IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 12, 2022, accepted February 2, 2022, date of publication February 8, 2022, date of current version February 16, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3150036

Local Path Planning: Dynamic Window Approach With Virtual Manipulators Considering Dynamic Obstacles

MASATO KOBAYASHI, (Graduate Student Member, IEEE), AND NAOKI MOTOI, (Member, IEEE)
Graduate School of Maritime Sciences, Kobe University, Kobe 658-0022, Japan
Corresponding author: Masato Kobayashi (171w951w@stu.kobe-u.ac.jp)

This work was supported in part by JSPS KAKENHI under Grant 19K04454.

ABSTRACT Local path planning considering static and dynamic obstacles for a mobile robot is one of challenging research topics. Conventional local path planning methods generate path candidates by assuming constant velocities for a certain period time. Therefore, path candidates consist of straight line and arc paths. These path candidates are not suitable for dynamic environments and narrow spaces. This paper proposes a novel local path planning method based on dynamic window approach with virtual manipulators (DWV). DWV consists of dynamic window approach (DWA) and virtual manipulator (VM). DWA is the local path planning method that performs obstacle avoidance for static obstacles under robot constraints. DWA also generates straight line and arc path candidates by assuming constant velocities. VM generates velocities of reflective motion by using virtual manipulators and environmental information. DWV generates path candidates by variable velocities modified by VM and predicted positions of static and dynamic obstacles. Therefore, in an environment with dynamic obstacles, the obstacle-avoidable paths which include non-straight line and non-arc paths are generated. The effectiveness of the proposed method was confirmed from simulation and experimental results.

INDEX TERMS Path planning, motion planning, collision avoidance, mobile robot, dynamic window approach.

I. INTRODUCTION

In recent years, research on service robots have been active due to infectious diseases and aging population. Service robots will be used more and more in the future to create a labor force instead of people [1]–[3]. These service robots are required to move around in dynamic environments and narrow spaces where robots can coexist with people [4].

Autonomous mobile robot systems are generally composed of localization [5], mapping [6], perception [7] and path planning [8]. Path planning has to consider static and dynamic obstacles. Path planning is roughly classified into two types; global path planning and local path planning. Global path planning generates a path that does not collide with obstacles based on a prior map [9]–[11]. However, global path planning cannot consider obstacles that do not exist in the prior map. Therefore, local path planning generates paths from sensors information in real-time. Local path planning deals with obstacles that are not on the prior map [12], [13].

This paper focuses on local path planning methods. Many local path planning methods were reported for the collision avoidance against static and dynamic obstacles [14], [15]. Fiorini and Shiller proposed a local path planning method for multi-robots called Velocity Obstacle (VO) [16]. VO considers dynamic obstacle avoidance by using velocity spaces. Berg *et al.* expanded VO for smooth and vibration-free path planning [17]. Xu *et al.* reported VO considering obstacles velocities that are larger than the maximum speed of the robot [18]. There are useful local path planning methods based on VO for multi-robot systems [16]–[19]. However, when VO methods generate velocity of obstacle-avoidable path candidates by using velocity space, robots velocities are assumed as constant. Thus, the path candidates of robots

The associate editor coordinating the review of this manuscript and approving it for publication was Liang Hu.

are a straight line and arc paths under the non-holonomic constraint. Therefore, the path candidates of robots decrease in narrow and dynamic environments because the robot collides with obstacles. These VO methods also do not consider dynamics constraints.

There are local path planning methods considering kinematics and dynamics constraints. Fox *et al.* reported dynamic window approach (DWA) [20]. DWA generates path candidates by using the velocity space with dynamics constraints (VSD). VSD is the velocity space that the robot can generate from current velocities. DWA selects the optimal path from path candidates. Dobrevski *et al.* reported local path planning based on DWA and deep reinforcement learning to improve path optimization [21]. Liu *et al.* developed the global dynamic path planning fusion algorithm combining jump-A* algorithm and DWA [22]. In addition, several useful local path planning methods based on DWA were reported [20]–[22]. However, DWA generates path candidates by assuming constant velocities for a certain period time. Therefore, path candidates are straight line and arc paths. These path candidates are not suitable in dynamic environments and narrow spaces, since these path candidates do not consider obstacles. In other words, many path candidates with collisions are generated in dynamic environments and narrow spaces.

There are local path planning methods to generate the path candidates including non-straight and non-arc paths. Howard *et al.* proposed State Lattice Planner (SLP) [23], [24]. SLP generates the path candidates by using data sets of robot states and robot constraints. However, SLP may not generate path candidates if there are obstacles near the robot. We reported the local path planning method based on virtual manipulators (VM) and DWA [25]. VM was reported by Yamazaki and Inaba [26]. In narrow spaces with static obstacles, our method achieved better results compared with DWA. However, our method did not consider dynamic obstacles and guarantee path candidates at velocities within VSD.

In order to consider static and dynamic obstacles and guarantee path candidates at velocities within VSD, this paper proposes dynamic window approach with virtual manipulators (DWV). DWV is a novel generation method of path candidates including non-straight and non-arc paths. DWV generates path candidates by variable velocities modified by VM and predicted positions of static and dynamic obstacles. Therefore, in an environment with static and dynamic obstacles, the obstacle-avoidable paths which include non-straight line and non-arc paths are generated. Extensive simulations and experiments were conducted to verify the effectiveness of DWV.

This paper consists of seven sections including this one. Section II shows the modeling of the robot. Section III explains DWA as the conventional method. Section IV proposes DWV. In Sections V and VI, simulation and experimental results are shown to confirm the usefulness of the proposed method. Section VII concludes this paper.

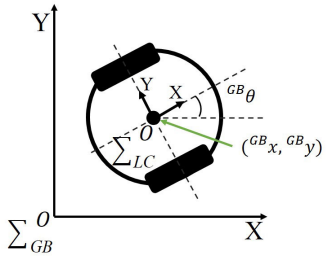


FIGURE 1. Coordinate system of mobile robot.

II. MODELING OF WHEELED MOBILE ROBOT

Fig. 1 shows the coordinate system of the robot. This paper defines the local coordinate system Σ_{LC} and the global coordinate system Σ_{GB}. The value in the global coordinate system is expressed as the superscript $^{GB}\bigcirc$. The variable of the local coordinate system does not have the superscript. The origin of the global coordinate system is set as an initial robot position. The origin of the local coordinate system is set as the center point of both wheels. As shown in Fig. 1, $(^{GB}x,\ ^{GB}y)$ and the angle $^{GB}\theta$ refer to the position and angle of the robot in the global coordinate system. The velocities of the global coordinate system $^{GB}\dot{x}$, $^{GB}\dot{y}$, and $^{GB}\dot{\theta}$ are derived as follows.

$$^{GB}\dot{\theta} = \omega \tag{1}$$

$$^{GB}\dot{x} = v\cos{}^{GB}\theta \tag{2}$$

$$^{GB}\dot{y} = v\sin{}^{GB}\theta \tag{3}$$

where v and ω are translational and angular velocities. $^{GB}\theta$ and $(^{GB}x, ^{GB}y)$ after t seconds are calculated as follows.

$$^{GB}\theta = \int_0^t \omega\, dt \tag{4}$$

$$^{GB}x = \int_0^t v\cdot\cos{}^{GB}\theta\, dt \tag{5}$$

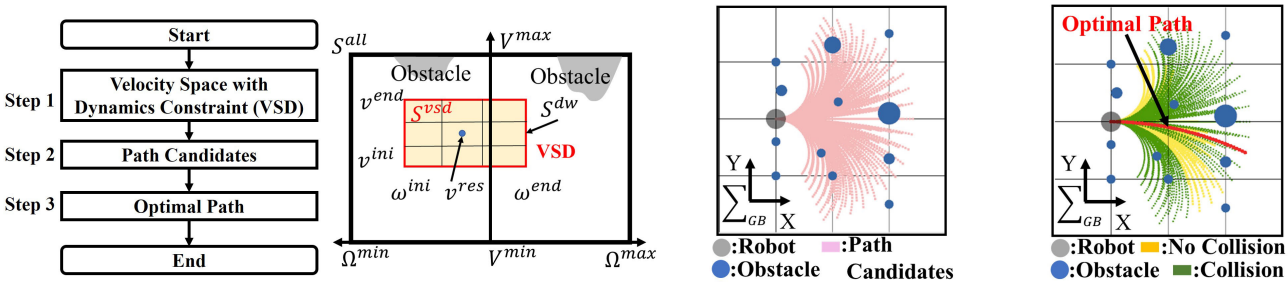
$$^{GB}y = \int_0^t v\cdot\sin{}^{GB}\theta\, dt \tag{6}$$

III. DYNAMIC WINDOW APPROACH (DWA)

A. OVERVIEW OF DWA

DWA is one of the practical local path planning methods [20]. Fig. 2 shows the overview of DWA. Fig. 2(a) shows the flowchart of DWA. In this paper, DWA is implemented by 3 steps.

