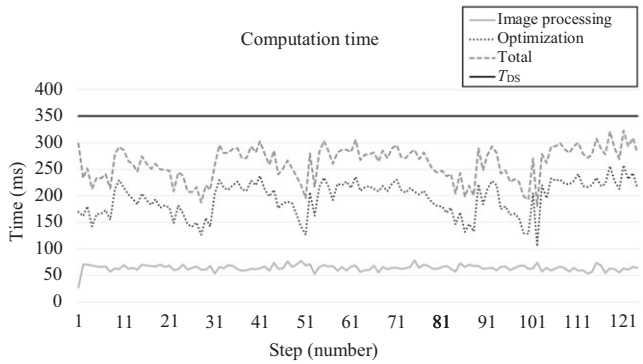

FIGURE 11 Computation time of various obstacle shapes and real-time goal change simulation

7 | CONCLUSIONS

In this paper, an algorithm using a monocular camera for the navigation of a biped humanoid robot was proposed. The information received from the monocular camera was subjected to image processing and the processed image was optimized by an optimization algorithm. The optimal point through which the robot could move was selected by implementing an optimization process. Thereafter, the walking command of the robot heading towards the optimal point was generated and input to the robot. To confirm the algorithm's usability, we simulated the proposed navigation by using a 3D dynamics simulator.

In this study, owing to the field of view limitation of the robot's camera, it was not possible to select an optimal point closer to the goal point when navigating. This

confirmed the necessity of gaze. To solve these problems, gaze control is currently being investigated [26]. Accordingly, in future work, we will develop a navigation algorithm to solve the camera's field of view problem by adding gaze control.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1C1B1006691).

