Model Predictive Obstacle Avoidance and Wheel Allocation Control of Mobile Robots Using Embedded CPU*

Naoki TAKAHASHI** and Kenichiro NONAKA**
** Department of Mechanical Systems Engineering, Tokyo City University
1–28–1 Tamazutsumi, Setagaya-ku, Tokyo 158–8557, Japan
E-mail: knonaka@tcu.ac.jp

Abstract
In this study, we propose a real-time model predictive control method for leg/wheel mobile robots which simultaneously achieves both obstacle avoidance and wheel allocation at a flexible position. The proposed method generates both obstacle avoidance path and dynamical wheel positions, and controls the heading angle depending on the slope of the predicted path so that the robot can keep a good balance between stability and mobility in narrow and complex spaces like indoor environments. Moreover, we reduce the computational effort of the algorithm by deleting usage of mathematical function in the repetitive numerical computation. Thus the proposed real-time optimization method can be applied to low speed on-board CPUs used in commercially-produced vehicles. We conducted experiments to verify efficacy and feasibility of the real-time implementation of the proposed method. We used a leg/wheel mobile robot which is equipped with two laser range finders to detect obstacles and an embedded CPU whose clock speed is only 80 MHz. Experiments indicate that the proposed method achieves improved obstacle avoidance comparing with the previous method in the sense that it generates an avoidance path with balanced allocation of right and left side wheels.

Key words: Mobile Robot, Embedded CPU, Model Predictive Control (MPC), Real-Time Optimization, Obstacle Avoidance Control

1. Introduction

Wheel driven robots have advantages in high energy efficiency, large movable load and strong stability. Especially, omni-directional wheeled robots, which can rotate and move in arbitrarily direction, are useful in narrow and complex spaces like indoor environment. In narrow spaces, mobile robots are required to reach desired position while they avoid obstacles on the way. Considering the necessity of keeping enough stability, however, mobile robots tend to require large footprint and it becomes difficult to achieve locomotion in narrow spaces. This is a trade off problem between stability and mobility. One solution of this challenge is to reconfigure its footprint depending on the surrounding environment. Mobile robots that can reconfigure a footprint to achieve high stability have been studied by many researchers(1),(2). Additionally, leg-wheel hybrid robots(3),(4) which are capable to walk on legs and drive on wheels and can reconfigure their footprint to a flexible position, have been developed.

Obstacle avoidance control methods for mobile robots have been studied widely. For example, the artificial potential field method(5), the global dynamics window approach(6), the trajectory generation algorithm using a Voronoi’s diagram and a Bézier curve(7) and nonlinear model predictive control method(8) have been reported. Additionally, the obstacle avoidance control method considering the robot’s shape has been proposed such as in Ref. (9) and Ref. (10). However, obstacle avoidance control methods, that involve reconfiguration of the footprint, have not been studied enough.
Model predictive control (MPC), also known as receding horizon control (RHC), is one of the real-time optimal control techniques used in a wide range of applications. MPC is a control method for constrained nonlinear systems which solves the on-line optimization problem over a finite time horizon. However, the computational effort tends to become large and implementation in embedded systems is difficult. Various techniques to reduce the computational effort of MPC are proposed\(^{(11)}\)–\(^{(13)}\). In Ref. \((11)\) and \((12)\), the custom RHC solver is proposed for the system with linear dynamics, convex objective functions and convex constraints. The computation time of this solver is several orders of magnitude faster than that of the conventional generalized optimization solver. In Ref. \((13)\), a real-time algorithm which combines the continuation method and GMRES method is proposed. This algorithm does not involve iterative optimization, and its computational time is almost constant for each control cycle.