- *Velocity Space with Dynamics Constraint (Step 1)*
  As shopwn in Fig. 2(b), the velocity space with dynamics constraint (VSD) is generated from current robot velocities and robot specifications.
- *Path Candidates (Step 2)*
  As shown in Fig. 2(c), DWA generates path candidates by assuming constant velocities within VSD. The color of path candidates is pink.
- *Optimal Path (Step 3)*



(a) Flowchart of DWA (b) Step1:VSD (c) Step2:Path Candidates (d) Step3:Optimal Path

FIGURE 2. Overview of DWA.

As shown in Fig. 2(d), DWA selects the optimal path from path candidates by the cost function. The color of the optimal path, collision paths, and no collision paths are red, green, and yellow. By using the velocity of the optimal path, the robot moves while avoiding obstacles.

The detail of each step is expressed in subsections III.B to III.D.

B. VELOCITY SPACE WITH DYNAMICS CONSTRAINT (STEP 1)

Fig. 2(b) shows the image diagram of VSD. In this paper, DWA generates VSD by three steps.

1) STEP 1-1: ALL VELOCITIES SPACE S^{all}

All of the velocity space S^{all}, which is determined from the robot specification, is described as follows.

$$S^{all} = \{(v, \omega) | v \in [V^{min}, V^{max}] \wedge \omega \in [\Omega^{min}, \Omega^{max}]\} \quad (7)$$

where V^{min}, V^{max}, Ω^{min} and Ω^{max} are the minimum and maximum translational and angular velocities.

2) STEP 1-2: VELOCITIES SPACE OF DYNAMIC WINDOW S^{dw}

The velocities space of dynamic window S^{dw}, which is the velocities the robot can generate velocities until the next time step, is expressed as follows.

$$S^{dw} = \{(v, \omega) | v \in [v^{res} - A^{max}\Delta T, v^{res} + A^{max}\Delta T] \wedge \omega \in [\omega^{res} - \Pi^{max}\Delta T, \omega^{res} + \Pi^{max}\Delta T]\} \quad (8)$$

where v^{res} and ω^{res} are translational and angular velocity responses as the current velocities. A^{max} and Π^{max} are maximum translational and angular acceleration. ΔT represents time step.

3) STEP 1-3: VELOCITY SPACE WITH DYNAMICS CONSTRAINT S^{vsd}

As shown in Fig. 2(b), S^{vsd} is as follows.

$$S^{vsd} = S^{all} \cap S^{dw} \quad (9)$$

$$= \{(v, \omega) \mid v \in [v^{ini}, v^{end}] \; wedge \; \omega \in [\omega^{ini}, \omega^{end}]\} \quad (10)$$

where v^{ini} and ω^{ini} are minimum translational and angular velocities in S^{vsd}. v^{end} and ω^{end} are maximum translational and angular velocities in S^{vsd}.

C. GENERATION OF PATH CANDIDATES (STEP 2)

Algorithm 1 shows the pseudo code of path candidates by DWA (Step 2). N^{tlv}, N^{agv}, and N^{all} mean the maximum number for translational velocity, angular velocity, and path candidates. DWA divides the velocity space S^{vsd} into N^{tlv} for translational velocity axis and N^{agv} for angular velocity axis. N^{all} constant velocity pairs are generated from ($N^{tlv} \cdot N^{agv}$). Thus, N^{all} constant velocity pairs generates N^{all} path candidates of the robot $\boldsymbol{O}^{rob}$. Path candidates of the robot $\boldsymbol{O}^{rob}$

Algorithm 1 Path Candidates by DWA

```
// Initialization
g ← 1, i ← 1
// ▼Step 2-1a
while N^tlv > g do
    h ← 1
    v^dwa ← v^ini + Δv(g − 1)
    // ▼Step 2-2a
    while N^agv > h do
        ω^dwa ← ω^ini + Δω(h − 1)
        // ▼Step 2-3a
        while f^max > f do
            < x_f^rob, y_f^rob, θ_f^rob > ← onePath(v^dwa, ω^dwa)
            < X_i^rob, Y_i^rob, Θ_i^rob > ← < x_f^rob, y_f^rob, θ_f^rob >
            f ← f + 1
        end while
        P_i^rob ← < X_i^rob, Y_i^rob, Θ_i^rob >
        O^rob ← P_i^rob
        i ← i + 1
        h ← h + 1
    end while
    g ← g + 1
end while
return O^rob
```

are expressed as follows.

$$\boldsymbol{O}^{rob} = [\boldsymbol{P}_1^{rob} \cdots \boldsymbol{P}_i^{rob} \cdots \boldsymbol{P}_{N^{all}}^{rob}]^T \tag{11}$$

$$\boldsymbol{P}_i^{rob} = [\boldsymbol{X}_i^{rob}\ \boldsymbol{Y}_i^{rob}\ \boldsymbol{\Theta}_i^{rob}]^T \tag{12}$$

where $\boldsymbol{P}_i^{rob}$ represents the i-th path candidate of the robot. $\boldsymbol{X}_i^{rob}$, $\boldsymbol{Y}_i^{rob}$, and $\boldsymbol{\Theta}_i^{rob}$ represent position and attitude of the robot. The detail of Algorithm 1 is explained as follows.

1) STEP 2-1A (ALGORITHM 1 LINES 3–6)

DWA divides the velocity space S^{vsd} into N^{tlv} for the translational velocity axis. The translational velocity v^{dwa} is selected as follows.

$$v^{dwa} = v^{ini} + \Delta v(g-1) \tag{13}$$

where Δv means step translational velocity. g $(1 \leq g \leq N^{tlv})$ represents number for translational velocity.

2) STEP 2-2A (ALGORITHM 1 LINES 7–9)

DWA divides the velocity space S^{vsd} into N^{agv} for angular velocity axis. The angular velocity ω^{dwa} is selected as follows.

$$\omega^{dwa} = \omega^{ini} + \Delta\omega(h-1) \tag{14}$$

where $\Delta\omega$ means step angular velocity. h $(1 \leq h \leq N^{agv})$ represents number for angular velocity.

3) STEP 2-3A (ALGORITHM 1 LINES 10–14)

The i-th path candidate $\boldsymbol{P}_i^{rob}$ is calculated from (v^{dwa}, ω^{dwa}). In Algorithm 1 (line 12), $onePath(v^{dwa}, \omega^{dwa})$ is expressed as follows.

$$\boldsymbol{X}_i^{rob} = [x_1^{rob} \cdots x_f^{rob} \cdots x_{f^{max}}^{rob}]^T \tag{15}$$

$$\boldsymbol{Y}_i^{rob} = [y_1^{rob} \cdots y_f^{rob} \cdots y_{f^{max}}^{rob}]^T \tag{16}$$

$$\boldsymbol{\Theta}_i^{rob} = [\theta_1^{rob} \cdots \theta_f^{rob} \cdots \theta_{f^{max}}^{rob}]^T \tag{17}$$

$$\theta_f^{rob} = \sum_{k=1}^{f} \int_{t_{k-1}}^{t_k} \omega^{dwa}\, dt \tag{18}$$

$$x_f^{rob} = \sum_{k=1}^{f} \int_{t_{k-1}}^{t_k} \upsilon^{dwa} \cdot \cos\theta_k\, dt \tag{19}$$

$$y_f^{rob} = \sum_{k=1}^{f} \int_{t_{k-1}}^{t_k} \upsilon^{dwa} \cdot \sin\theta_k\, dt \tag{20}$$

where f $(1 \leq f \leq f^{max})$ means number for time step. $f_{max} = \frac{T^{max}}{\Delta T}$ is calculated from the predicted time T^{max} and time step ΔT.

By repeating these processes N^{all} times, path candidates of the robot $\boldsymbol{O}^{rob}$ are generated.