ORCID


Young-Dae Hong http://orcid.org/0000-0002-6174-6442

REFERENCES

1. A. Nagariya et al., Mobile robot navigation amidst humans with intents and uncertainties: A time scaled collision cone approach, *Proc. IEEE Annu. Conf. Decision Contr.*, Osaka, Japan, Dec. 15–18, 2015, pp. 2773–2779.
2. K. Qian et al., Mobile robot navigation in unknown corridors using line and dense features of point clouds, *Proc. Annu. Conf. IEEE Ind. Electron. Soc.*, Yokohama, Japan, Nov. 9–12, 2015, pp. 1831–1836.
3. S. Mehmood et al., Stereo-vision based autonomous underwater navigation – The platform SARSTION, *Proc. Int. Bhurban Conf. Appl. Sci. Technol.*, Islamabad, Pakistan, Jan. 12–16, 2016, pp. 554–559.
4. J. Garimort, A. Hornung, and M. Bennewitz, Humanoid navigation with dynamic footstep plans, *Proc. IEEE Int. Conf. Robotics Autom.*, Shanghai, China, May 9–13, 2011, pp. 3982–3987.
5. P. Karkowski and M. Bennewitz, Real-time footstep planning using a geometric approach, *Proc. IEEE Int. Conf. Robotics Autom.*, Stockholm, Sweden, May 16–21, 2016, pp. 1782–1787.
6. H. Li et al., A humanoid robot localization method for biped navigation in human-living environments, *Proc. IEEE Int. Conf. Cyber Technol. Autom., Contr., Intell. Syst.*, Sheyang, China, June 8–12, 2015, pp. 540–544.
7. A. Amanatiadis, *A multisensor indoor localization system for biped robots operating in industrial environments*, IEEE Trans. Ind. Electron. **63** (2016), no. 12, 7597–7606.
8. J. Delfin, H. M. Becerra, and G. Arechavaleta, Humanoid localization and navigation using a visual memory, *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Cancun, Mexico, Nov. 15–17, 2016, pp. 725–731.
9. S. Wen et al., Camera recognition and laser detection based on EKF-SLAM in the autonomous navigation of humanoid robot, in *Journal of Intelligent & Robotic Systems*, Springer, Netherlands, 2017, pp. 1–13.
10. D. Maier, M. Bennewitz, and C. Stachniss, Self-supervised obstacle detection for humanoid navigation using monocular vision and sparse laser data, *Proc. IEEE Int. Conf. Robotics Autom.*, Shanghai, China, May 9–13, 2011, pp. 1263–1269.
11. G. Brooks, P. Krishnamurthy, and F. Khorrami, Humanoid robot navigation and obstacle avoidance in unknown environments, *Proc. Asian Contr. Conf.*, Istanbul, Turkey, June 23–26, 2013, pp. 1–6.
12. Y. Furuta et al., Transformable semantic map based navigation using autonomous deep learning object segmentation, *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Cancun, Mexico, Nov. 15–17, 2016, pp. 614–620.
13. A. Faragasso et al., Vision-based corridor navigation for humanoid robots, *Proc. IEEE Int. Conf. Robotics Autom.*, Karlsruhe, Germany, May 6–10, 2013, pp. 3190–3195.
14. M. Ferro et al., Omnidirectional humanoid navigation in cluttered environments based on optical flow information, *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Cancun, Mexico, Nov. 15–17, 2016, pp. 75–80.
15. L. George and A. Mazel, Humanoid robot indoor navigation based on 2D bar codes: Application to the NAO robot, *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Atlanta, GA, USA, Oct. 15–17, 2013, pp. 329–335.
16. S. Kajita et al., A realtime pattern generator for biped walking, *Proc. IEEE Int. Conf. Robotics Autom.*, Washington, DC, USA, May 11–15, 2002, pp. 31–37.
17. C. C. Hsu et al., *Distance measurement based on pixel variation of CCD images*, ISA Trans. **48** (2009), no. 4, 389–395.
18. H. Kim et al., *Distance measurement using a single camera with a rotating mirror*, Int. J. Contr., Autom. Syst. **3** (2005), no. 4, 542–551.
19. Z. Zhang et al., *A novel absolute localization estimation of a target with monocular vision*, Int. J. Light Electron Opt. **124** (2013), no. 12, 1218–1223.
20. L. F. Posada et al., Floor segmentation of omnidirectional images for mobile robot visual navigation, *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Taipei, Taiwan, Oct. 18–22, 2010, pp. 804–809.
21. Y. G. Kim and H. Kim, Layered ground floor detection for vision-based mobile robot navigation, *Proc. IEEE Int. Conf. Robotics Autom.*, New Orleans, LA, USA, Apr. 26–May 1, 2004, pp. 13–18.
22. Y. Li and S. T. Birchfield, Image-based segmentation of indoor corridor floors for a mobile robot, *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Taipei, Taiwan, October 18–22, 2010, pp. 837–843.
23. J. L. Barron, D. J. Fleet, and S. S. Beauchemin, *Performance of optical flow techniques*, Int. J. Comput. Vision **12** (1994), no. 1, 43–77.
24. J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. IEEE Int. Conf. Neural Netw.*, Perth, Australia, Nov. 27–Dec. 1, 1995, pp. 1942–1948.
25. O. Michel, *Cyberbotics Ltd. Webots™: Professional mobile robot simulation*, Int. J. Adv. Robotic Syst. **1** (2004), 39–42.
26. J. K. Yoo and J. H. Kim, *Gaze control-based navigation architecture with a situation-specific preference approach for humanoid robots*, IEEE/ASME Trans. Mechatron. **20** (2004), no. 5, 2425–2436.

AUTHOR BIOGRAPHIES

Young-Joong Han received his BS and MS degrees in electrical and computer engineering from Ajou University, Suwon, Rep. of Korea in 2016 and 2018, respectively. He is currently a research engineer for TmaxSoft. His current research interests include humanoid robotics, in particular, humanoid robot navigation and optimization-based robot control.

In-Seok Kim received his BS and MS degrees in electrical and computer engineering from Ajou University, Suwon, Rep. of Korea in 2016 and 2018, respectively. He is currently a research engineer for Hyundai Robotics, Yongin, Rep. of Korea. His current research interests include humanoid robotics, in particular, bipedal walking pattern generation & control, and path planning for robot navigation.

Young-Dae Hong received his BS, MS, and PhD degrees in electrical engineering from KAIST, Daejeon, Rep. of Korea, in 2007, 2009, and 2013, respectively. Since 2014, he has been with the Department of Electrical and Computer Engineering, Ajou University, Suwon, Rep. of Korea, where he is currently an assistant professor. His current research interests include humanoid robotics, in particular, bipedal walking pattern generation and control, footstep planning, and optimization-based robot control.