In our past studies, we performed high speed model predictive obstacle avoidance control for an embedded CPU\(^{(14)}\) and model predictive wheel allocation control along a reference polynomial trajectory for leg/wheel robots\(^{(15)}\). Although obstacle avoidance and wheel allocation should be controlled by MPC simultaneously for total optimality, they had been considered separately so far. This may cause an unbalanced wheel allocation at the proximity of the obstacles. In this study, we propose a simultaneous obstacle avoidance and wheel allocation control method using real-time MPC to achieve a good balance between stability and mobility. Moreover, we reduce the computational effort by deleting usage of mathematical function logics in iterative computations. Thus, the proposed method can be applied to low speed on-board CPUs widely used in commercially-produced vehicles. Furthermore, we implement the proposed method with an embedded CPU and conduct experiments to verify feasibility and efficacy of the proposed method.

2. Planar leg/wheel mobile robot system

2.1. Description of the system

In this study, we focus on omni-directional leg-wheel mobile robots that can allocate wheels at a flexible position\(^{(15)}\). On a plane surface, the robot has two degrees of freedom of translational movement and one degree of freedom of rotational movement, and its legs have three degrees of freedom. Figure 1 depicts the system coordinates. The \(X − Y\) coordinate system denotes inertial coordinates fixed to the experimental field, and the \(x − y\) coordinate system stands for the relative coordinates fixed to the robot. Let \((X_r(t) \quad Y_r(t))\) be the robot position represented in \(X − Y\) coordinates, \(\theta_r(t)\) be the heading of the robot, and \((x_{ij}^w(t) \quad y_{ij}^w(t))\), for \(i = 0, 1\) and \(j = 0, 1\), be wheel positions represented in \(x − y\) coordinates, respectively. The superscripts \(ij\) indicate the corresponding leg; 00, 01, 10 and 11 denote front-left, front-right, rear-left and rear-right one, respectively. We assume that the robot can measure \((X_r(t) \quad Y_r(t))\), \(\theta_r(t)\) and the positions of vertices of the surrounding obstacles using external sensors. The control input to the robot is assumed to be translational and rotational speeds and desired wheel positions. Joint angles and wheel rotational speeds to assign wheels at desired positions and achieve desired velocities can be obtained using the optimization method considering
limitations of joint angles(16).

2.2. Coordinates and variables along the predicted path

Figure 2 depicts local coordinates and variables used in MPC throughout this paper. Let $s \in [0, s_l]$ be the travel distance along the predicted path starting from the robot position at time $t$ as $s = 0$ at the end of the predicted path $s = s_l$; we represent the robot’s and wheel’s positions in terms of $s$ in this study. It should be noted that $s$ is redefined at each sampling time $t$. The translational velocity of the robot at $t$ is defined as $V(t) := ds/dt$, which is assumed to be kept constant through $s \in [0, s_l]$. As will be shown in section 3.2, introducing the path length $s$ instead of $t$ enables the separation of both the wheel speed and its related constraints from the optimization of the other variables(15). Let $\psi(s)$ be the angle of movement direction with respect to the $X$–axis, where $\tan \psi(s)$ is the slope of $s$. The predicted robot position and heading angle are represented as $(X_i(s), Y_i(s))$ and the predicted wheel positions are represented as $(x_i^w(s), y_i^w(s)) (i, j = 0, 1)$ in $x(s) – y(s)$ coordinates. For simplicity, we assume that $x_i^w(s)$ is kept constant and $y_i^w(s)$ is a variable, which is adaptively changed according to the passage width. $\theta(s) = \theta_i(s) - \psi(s)$ is the angular difference between the direction of $V(t)$ and $x(s)$–axis.

Figure 3 depicts the definition of the passage width along the predicted path, where $O_i$ is the $i$-th obstacle whose vertices are numbered by $l = 1, 2, \cdots$ in a counterclockwise direction with coordinates $(X_{Oi,l}, Y_{Oi,l})$. In this study, the wheel positions along $y$–axis, $y_i^w(s)$ ($i, j = 0, 1$) are optimized with respect to the passage width. We define $D^f(s)$ as a distance from each wheel to the nearest wall of the obstacles in the direction normal to $V(t)$. Let $D_{max}$ be the maximum reachable length for each wheel. When the obstacle does not exist in the direction of the $y$–axis, we set $D^f(s) = D_{max}$. Then $D^f(s)$ is given by

$$D^f(s) = \begin{cases} 
\sqrt{\frac{y^f_1 - y^f_0}{\cos \psi(s)}} & \text{(if } 0 \leq \gamma \leq 1, 0 \leq \xi \leq 1) \\
D_{max} & \text{(else)}
\end{cases}$$