D. OPTIMAL PATH (STEP 3)

Path candidates of the robot $\boldsymbol{O}^{rob}$ are evaluated by the cost function. The cost function is derived as follows.

$$c^{dwa} = W^{ang} \cdot c^{ang} + W^{vel} \cdot c^{vel} + W^{obs} \cdot c^{obs} \tag{21}$$

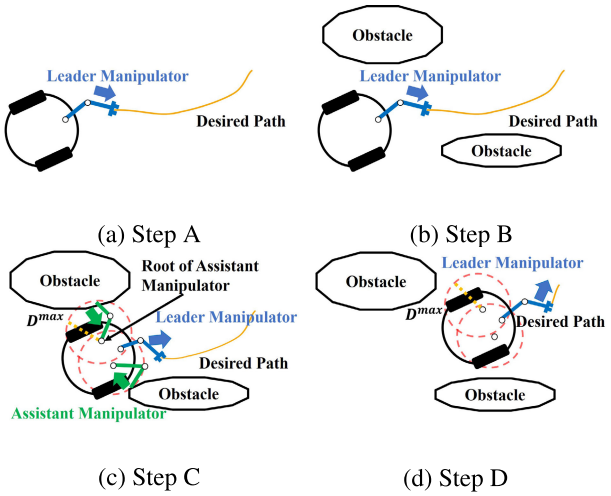


FIGURE 3. Concept of virtual manipulators method.

where W^{ang}, W^{vel} and W^{obs} are weight coefficients. c^{ang} represents the cost of the direction to the goal. c^{vel} means the cost of the current velocity. c_{obs} is the distance from the robot and the obstacle. The optimal path is chosen from path candidates by DWA to maximize the cost function (21).

Finally, velocities of the optimal path define optimal translational and angular velocities v^{opt} and ω^{opt}. Velocity commands of DWA are expressed as follows.

$$v^{cmd} = v^{opt} \tag{22}$$

$$\omega^{cmd} = \omega^{opt} \tag{23}$$

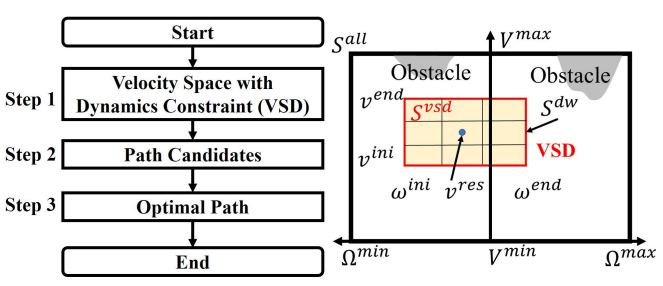
where v^{cmd} and ω^{cmd} are translational and angular velocity commands. By using velocity commands of DWA, the robot reaches the goal position during avoiding obstacles.

IV. VIRTUAL MANIPULATORS (VM)

A. CONECEPT OF VIRTUAL MANIPULATORS

The reflective motion using VM was reported by Yamazaki and Inaba [26]. VM consists of two types of manipulators; the virtual leader manipulator and virtual assistant manipulators. The virtual leader manipulator is used for path-following. Virtual assistant manipulators are used for obstacle avoidance. Fig. 3 shows the concept of VM.

1) The robot follows the desired path using the virtual leader manipulator (Fig. 3(a)).
2) The robot detects obstacles from the distance sensor (Fig. 3(b)).
3) Virtual assistant manipulators generate obstacle avoidance movement when the distance between the obstacle and roots of virtual assistant manipulators is less than the threshold value D^{max} (Fig. 3(c)).
4) Virtual assistant manipulators are deleted when the distance between the obstacle and roots of virtual assistant manipulators is larger than the threshold value D^{max} (Fig. 3(d)).



(a) Flowchart of DWV (b) Step1:VSD (c) Step2:Path Candidates (d) Step3:Optimal Path

FIGURE 4. Overview of DWV.

B. GENERATION OF VIRTUAL MANIPULATORS

The reflexible motion using virtual 2 link manipulator is calculated as follows.

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} + \Lambda(\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J})(\boldsymbol{\Theta}_{\boldsymbol{nj}}^{\boldsymbol{ref}} - \boldsymbol{\theta}_{\boldsymbol{nj}}^{\boldsymbol{res}}) \tag{24}$$

where $\dot{\boldsymbol{q}}$ is the state vector. This state vector is written as $\dot{\boldsymbol{q}} = \left[v^{vm}, \omega^{vm}, \dot{\theta}_{01}, \dot{\theta}_{02}, \dot{\theta}_{11}, \dot{\theta}_{12}, \cdots, \dot{\theta}_{n1}, \dot{\theta}_{n2}\right]^T$. v^{vm} and ω^{vm} mean the translational and the angular velocities command of the robot with virtual manipulators. $\dot{\theta}_{n1}$ and $\dot{\theta}_{n2}$ are angle velocities of n-th virtual manipulators. $\boldsymbol{J}^{\#}$ and Λ represent the pseudo-inverse Jacobian matrix and the weight coefficients of the null-space. $\boldsymbol{\Theta}_{\boldsymbol{nj}}^{\boldsymbol{ref}}$ and $\boldsymbol{\theta}_{\boldsymbol{nj}}^{\boldsymbol{res}}$ represent the reference and response angle of virtual manipulators. j represents 1st or 2nd joint number ($j = 1, 2$). The detail of the virtual manipulators method is described in the references [26].

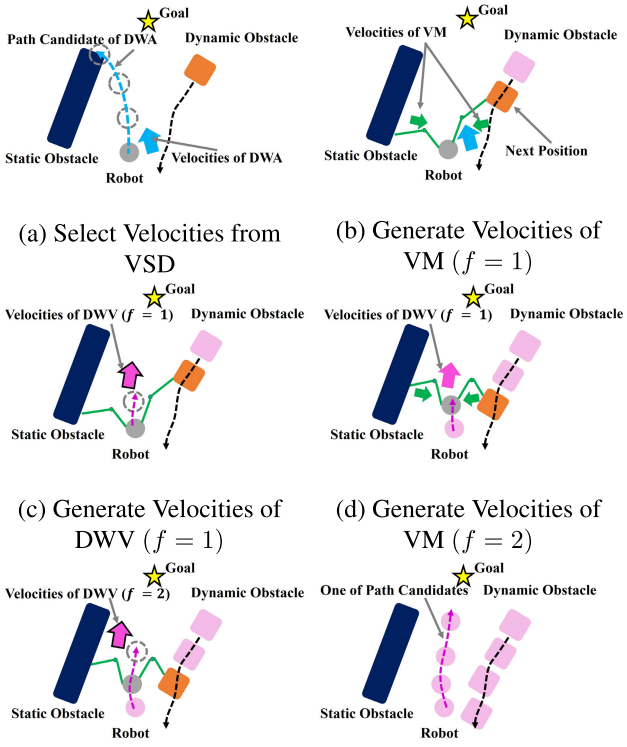
V. DYNAMIC WINDOW APPROACH WITH VIRTUAL MANIPULATORS (DWV)

A. OVERVIEW OF DWV

As shown in Figs. 2 and 4(a), the framework of the flowchart for DWV is the same as DWA. Moreover, DWV has made two changes from DWA, such as path candidates (Step 2) and optimal path (Step 3).

- *Velocity Space with Dynamics Constraint (Step 1)*
  As shown in Fig. 4(b), VSD is generated from robot velocities and specifications. VSD is expressed by (10).
- *Path Candidates (Step 2)*
  As shown in Fig. 4(c), DWV generates the path candidates by using DWA with VM. The color of path candidates is pink. By using VM and predicted obstacle position, DWV generates path candidates considering static and dynamic obstacles.
- *Optimal Path (Step 3)*
  As shown in Fig. 4(d), DWV selects the optimal path from path candidates. The color of the optimal path and no collision paths is red and yellow. DWV designed the cost function to consider dynamic obstacles. By using the velocity of the optimal path, the robot moves while avoiding obstacles.

The detail of each step is expressed in subsections V.B to V.C.



(a) Select Velocities from VSD (b) Generate Velocities of VM ($f = 1$)

(c) Generate Velocities of DWV ($f = 1$) (d) Generate Velocities of VM ($f = 2$)

(e) Generate Velocities of DWV ($f = 2$) (f) One of Path Candidates by DWV

FIGURE 5. Concept of DWV.

B. GENERATION OF PATH CANDIDATES (STEP 2)

Fig. 5 shows the concept of path candidates by DWV.

1) Firstly, DWV selects velocities from VSD. DWA also uses these velocities. In DWA, the velocities of the robot are assumed as constant. Thus, the path candidates of DWA are arc paths (Fig. 5(a)).
2) DWV uses VM for each step f to generate obstacle-avoidable paths. VM generates velocities of obstacle avoidance by using next position of obstacles (Fig. 5(b)).

3) Velocities of DWV ($f = 1$) are generated from velocities of DWA and VM (Fig. 5(c)).
4) VM generates velocities of obstacle avoidance by using next position of obstacles (Fig. 5(d)).
5) Velocities of DWV($f = 2$) are generated from previous velocities of DWV($f = 1$) and current velocities of VM (Fig. 5(e)).
6) By repeating these process f^{max} times, one of path candidates by DWV $\boldsymbol{P}_i^{rob}$ are generated.(Fig. 5(f)).

Algorithm 2 shows the pseudo code of path candidates by DWV (Step 2). The generation of path candidates by DWV is explained as follows.