(1)
\[ Y^i_j = Y^i_1 + (-1)^j \cdot D_{\text{max}} \cdot \cos \psi(s). \]

Since \( D^{ij}(s) \) may vary discontinuously when an obstacle appears, we use the smoothed passage width \( D^{ij}_{\text{min}} \) which is computed through a low pass filter given as follows:

\[
T \dot{D}^{ij}_{\text{min}} + \dot{D}^{ij}_{\text{min}} = D^{ij}_{\text{min}} \tag{2}
\]

where \( T \) is the positive constant representing the time constant of low pass filter, and \( D^{ij}_{\text{min}} \) is the smallest passage width at each wheel position as follows:

\[
D^{ij}_{\text{min}} = \min_{s[01]} D^{ij}(s). \tag{3}
\]

2.3. Distance to obstacles

Figure 4 depicts the distance to obstacles: \( d_n(x_r(s)) \) is the shortest distance between the robot and the obstacle \( O_n \). \( d_n(x_r(s)) \) is switched between two cases when the nearest point of \( O_n \) is on the side (Case 1) or at the vertex (Case 2). In each case, \( d_n(x_r(s)) \) is expressed as follows:

\[
d_n(x_r(s)) = \frac{m(x_r(s))}{\sqrt{(X_{O_n,j+1} - X_{O_n,j})^2 + (Y_{O_n,j+1} - Y_{O_n,j})^2}} \tag{4}
\]

where

\[
m(x_r(s)) = (Y_{O_n,j+1} - Y_{O_n,j})(X_r(s) - X_{O_n,j}) - (X_{O_n,j+1} - X_{O_n,j})(Y_r(s) - Y_{O_n,j}). \tag{5}
\]

\( X_{O_n,j} \) and \( Y_{O_n,j} \) are coordinates of a vertex \( l \) of \( O_n \) represented in \( x - y \) coordinates.

3. Simultaneous obstacle avoidance and wheel allocation by MPC

In our past studies, we proposed a model predictive obstacle avoidance control for a point mass vehicle\(^{(14)}\) and a model predictive wheel allocation method for leg/wheel mobile robots\(^{(15)}\). Although obstacle avoidance and wheel allocation should be considered simultaneously, these had been considered separately. To improve total optimality, we propose a simultaneous model predictive obstacle avoidance and wheel allocation control. In this section, the dynamics of the robot, constraints for the control input, and the objective function of the proposed method are presented.

3.1. Dynamics of the mobile robot

Let the state vector \( x(s) \in \mathbb{R}^7 \), which represents vehicle position, heading angle, and \( y \)-axis position of each wheel, be defined by

\[
x(s) = [x_r(s)^T \ x_w(s)^T]^T \tag{6}
\]

where

\[
x_r(s) = [X_r(s) \ Y_r(s) \ \theta_r(s)]^T \tag{7}
\]

\[
x_w(s) = [y_{w0}(s) \ y_{w10}(s) \ y_{w11}(s)]^T. \tag{8}
\]

Let the corresponding velocity vector \( u(s) \in \mathbb{R}^7 \) with respect to \( s \) be defined by

\[
u(s) = [\tau(s)^T \ \mu(s)^T]^T \tag{9}
\]

where

\[
\tau(s) = [\tau_X(s) \ \tau_Y(s) \ \tau_\theta(s)]^T \tag{10}
\]

\[
\mu(s) = [\mu_{00}^0(s) \ \mu_{01}^0(s) \ \mu_{10}^0(s) \ \mu_{11}^0(s)]^T, \tag{11}
\]

where \( \tau_X(s), \tau_Y(s), \tau_\theta(s) \) and \( \mu_{ij}^0(s) \) are derivatives with respect to \( s \) of \( X_r(s), Y_r(s), \theta_r(s) \) and \( y_{w}^0(s) \), respectively. We assume that \( u(s) \) is the control input in this study.
The state space equation along with $s$ is represented as a single integrator system:

$$\frac{dx(s)}{ds} = u(s).$$  \hfill (12)

It is noted that the $[\tau_X(s), \tau_Y(s)]^T$ vector represents the unit tangent vector along the predictive path. The constraints related to the control input and obstacle avoidance are shown in the following.