Algorithm 2 Path Candidates by DWV

1: // *Initialization*
2: $g \leftarrow 1, i \leftarrow 1$
3: // ▼*Step 2-1b*
4: **while** $N^{tlv} > g$ **do**
5: $v^{dwa} \leftarrow v^{ini} + \Delta v(g-1)$
6: $h \leftarrow 1$
7: // ▼*Step 2-2b*
8: **while** $N^{agv} > h$ **do**
9: $\omega^{dwa} \leftarrow \omega^{ini} + \Delta\omega(h-1)$
10: $f \leftarrow 1$
11: **while** $f^{max} > f$ **do**
12: // ▼*Step 2-3b*
13: $\omega_f^{vm} \leftarrow calcVM(\boldsymbol{O}^{obs})$
14: // ▼*Step 2-4b*
15: $v^{dwv} \leftarrow v^{dwa}$
16: **if** $f = 1$ **then**
17: $\omega_f^{dwv} \leftarrow \omega^{dwa} + \omega_f^{vm}$
18: **else**
19: $\omega_f^{dwv} \leftarrow \omega_{f-1}^{dwv} + \omega_f^{vm}$
20: **end if**
21: // ▼*Step 2-5b*
22: $< v^{dwv}, \omega_f^{dwv} > \leftarrow checkVelocity(v^{dwv}, \omega_f^{dwv})$
23: // ▼*Step 2-6b*
24: $< x_f^{rob}, y_f^{rob}, \theta_f^{rob} > \leftarrow onePath(v^{dwv}, \omega_f^{dwv})$
25: $< \boldsymbol{X}_i^{rob}, \boldsymbol{Y}_i^{rob}, \boldsymbol{\Theta}_i^{rob} > \leftarrow < x_f^{rob}, y_f^{rob}, \theta_f^{rob} >$
26: $f \leftarrow f + 1$
27: **end while**
28: $\boldsymbol{P}_i^{rob} \leftarrow < \boldsymbol{X}_i^{rob}, \boldsymbol{Y}_i^{rob}, \boldsymbol{\Theta}_i^{rob} >$
29: // ▼*Step 2-7b*
30: **if** $collision(\boldsymbol{O}^{obs}, \boldsymbol{P}_i^{rob})$ **then**
31: $delete(\boldsymbol{P}_i^{rob})$
32: **else**
33: $\boldsymbol{O}^{rob} \leftarrow \boldsymbol{P}_i^{rob}$
34: $i \leftarrow i + 1$
35: **end if**
36: $h \leftarrow h + 1$
37: **end while**
38: $g \leftarrow g + 1$
39: **end while**
40: **return** $\boldsymbol{O}^{rob}$

1) STEP 2-1B (ALGORITHM 2 LINES 3–5)

The translational velocity v^{dwa} is selected from VSD.

$$v^{dwa} = v^{ini} + \Delta v(g-1) \tag{25}$$

2) STEP 2-2B (ALGORITHM 2 LINES 7–9)

The angular velocity ω^{dwa} is selected from VSD.

$$\omega^{dwa} = \omega^{ini} + \Delta\omega(h-1) \tag{26}$$

3) STEP 2-3B (ALGORITHM 2 LINES 12–13)

In DWV, velocities considering static and dynamic obstacles are generated by predicted positions of obstacles $\boldsymbol{O}^{obs}$ and VM. Predicted positions of obstacles $\boldsymbol{O}^{obs}$ are shown as follows.

$$\boldsymbol{O}^{obs} = [\boldsymbol{P}_1^{obs} \cdots \boldsymbol{P}_s^{obs} \cdots \boldsymbol{P}_{s^{max}}^{obs}]^T \tag{27}$$
$$\boldsymbol{P}_s^{obs} = [\boldsymbol{X}_s^{obs}\ \boldsymbol{Y}_s^{obs}]^T \tag{28}$$
$$\boldsymbol{X}_s^{obs} = [x_{s,1}^{obs} \cdots x_{s,f}^{obs} \cdots x_{s,f^{max}}^{obs}]^T \tag{29}$$
$$\boldsymbol{Y}_s^{obs} = [y_{s,1}^{obs} \cdots y_{s,f}^{obs} \cdots y_{s,f^{max}}^{obs}]^T \tag{30}$$

where s ($1 \leq s \leq s^{max}$) means the number for obstacles. In this paper, VM consisted only virtual assistant manipulators. By using minimum distance d_f^{rob} between f-th positions of obstacles and roots of VM, angular velocity ω^{vm} is calculated from (24). The f-th angular velocity ω_f^{vm} are calculated as follows.

$$\omega_f^{vm} = \begin{cases} \omega^{vm}, & if\ d_f^{rob} \leq D^{max} \\ 0, & otherwise \end{cases} \tag{31}$$

4) STEP 2-4B (ALGORITHM 2 LINES 14–20)

The f-th velocities of DWV are calculated as follows.

$$v^{dwv} = v^{dwa} \tag{32}$$
$$\omega_f^{dwv} = \begin{cases} \omega^{dwa} + \omega_f^{vm}, & if\ f = 1 \\ \omega_{f-1}^{dwv} + \omega_f^{vm}, & otherwise \end{cases} \tag{33}$$

where v^{dwv} and ω_f^{dwv} mean translational and the angular velocities of DWV.

5) STEP 2-5B (ALGORITHM 2 LINES 21–22)

The f-th velocities of DWV v^{dwv} and ω_f^{dwv} are checked within the f-th velocity space of DWV S_f^{dwv}. The f-th velocity space of DWV S_f^{dwv} consists of f-th velocity space of dynamic window S_f^{pdw} and S^{all}. The f-th velocity space of DWV S_f^{dwv} are expressed as follows.

$$S_f^{pdw} = \{(v,\omega) | v \in [v^{dwv} - A^{max}\Delta T, v^{dwv} + A^{max}\Delta T] \wedge \omega \in [\omega_{f-1}^{dwv} - \Pi^{max}\Delta T, \omega_{f-1}^{dwv} + \Pi^{max}\Delta T]\} \tag{34}$$

$$S_f^{dwv} = \begin{cases} S^{all} \cap S^{dw}, & if\ (f = 1) \\ S^{all} \cap S_f^{pdw}, & otherwise \end{cases} \tag{35}$$

$$= \{(v,\omega) \mid v \in [v_f^{ini}, v_f^{end}] \wedge \omega \in [\omega_f^{ini}, \omega_f^{end}]\} \tag{36}$$

where v_f^{ini} and ω_f^{ini} are minimum translational and angular velocities in S_f^{dwv}. v_f^{end} and ω_f^{end} are maximum translational and angular velocities in S_f^{dwv}. When v^{dwv} or ω_f^{dwv} are not within S_f^{dwv}, v^{dwv} and ω_f^{dwv} are expressed as follows.

$$v^{dwv} = \begin{cases} v_f^{ini}, & if\ v^{dwv} < v_f^{ini} \\ v_f^{end}, & if\ v^{dwv} > v_f^{end} \\ v^{dwv}, & otherwise \end{cases} \tag{37}$$

$$\omega_f^{dwv} = \begin{cases} \omega_f^{ini}, & if\ \omega_f^{dwv} < \omega_f^{ini} \\ \omega_f^{end}, & if\ \omega_f^{dwv} > \omega_f^{end} \\ \omega_f^{dwv}, & otherwise \end{cases} \tag{38}$$

6) STEP 2-6B (ALGORITHM 2 LINES 23–25)

The predicted robot position is calculated from v^{dwv} and ω_f^{dwv}. The i-th path candidate $\boldsymbol{P_i}$ is calculated as follows.

$$\boldsymbol{X}_i^{rob} = [x_1^{rob} \cdots x_f^{rob} \cdots x_{f^{max}}^{rob}]^T \tag{39}$$

$$\boldsymbol{Y}_i^{rob} = [y_1^{rob} \cdots y_f^{rob} \cdots y_{f^{max}}^{rob}]^T \tag{40}$$

$$\boldsymbol{\Theta}_i^{rob} = [\theta_1^{rob} \cdots \theta_f^{rob} \cdots \theta_{f^{max}}^{rob}]^T \tag{41}$$

$$\theta_f^{rob} = \sum_{k=1}^{f} \int_{t_{k-1}}^{t_k} \omega_k^{dwv}\, dt \tag{42}$$

$$x_f^{rob} = \sum_{k=1}^{f} \int_{t_{k-1}}^{t_k} \upsilon^{dwv} \cdot \cos\theta_k\, dt \tag{43}$$

$$y_f^{rob} = \sum_{k=1}^{f} \int_{t_{k-1}}^{t_k} \upsilon^{dwv} \cdot \sin\theta_k\, dt \tag{44}$$

By using v^{dwv} and ω_f^{dwv}, path candidates by DWV include non-straight line and non-arc paths as shown in Fig. 4(c), since DWV modifies angular velocities ω_f^{dwv} for each time step.