### 3.2. Constraints for the control inputs along the predicted path

Since control vector $[\tau_X(s), \tau_Y(s)]^T$ is the unit tangent vector along the path, input $\tau_X(s)$ and $\tau_Y(s)$ should satisfy the following constraint:

$$C(\tau(s)) := \tau_X(s)^2 + \tau_Y(s)^2 - 1 = 0. \hfill (13)$$

Considering the physical limitation of $y$-axial wheel speeds, $\mu^{ij}(s)$ should satisfy the following constraint:

$$|\mu^{ij}(s)| \cdot V(t) \leq \mu_{\text{max}} \hfill (14)$$

where $\mu_{\text{max}}$ is the upper bound of the speed in $y$-direction determined by the motor performance. In order to satisfy the constraint Eq. (14), $V(t)$ is computed as follows:

$$V(t) = \min_{i,j} V^{ij}(t) \hfill (15)$$

with

$$V^{ij}(t) = \begin{cases} V_r & \text{if } \mu_{\text{max}} \geq |\mu^{ij}(s)| \cdot V_r \\ \frac{\mu_{\text{max}}}{|\mu^{ij}(s)|} & \text{if } \mu_{\text{max}} < |\mu^{ij}(s)| \cdot V_r \end{cases} \hfill (16)$$

where $V^{ij}(t)$ is the velocity of the robot which satisfies the constraint (14) for each wheel and $V_r$ is the reference velocity of the robot. Thus, $V(t) = ds/dt$ can be determined explicitly without any numerical optimization; this is the advantage to use $s$ instead of $t$ to reduce the computational effort. If $t$ is used, then the four inequality constraints related to the $y$-directional speed of each wheel, $\mu^{ij} V_r$, should be incorporated into the optimization, which obviously increases the computational efforts.

### 3.3. Continuous MPC for obstacle avoidance

In order to achieve both path tracking and obstacle avoidance, we use model predictive control involving obstacle avoidance based on the artificial potential field method. Suppose the objective function is given by

$$J = \phi(x(s_0)) + \int_{s_0}^{s_h} L(x(s), u(s)) ds \hfill (17)$$

with

$$\phi(x(s_0)) = \frac{1}{2} (x(s_0) - x_{\text{ref}}(s_0))^T S(x(s_0) - x_{\text{ref}}), \hfill (18)$$

$$L(x(s), u(s)) = \frac{1}{2} (x(s) - x_{\text{ref}}(s))^T Q(x(s) - x_{\text{ref}}(s))$$

$$+ \frac{1}{2} u(s)^T Ru(s) + w_1 p(x(s)) + w_2 q(\theta_r(s)) \hfill (19)$$

where $s_h \in \mathbb{R}$ is the length of the horizon, $x_{\text{ref}}(s) \in \mathbb{R}^7$ is a reference state vector, $S \in \mathbb{R}^{7 \times 7}$, $Q \in \mathbb{R}^{7 \times 7}$, and $R \in \mathbb{R}^{7 \times 7}$ are positive definite matrices, and $w_1, w_2 \in \mathbb{R}$ are positive scalar weights. It should be noted that $x(0)$ equals the current state. $p(x_r(s))$ is the sum:

$$p(x_r(s)) = \sum_{n=1}^{N_o} \frac{1}{d_n(x_r(s))^2} \hfill (20)$$

where $d_n(x_r(s))$ is the shortest distance between the robot and the obstacle, $n$ is the index number of the obstacle and $N_o$ is the total number of detected obstacles. Equation (20) represents
an artificial potential field\(^5\), \(p(x_r(s))\) is a positive, continuous and differentiable function except when \(d_w(x_r(s)) = 0\), which grows up to infinity as \(d_w(x_r(s)) \to 0\). \(q(\theta_r(s))\) is the sum of the y-axis passage width at each wheel position given as follows:

\[
q(\theta_r(s)) = \sum_{i} \left( \frac{\tilde{j}_{min}^i}{\cos \delta\theta(s)} \right)^2.
\] (21)