7) STEP 2-7B (ALGORITHM 2 LINE 29–35)

The collision between the robot and obstacles is checked. If the robot collides with obstacles, this path candidate is not used for the optimal path.

By repeating these processes N^{all} times, path candidates of the robot $\boldsymbol{O}^{rob}$ are generated.

C. OPTIMAL PATH (STEP 3)

Path candidates of the robot $\boldsymbol{O}^{rob}$ are evaluated by the cost function. DWV has adopted the distance to the goal position as a cost function to reduce the restrictions on avoiding robot obstacles. The cost function is derived as follows.

$$c^{dwv} = W^{pos} \cdot c^{pos} + W^{vel} \cdot c^{vel} + W^{sdo} \cdot c^{sdo} \tag{45}$$

where W^{pos} and W^{sdo} are the weight coefficients of position and obstacles. c^{pos} is the distance from the goal position and the predicted position of the robot at f^{max}. c^{sdo} is the shortest distance calculated from the predicted positions of the robot and obstacles. c^{sdo} is calculated for each step f. DWV considers static and dynamic obstacles in the phase of selecting an optimal path. The optimal path is selected from path candidates by DWV to maximize the cost function.

Finally, velocities of the optimal path define optimal translational and angular velocity v^{opt} and ω_f^{opt}. Velocity commands of DWV are expressed as follows.

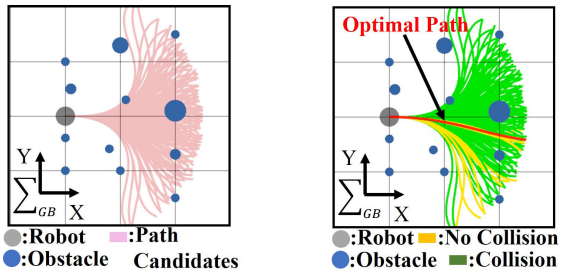
$$v^{cmd} = v^{opt} \tag{46}$$

$$\omega^{cmd} = \omega_1^{opt} \tag{47}$$

where v^{opt} and ω_1^{opt} are optimal velocities of 1st time step. By using velocity commands of DWV, the robot reaches the goal position during avoiding obstacles.

TABLE 1. Control parameters.

W^{ang}	Weight Coefficient of Angle	1
W^{vel}	Weight Coefficient of Velocity	5
W^{obs}	Weight Coefficient of Obstacle	0.1
W^{pos}	Weight Coefficient of Position	20
W^{sdo}	Weight Coefficient of Static and Dynamic Obstacle	0.1
V^{max}	Maximum Translational Velocity	0.55 $[m/s]$
V^{min}	Minimize Translational Velocity	-0.3 $[m/s]$
Ω^{max}	Maximum Angular Velocity	5 $[rad/s]$
Ω^{min}	Minimize Angular Velocity	-5 $[rad/s]$
A^{max}	Maximum Translational Acceleration	2 $[m/s^2]$
Π^{max}	Maximum Angular Acceleration	5 $[rad/s^2]$
N^{tlv}	Number of Translational Velocity	6
N^{agv}	Number of Angular Velocity	20
T^{max}	Maximum Predicted Time	4 $[s]$
ΔT	Time Step	0.1 $[s]$
D^{max}	Generation Value of Virtual Manipulators	0.5 $[m]$
L^{ass}	Length of Virtual Assistant Manipulators	0.3 $[m]$
Θ_{11}^{ref}	Reference 1st Angle of 1st Virtual Manipulators	90 $[deg]$
Θ_{12}^{ref}	Reference 2nd Angle of 1st Virtual Manipulators	-180 $[deg]$
Θ_{21}^{ref}	Reference 1st Angle of 2nd Virtual Manipulators	-90 $[deg]$
Θ_{22}^{ref}	Reference 2nd Angle of 2nd Virtual Manipulators	180 $[deg]$
Λ	Weight Coefficients of Virtual Manipulators	0.075
N^{pos}	Number of Samples in Terminal State Position	40
N^{hea}	Number of Samples in Terminal State Heading	3
L^{pat}	Path Length of SLP	2.5$[m]$
Υ^{min}	Minimum Angular Range of Terminal Position	-60$[deg]$
Υ^{max}	Maximum Angular Range of Terminal Position	60$[deg]$
Φ^{min}	Minimum Angular Range of Heading Angle	-30$[deg]$
Φ^{max}	Maximum Angular Range of Heading Angle	30$[deg]$



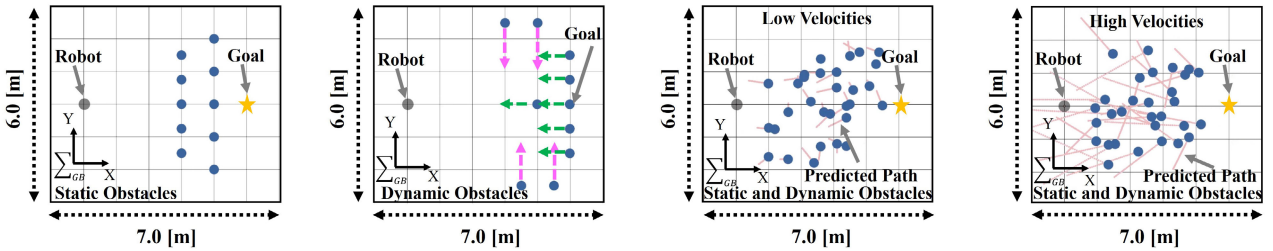
(a) Path Candidates (b) Optimal Path

FIGURE 6. Overview of SLP.

VI. SIMULATION

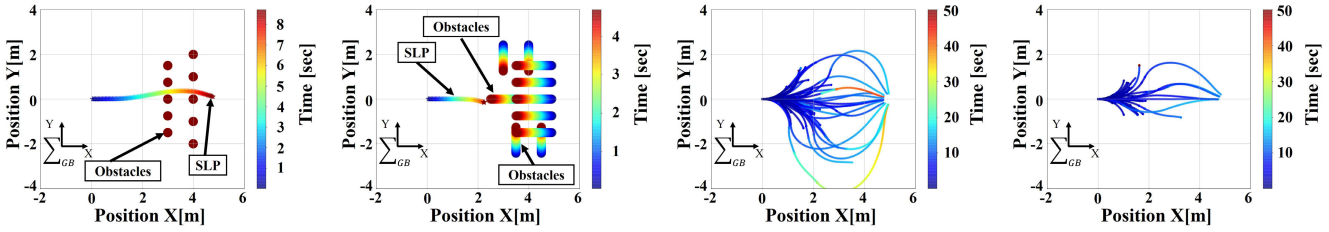
A. LOCAL PATH PLANNIG METHODS

"SLP", "DWA", and "DWV" are used as the local path planning in simulations. SLP and DWA are treated as the conventional methods. DWV is implemented as the proposed method. Table 1 shows control parameters. Robot Operating System (ROS) was used [27]–[30]. Fig. 6 shows the



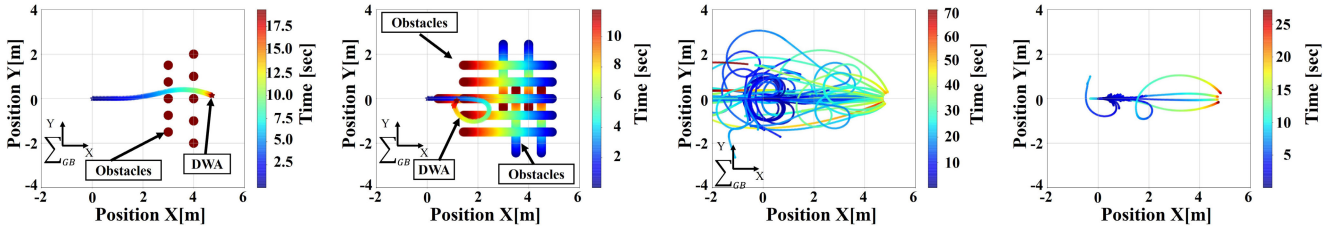
(a) Case S1 (b) Case S2 (c) Case S3 (d) Case S4

FIGURE 7. Simulation environments.