A reference state vector \(x_{ref}(s)\) is defined by

\[
x_{ref}(s) = [x_{ref}(s)^T \ x_{w,ref}(s)^T]^T
\] (22)

where

\[
x_{ref}(s) = [X_{ref}(s) \ Y_{ref}(s) \ \theta_{ref}(s)]^T
\] (23)

\[
x_{w,ref}(s) = [y_{w,ref}^0(s) \ y_{w,ref}^1(s) \ y_{w,ref}^{11}(s)]^T.
\] (24)

\(x_{ref}(s)\) is a given vector representing the desired path and \(x_{w,ref}(s)\) is a given reference wheel position computed by Eq. (49) shown later.

Then, the MPC optimization problem is represented as follows:

\[
\text{minimize} \quad J \quad \text{subject to} \quad \frac{dx(s)}{ds} = u(s) \quad \text{and} \quad C(\tau(s)) = 0.
\] (25)

In order to incorporate the constraint (13) into a recursive optimization framework, Eq. (19) is modified using the Lagrange multiplier method\(^{17}\) as follows:

\[
\hat{L}(x(s), u(s), \nu(s)) := L(x(s), u(s)) + \nu(s)C(\tau(s)) + w_CC(\tau(s))^2
\] (26)

where \(\nu(s) \in \mathbb{R}\) is the Lagrange multiplier and \(w_C \in \mathbb{R}\) is a scalar positive weight constant. Thus Hamiltonian is given by

\[
H(s) := H(x(s), u(s), \lambda(s), \nu(s)) = \hat{L}(x(s), u(s), \nu(s)) + \lambda^T(s)u(s)
\] (27)

where \(\lambda(s) \in \mathbb{R}^7\) is a costate vector. Using Eq. (27), Eq. (17) is rewritten as:

\[
\dot{J}(x(s), u(s), \lambda(s), \nu(s)) = \phi(x(s), s) + \int_0^s \left( H(s) - \lambda^T(s) \frac{dx(s)}{ds} \right) ds.
\] (28)

Thus, Eq. (25) can be modified as follows:

\[
\text{minimize} \quad J(x(s), u(s), \lambda(s), \nu(s)) = \phi(x(s), s) + \int_0^s \left( H(s) - \lambda^T(s) \frac{dx(s)}{ds} \right) ds \\
\text{subject to} \quad \frac{dx(s)}{ds} = u(s).
\] (29)

The Euler-Lagrange equations obtained from the stationary conditions of Eq. (28) derived from the variational principle are as follows:

\[
\frac{dx(s)}{ds} = u(s)
\] (30)

\[
x(0) = x_0
\] (31)

\[
\frac{d\lambda(s)}{ds} = - \left( \frac{\partial H(s)}{\partial x(s)} \right)^T
\] (32)

\[
\lambda(s_h) = \left( \frac{\partial \phi(x(s))}{\partial x(s)} \right)^T_{s=s_h}
\] (33)

\[
\frac{\partial H(s)}{\partial u(s)} = 0
\] (34)

where \(x_0 \in \mathbb{R}^7\) is the initial state vector.
3.4. Discrete time MPC for obstacle avoidance

In order to implement the proposed method, Eqs. (30) ~ (34) are discretized into discrete time Euler-Lagrange equations as follows:

\[
x(k + 1) = x(k) + u(k)\Delta s \tag{35}
\]
\[
x(0) = x_0 \tag{36}
\]
\[
\lambda(k) = \left(\frac{\partial H}{\partial x}(k)\right)^T \tag{37}
\]
\[
\lambda(N) = \left(\frac{\partial \phi}{\partial x}(k)\right)_{k=N} \tag{38}
\]
\[
\frac{\partial H}{\partial u}(k) = 0 \tag{39}
\]

where \(k \in [0, N]\) is the sampling sequence, \(N\) is the division number of the horizon along the path and \(\Delta s(= s_i/N)\) is the positive scalar representing the discrete interval travel length. Optimal inputs are computed by solving Eqs. (35) ~ (39) using the numerical optimization algorithms given later.