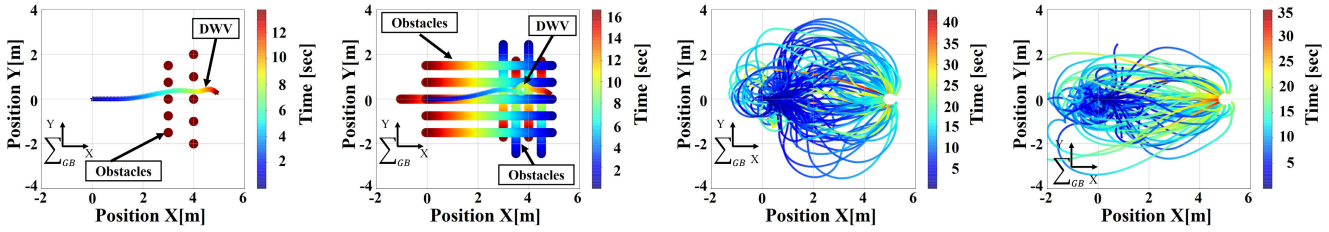
(a) Case S1 (b) Case S2 (c) Case S3 (d) Case S4

FIGURE 8. Simulation results of SLP.



(a) Case S1 (b) Case S2 (c) Case S3 (d) Case S4

FIGURE 9. Simulation results of DWA.



(a) Case S1 (b) Case S2 (c) Case S3 (d) Case S4

FIGURE 10. Simulation results of DWV.

path candidates and optimal path of SLP. Parameters of SLP path candidates are shown as follows. N^{pos}, N^{hea} and L^{pat} represent the number of samples in terminal state position and heading, and the terminal position horizon. Υ^{min}, Υ^{max}, Φ^{min} and Φ^{max} represent the angular range of the terminal position sampling and the angular range of the terminal heading angle offsets. The detail of the SLP is described in the references [23], [24]. In this simulation, DWV was equipped with two VM. The root position of 1st and 2nd VM in robot coordinate system were $(x_1^{vam}, y_1^{vam}) = (0.0, -0.1)$ and $(x_2^{vam}, y_2^{vam}) = (0.0, 0.1)$.

B. SIMULATION SETUP

As shown in Table 2, there are four simulation cases. Fig. 7 shows the simulation environment in *Cases S1-S4*. In Fig. 7, the gray and blue circles are the robot and

TABLE 2. Simulation setup.

Case	Obstacle Type	Obstacle Position	Obstacle Velocity
$CaseS1$	Static Obstacles	Constant	0
$CaseS2$	Dynamic Obstacles	Constant	Constant
$CaseS3$	Static / Dynamic Obstacles	Random	Random (Low)
$CaseS4$	Static / Dynamic Obstacles	Random	Random (High)

obstacles. In Figs. 7(c)-(d), the pink lines show the moving direction. The start and goal position of all cases are $({}^{GB}x^{start}, {}^{GB}y^{start}) = (0.0, 0.0)$ and $({}^{GB}x^{goal}, {}^{GB}y^{goal}) = (5.0, 0.0)$. When the distance between the robot and the goal position is less than 0.3 [m], it is judged as a goal. The simulation cases are defined as follows.

- *Case S1: Static Obstacles*
  As shown in Fig. 7(a), there are 10 static obstacles in the simulation environment.
- *Case S2: Dynamic Obstacles*
  As shown in Fig. 7(b), there are 10 dynamic obstacles in the simulation environment. These obstacles moves at 0.3 [m/s] (green) and 0.25[m/s] (pink).
- *Case S3: Static / Dynamic Obstacles (Low Velocities)*
  As shown in Fig. 7(c), there are 30 static and dynamic obstacles in the simulation environment. These obstacles are placed in random positions and given random velocities that are lower than maximum velocities of the robot. The velocities of obstacles were set randomly in the range of 0.0 [m/s] to 0.2 [m/s].
- *Case S4: Static / Dynamic Obstacles (High Velocities)*
  As shown in Fig. 7(d), there are 30 static and dynamic obstacles in the simulation environment. These obstacles are placed in random positions and given random velocities that are higher than maximum velocities of the robot. The velocities of obstacles were set randomly in the range of 0.0 [m/s] to 0.6 [m/s].

Case S1 and *Case S2* were simulated 1 times. *Case S3* and *Case S4* were simulated 100 times.

C. SIMULATION RESULTS

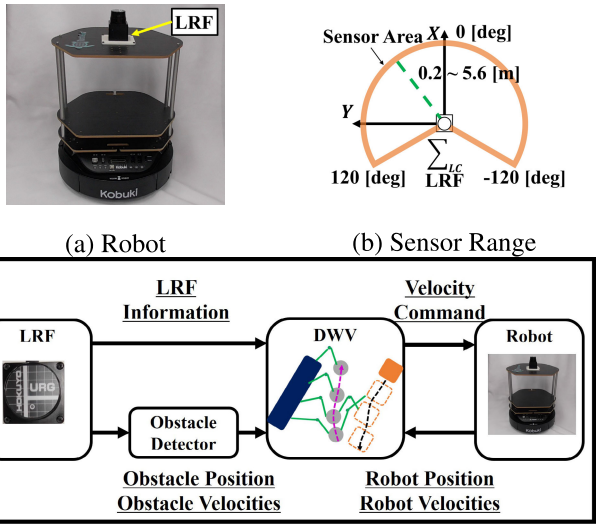
Figs. 8-10 show trajectory results of SLP, DWA and DWV in *Case S1-Case S4*. Figs. 8-10(a)(b) show trajectories results of robot and obstacles in *Case S1-Case S2*. Figs. 8-10(c)(d) show only trajectories of the robot for 100 simulation times in *Case S3-Case S4*. Table 3 shows simulation results of all cases. Table 3 contains the success rate in reaching the goal without the collision, the average travel time only in reaching the goal, the trajectory length (TL) only in reaching the goal, and the movement posture displacement (PD) only in reaching the goal.

In *Case S1*, all methods reached the goal position as shown in Figs. 8-10(a). SLP reached the goal position earlier than DWA and DWV. This is because the SLP moved at maximum translational velocity on the optimal path. From Table 3 and Fig. 8(a), the best result in *Case S1* was obtained by SLP.

In *Case S2*, DWV reached the goal position as shown in Figs. 8-10(b). From Figs. 8-9(b), SLP and DWA did not reach

TABLE 3. Simulation results.

Case	Method	Success [%]	Time [sec]	TL [m]	PD [rad]
$CaseS1$ 1 time	SLP	100	8.708	4.835	1.324
	DWA	100	19.301	4.814	0.950
	DWV	100	13.711	5.009	2.362
$CaseS2$ 1 time	SLP	0	-	-	-
	DWA	0	-	-	-
	DWV	100	16.711	5.490	5.681
$CaseS3$ 100 times	SLP	14	18.223	6.091	2.874
	DWA	32	42.556	9.678	4.398
	DWV	85	20.919	7.562	7.392
$CaseS4$ 100 times	SLP	6	12.959	5.071	2.096
	DWA	4	23.425	5.765	2.422
	DWV	70	22.135	8.003	7.877



(a) Robot (b) Sensor Range

(c) System Configuration

FIGURE 11. Experiment setup.

the goal position. SLP and DWA generated path candidates not considering dynamic obstacles. SLP and DWA considered dynamic obstacles when the optimal path was selected from path candidates. However, there were no path candidates without collision. Therefore, SLP and DWA collided with dynamic obstacles. On the other hand, DWV generated path candidates considering dynamic obstacles. DWV also considered dynamic obstacles when the optimal path was selected from path candidates. Thus, DWV reached the goal position. From Table 3 and Fig. 10(b), the best results in *Case S2* was obtained by DWV.

In *Case S3*, all method reached the goal position as shown in Figs. 8-10(c). As shown in Fig. 8(c), the goal trajectories of SLP were close to straight lines. The position and velocity of obstacles were given randomly for 100 simulation times. There were a few easy situations for SLP, such as obstacles did not move towards the robot. Fig. 9(c) shows the goal trajectories of DWA. When DWA reached the goal position with avoiding moving obstacles, DWA sometimes generated back movements. The goal time of DWA was longer than other methods. From Table 3 and Fig. 10(c), DWV had the

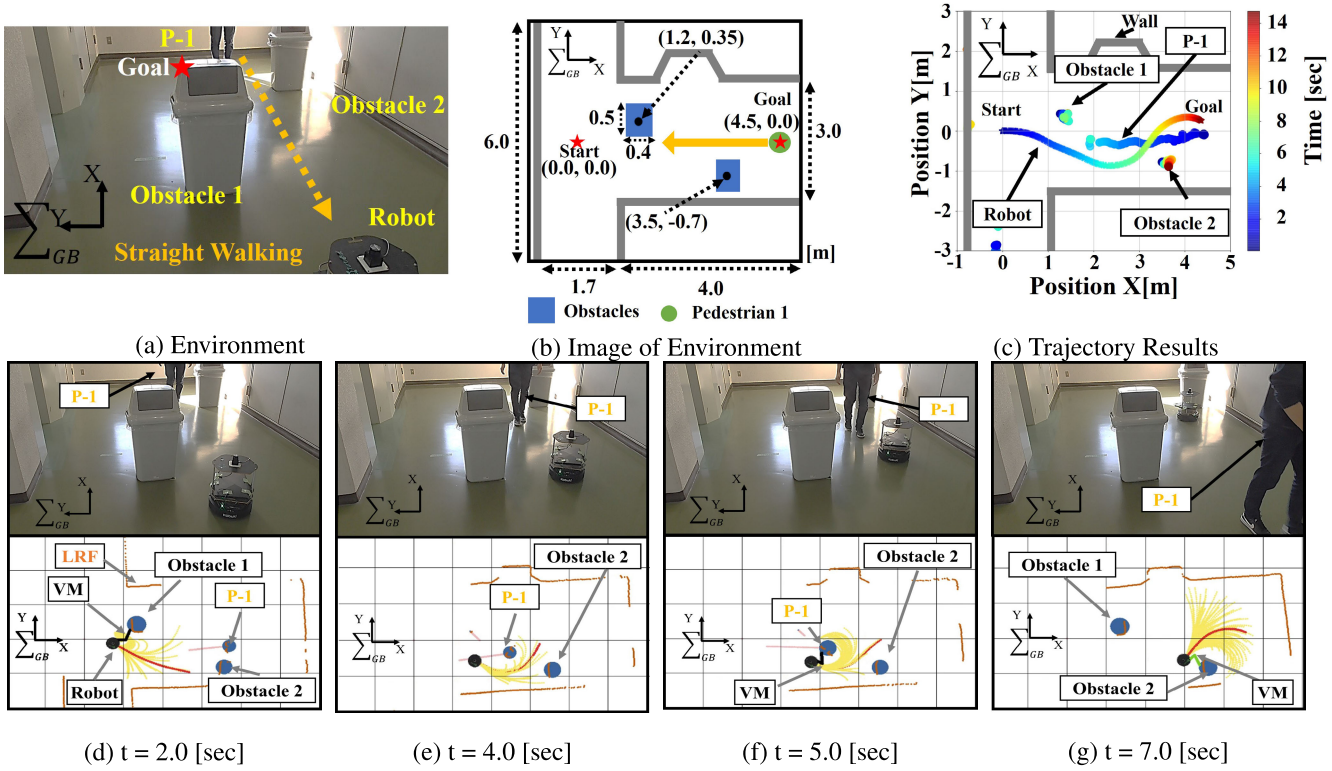


FIGURE 12. Experimental results (Case E1).

highest success rate of reaching the goal at 85%. The best result in *Case S3* was obtained by DWV.

In *Case S4*, all method reached the goal position as shown in Figs. 8-10(d). As shown in Figs. 8-9(d), the goal trajectories of SLP and DWA were close to straight lines. The position and velocity of obstacles were given randomly for 100 simulation times. There were a few easy situations for SLP and DWA, such as obstacles did not move towards the robot. Moreover, DWV sometimes collided with obstacles in *Case S4*. However, DWV had the highest success rate of reaching the goal at 70% as shown in Table 3 and Fig. 10(d). The best result in *Case S4* was obtained by DWV.

The effectiveness of the proposed method was confirmed from the simulation results.

VII. EXPERIMENT

A. EXPERIMENT SYSTEM

As shown in Fig. 11(a), Turtlebot2 [31] and ROS Melodic were used in these experiments. Turtlebot2 was equipped with the laser range finder (LRF) that was URG-04LX-UG01 [32]. The Range of LRF was shown in Fig. 11(b). Fig. 11(c) shows the system configuration. Firstly, environmental information was obtained by LRF. Secondly, the obstacle position and velocities were generated by *obstacle_detector* [33]. In these experiments, *obastacle_detector* got only static and dynamic obstacles that radius was less than 0.25 [m]. Finally, DWV generated the velocity command considering static and dynamic obstacles from *obastacle_detector* and LRF information. As shown in Table 1, the same parameters as in the simulation were set for these experiments.

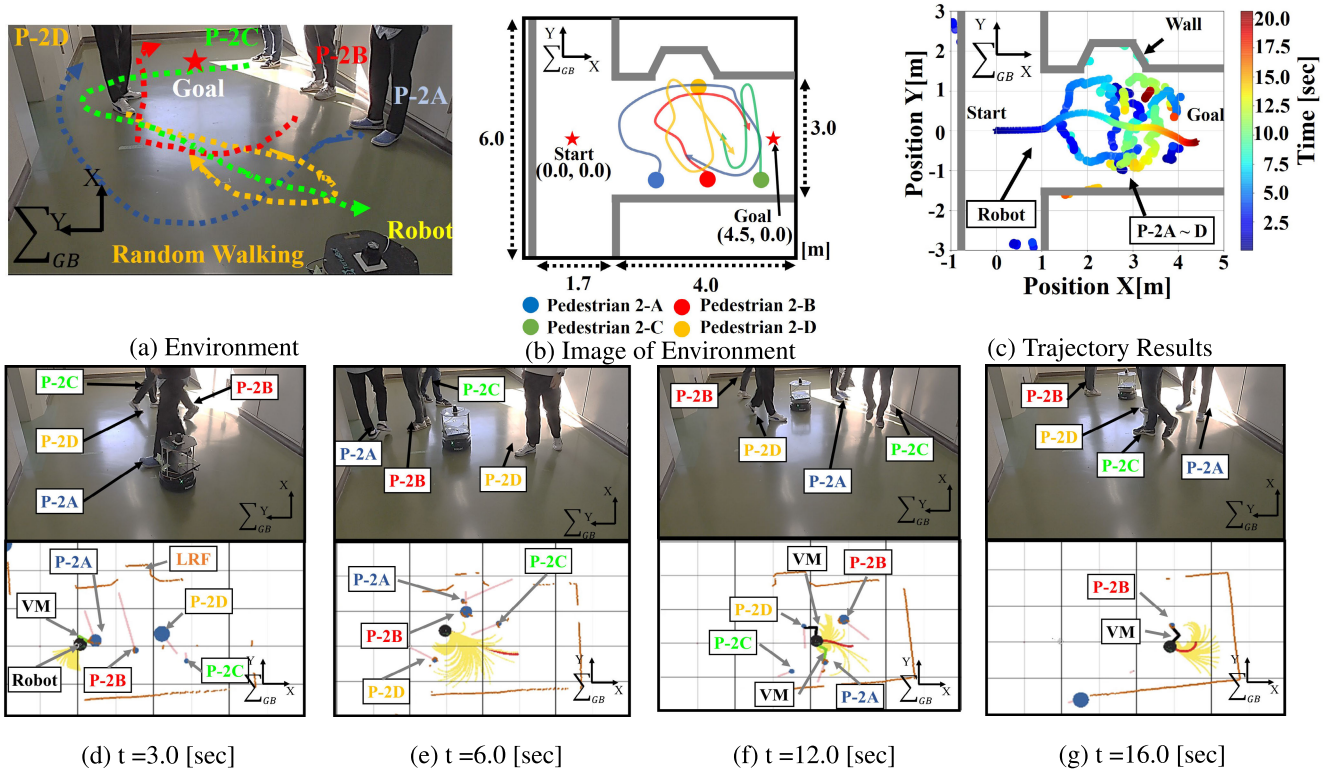
B. EXPERIMENT SETUP

There were two experiment cases. The start and goal position was $({}^{GB}x^{start}, {}^{GB}y^{start}) = (0.0, 0.0)$ and $({}^{GB}x^{goal}, {}^{GB}y^{goal}) = (4.5, 0.0)$ as shown in Figs. 12-13.

- *Case E1: Straight Walking*
  As shown in Figs. 12(a)(b), there were 2 static obstacles, the 1 pedestrian (P-1) and walls in the experiment environment. P-1 walked in the straight line in the narrow space.
- *Case E2: Random Walking*
  As shown in Figs. 13(a)(b), there were 4 pedestrians (P-2A$\sim$ 2D) and walls in the experiment environment. 4 pedestrians walked randomly in the narrow space.

C. EXPERIMENT RESULTS

Figs. 12-13 show experiments results in *Cases E1-E2*. Figs. 12-13 (c) show trajectory results in *Cases E1-E2*. Figs. 12-13 (d)-(g) show the experimental environment, path candidates and optimal path at a certain time in *Cases E1-E2*. The black and blue circles are the



(a) Environment (b) Image of Environment (c) Trajectory Results

(d) t =3.0 [sec] (e) t =6.0 [sec] (f) t =12.0 [sec] (g) t =16.0 [sec]

FIGURE 13. Experimental results (Case E2).

robot and obstacles. The black and green lines from the robot are VM. The yellow and red lines are path candidates and the optimal path of DWV. The pink lines show the predicted position of pedestrians. The orange points are LRF information.