To conform the length of the reference path to the length of the horizon and divide the reference path equally, we define the discretized reference state of the robot \(x_{r,ref}(k)\) in the inertial coordinate system as follows:

\[
X_{r,ref}(k) = X_r(0) + k \cdot \cos \eta \cdot \Delta s \tag{40}
\]
\[
Y_{r,ref}(k) = \frac{Y_{ref} - Y_{r,0}}{X_{ref} - X_{r,0}}(X_{r,ref}(k) - X_{r,0}) + Y_{r,0} \tag{41}
\]
\[
\theta_{r,ref}(k) = \begin{cases} 
\theta_r(0) & \text{(if } k = 0) \\
\psi(k - 1) & \text{(if } k > 0) 
\end{cases} \tag{42}
\]

where

\[
\eta = \tan^{-1}\left(\frac{Y_{ref} - Y_{r,0}}{X_{ref} - X_{r,0}}\right) \tag{43}
\]
\[
\psi(k) = \tan^{-1}\left(\frac{\tau_y(k)}{\tau_x(k)}\right) \tag{44}
\]

\((X_{r,0}, Y_{r,0})\) and \((X_{ref}, Y_{ref})\) are the initial and the terminal position of the robot, respectively. Equations (40) ~ (42) show that the reference trajectory of the robot is a straight-line equation between \((X_{r,0}, Y_{r,0})\) and \((X_{ref}, Y_{ref})\), and \(\theta_{r,ref}(s)\) indicates the direction of the tangent vector along the predicted path. Equation (42) requires the robot to change the heading direction toward the slope of the predicted passage.

3.5. Reduction of computational effort

3.5.1. Transformation of state vector

Equation (44) is computed repeatedly in the optimization, which requires longer computational time due to the mathematical function of arc tangent. To reduce the computational effort of the arc tangent, we modify \(x_r(k), \tau_x(s)\) and \(x_{r,ref}(k)\) as \(x'_{r}(k), \tau'_{x}(s)\) and \(x'_{r,ref}(k)\) described by

\[
x'_{r}(k) = [X_r(k) \quad Y_r(k) \quad \tan(\theta_r(k))]^T \tag{45}
\]
\[
\tau'_{x}(s) = \frac{d \tan(\theta(s))}{ds} \tag{46}
\]
\[
x'_{r,ref}(k) = [X_{r,ref}(k) \quad Y_{r,ref}(k) \quad \tan(\theta_{r,ref}(k))]^T \tag{47}
\]

where

\[
\tan(\theta_{r,ref}(k)) = \begin{cases} 
\tan(\theta_r(0)) & \text{(if } k = 0) \\
\tan(\psi(k - 1)) = \frac{\tau_y(k - 1)}{\tau_x(k - 1)} & \text{(if } k > 0) 
\end{cases} \tag{48}
\]

It is noted that \(\tan(\theta_r(k))\) and \(\tan(\theta_{r,ref}(k))\) are computed only one time at only \(k = 0\). Since Eq. (45) restricts \(\theta_r(k)\) to \(-\pi/2 < \theta_r(k) < \pi/2\), however, \(\theta_{r,ref}(k)\) is also limited to \(-\pi/2 <
\( \theta_{\text{ref}}(k) < \pi/2 \) because \( \tau_{x}(k) \) is kept to be positive. In the rest of this paper, we replace \( x_{i}^{*}(k) \), \( \tau_{y}^{*}(s) \), and \( x_{\text{ref}}^{*}(k) \) by \( x_{i}(k) \), \( \tau_{y}(s) \), and \( x_{\text{ref}}(k) \), respectively.