In *Case E1*, there were 2 static obstacles, 1 pedestrian, and walls in the narrow space. As shown in Figs. 12(c)-(g), the robot reached the goal without collision. From Fig. 12(d), DWV generated path candidates considering Obstacle 1 and walls. From Fig. 12(e), the robot avoided Obstacles 1. DWV generated path candidates considering P-1, Obstacle 2, and the walls. From Fig. 12(f), DWV generated the path candidates avoiding P-1, Obstacle 2, and the walls. From Fig. 12(g), the robot avoided P-1. DWV generated path candidates considering Obstacle 2. Finally, the robot reached the goal position without collision.

In *Case E2*, there were 4 pedestrians and walls in the narrow space. As shown in Figs. 13(c)-(g), the robot reached the goal without collision. From Fig. 13(d), DWV generated path candidates considering P-2A, so the robot moved backward. From Fig. 13(e), the robot avoided P-2A. DWV generated path candidates considering P-2A∼ 2D and walls. From Fig. 13(f), the robot avoided P-2A and P-2C. DWV generated the path candidates considering P-2A, P-2B, and P-2D. From Fig. 13(g), the robot avoided P-2A, P-2B, and P-2D. DWV generated path candidates considering P-2B and walls. Finally, the robot reached the goal position without collision.

The effectiveness of the proposed method was confirmed from the experiment results.

VIII. CONCLUSION

This paper proposed the novel local path planning method called DWV. DWV was composed of DWA and VM for static and dynamic obstacle avoidance. DWV generated obstacle-avoidable path candidates which include non-straight line and non-arc paths. These path candidates were generated by variable velocities considering predicted positions of static and dynamic obstacles. DWV considered kinematics and dynamics constraints. The effectiveness of the proposed method was demonstrated by simulations and experiments.

In future works, we will work to evaluate DWV as follows.

- *Parameter Design of DWV*
  The number of parameters was increased by considering virtual manipulators. The parameter design method should be clarified.
- *DWV with Global Path Planning*
  We evaluated DWV using only the local path planning method. We will apply both DWV and global path planning.
- *Various Environments, Robots and Sensors*
  We evaluated DWV with the robot and environments. We will evaluate DWV for various robots, environments, and sensors.

REFERENCES

[1] C. Nam, S. Lee, J. Lee, S. H. Cheong, D. H. Kim, C. Kim, I. Kim, and S.-K. Park, "A software architecture for service robots manipulating objects in human environments," *IEEE Access*, vol. 8, pp. 117900–117920, 2020.

[2] M. A. V. J. Muthugala and A. G. B. P. Jayasekara, "A review of service robots coping with uncertain information in natural language instructions," *IEEE Access*, vol. 6, pp. 12913–12928, 2018.

[3] L. Lestingi, M. Askarpour, M. M. Bersani, and M. Rossi, "A deployment framework for formally verified human-robot interactions," *IEEE Access*, vol. 9, pp. 136616–136635, 2021.

[4] M. A. K. Niloy, A. Shama, R. K. Chakrabortty, M. J. Ryan, F. R. Badal, Z. Tasneem, M. H. Ahamed, S. I. Moyeen, S. K. Das, M. F. Ali, M. R. Islam, and D. K. Saha, "Critical design and control issues of indoor autonomous mobile robots: A review," *IEEE Access*, vol. 9, pp. 35338–35370, 2021.

[5] A. Motroni, A. Buffi, and P. Nepa, "A survey on indoor vehicle localization through RFID technology," *IEEE Access*, vol. 9, pp. 17921–17942, 2021.

[6] Y. Zheng, S. Chen, and H. Cheng, "Real-time cloud visual simultaneous localization and mapping for indoor service robots," *IEEE Access*, vol. 8, pp. 16816–16829, 2020.

[7] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39830–39846, 2020.

[8] C. Chen, J. Jiang, N. Lv, and S. Li, "An intelligent path planning scheme of autonomous vehicles platoon using deep reinforcement learning on network edge," *IEEE Access*, vol. 8, pp. 99059–99069, 2020.

[9] F. Li, X. Fan, and Z. Hou, "A firefly algorithm with self-adaptive population size for global path planning of mobile robot," *IEEE Access*, vol. 8, pp. 168951–168964, 2020.

[10] M. Luo, X. Hou, and J. Yang, "Surface optimal path planning using an extended Dijkstra algorithm," *IEEE Access*, vol. 8, pp. 147827–147838, 2020.

[11] Z. Liu, H. Liu, Z. Lu, and Q. Zeng, "A dynamic fusion pathfinding algorithm using Delaunay triangulation and improved A-star for mobile robots," *IEEE Access*, vol. 9, pp. 20602–20621, 2021.

[12] Q. Jin, C. Tang, and W. Cai, "Research on dynamic path planning based on the fusion algorithm of improved ant colony optimization and dynamic window method," *IEEE Access*, early access, Mar. 9, 2021, doi: 10.1109/ACCESS.2021.3064831.

[13] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "DWA-RL: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 6057–6063.

[14] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[15] K. Cai, C. Wang, J. Cheng, C. W. De Silva, and M. Q. Meng, "Mobile robot path planning in dynamic environments: A survey," *Instrumentation*, vol. 6, no. 2, pp. 92–102, 2019.

[16] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[17] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1928–1935.

[18] T. Xu, S. Zhang, Z. Jiang, Z. Liu, and H. Cheng, "Collision avoidance of high-speed obstacles for mobile robots via maximum-speed aware velocity obstacle method," *IEEE Access*, vol. 8, pp. 138493–138507, 2020.

[19] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 696–706, Aug. 2011.

[20] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[21] M. Dobrevski and D. Skocaj, "Adaptive dynamic window approach for local navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 6930–6936.

[22] L. Liu, J. Yao, D. He, J. Chen, J. Huang, H. Xu, B. Wang, and J. Guo, "Global dynamic path planning fusion algorithm combining jump-A* algorithm and dynamic window approach," *IEEE Access*, vol. 9, pp. 19632–19638, 2021.

[23] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *Int. J. Robot. Res.*, vol. 26, no. 2, pp. 141–166, 2007.

[24] T. M. Howard, C. J. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *J. Field Robot.*, vol. 25, nos. 6–7, pp. 325–345, 2008.

[25] M. Kobayashi and N. Motoi, "Local path planning method based on virtual manipulators and dynamic window approach for a wheeled mobile robot," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2021, pp. 499–504.

[26] K. Yamazaki and M. Inaba, "Trajectory control of wheeled mobile robots based on virtual manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 2974–2978.

[27] H. Zhang, C. Zhang, W. Yang, and C.-Y. Chen, "Localization and navigation using QR code for mobile robot in indoor environment," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2015, pp. 2501–2506.

[28] J. Park, R. Delgado, and B. W. Choi, "Real-time characteristics of ROS 2.0 in multiagent robot systems: An empirical study," *IEEE Access*, vol. 8, pp. 154637–154651, 2020.

[29] H. Lee, H. Seo, and H. Kim, "Trajectory optimization and replanning framework for a micro air vehicle in cluttered environments," *IEEE Access*, vol. 8, pp. 135406–135415, 2020.

[30] D. Jin, Z. Fang, and J. Zeng, "A robust autonomous following method for mobile robots in dynamic environments," *IEEE Access*, vol. 8, pp. 150311–150325, 2020.

[31] *TurtleBot2*. Accessed: Feb. 5, 2022. [Online]. Available: https://www.turtlebot.com/turtlebot2/

[32] *URG-04LX-UG01*. Accessed: Feb. 5, 2022. [Online]. Available: https://www.hokuyo-aut.jp/search/single.php?serial=166

[33] *Obstacle_Detector*. [Online]. Available: https://github.com/tysik/obstacle_detector

MASATO KOBAYASHI (Graduate Student Member, IEEE) received the B.E. and M.E. degrees in marine engineering from Kobe University, Japan, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree with the Graduate School of Maritime Sciences. From 2019 to 2021, he was with the Technology Development Division, Seiko Epson Corporation, Japan. He is also working as a Research Intern at OMRON SINIC X Corporation, Japan. His current research interests include robotics, motion control, and haptic.

NAOKI MOTOI (Member, IEEE) received the B.E. degree in system design engineering and the M.E. and Ph.D. degrees in integrated design engineering from Keio University, Japan, in 2005, 2007, and 2010, respectively. In 2007, he joined the Partner Robot Division, Toyota Motor Corporation, Japan. From 2011 to 2013, he was a Research Associate at Yokohama National University, Japan. Since 2014, he has been with Kobe University, Japan, where he is currently an Associate Professor. From 2019 to 2020, he also held the position of a Visiting Professor at the Automation and Control Institute (ACIN), TU Wien, Austria. His current research interests include robotics, motion control, and haptic.