3.5.2. Approximation of reference wheel position

The reference \( y \)-axis wheel position, needed to avoid obstacles \( y_{w, \text{ref}}^{j} \), is defined as follows:

\[
y_{w, \text{ref}}^{ij} = (-1)^j \left( \frac{\tilde{D}_{ij}^{\text{min}}}{\cos \delta \theta(s)} + \rho \right)
\]  

(49)

where \( \rho (> 0) \) is the safety margin between each wheel and the passage width. If \( y_{w, \text{ref}}^{ij} \) exceeds the range of movement, \( y_{w, \text{min}} \leq \left| y_{w}^{ij}(s) \right| \leq y_{w, \text{max}} \cdot y_{w, \text{ref}}^{ij} \) takes the limit \( (-1)^j \cdot y_{w, \text{min}} \) or \( (-1)^j \cdot y_{w, \text{max}} \). When \( y_{w, \text{ref}}^{ij} \) varies discontinuously, the optimal path of each wheel also varies discontinuously, which requires longer computational time to reach the optimal path through model predictive computation because the initial point of optimization tends to be far from the optimal one. Using the smoothed passage width \( \tilde{D}_{ij} \) of Eq. (2) to compute \( y_{w, \text{ref}}^{ij} \), we avoid a discontinuous \( y_{w, \text{ref}}^{ij} \). At every optimization stage, Eq. (49) and its derivative, which include the computation of \( \sin \delta \theta(s) \) and \( \cos \delta \theta(s) \), are updated. Since the trigonometric function requires longer computational time, we approximate \( \sin \delta \theta(s) \) and \( \cos \delta \theta(s) \) by

\[
\sin \delta \theta(s) \approx \delta \theta(s) - \frac{1}{3!} \delta \theta(s)^{3}
\]  

(50)

\[
\cos \delta \theta(s) \approx 1 - \frac{1}{2} \delta \theta(s)^{2}
\]  

(51)

where

\[
\delta \theta(s) \approx \begin{cases} 
\tan \delta \theta(s) - \frac{1}{3} \tan \delta \theta(s)^{3} & \text{(if } -1 \leq \tan \delta \theta(s) \leq 1) \\
\frac{2}{3} & \text{(if } \tan \delta \theta(s) > 1) \\
-\frac{2}{3} & \text{(if } \tan \delta \theta(s) < -1)
\end{cases}
\]

(52)

\[
\tan \delta \theta(s) = \frac{\tan \theta - \tan \theta_{\text{ref}}}{1 + \tan \theta \tan \theta_{\text{ref}}}
\]

(53)

Equations (50) and (51) are the 3rd degree MacLaurin polynomials of \( \sin \delta \theta(s) \) and \( \cos \delta \theta(s) \) for the argument \( \delta \theta(s) \) respectively, and Eq. (52) is the 3rd degree MacLaurin polynomial of \( \tan^{-1}(\tan \delta \theta(s)) \) for the argument \( \tan \delta \theta(s) \) which can be quickly computed from \( \tan \theta_{r} \) in Eq. (45) and \( \tan \theta_{\text{ref}} \) in Eq. (48). We saturate Eq. (52) at \( \tan \delta \theta(s) = 1 \) and \( \tan \delta \theta(s) = -1 \) to avoid monotonic decreasing or increasing \( \delta \theta(s) \) at \( |\tan \delta \theta(s)| > 1 \) due to the approximation to the cubic polynomial.

Since Eqs. (28) and (35)–(39) are represented by four arithmetic operations, the optimization can be computed rapidly even on low speed embedded CPUs which are not equipped with logics of mathematical functions. The algorithm of the optimization is as follows:

1. Initialization of \( x(0) = x_0, u(0) = u_0, v(k) = v_0 \).
2. Computation of \( x(k) \) from \( k = 1 \) to \( k = N \) by Eq. (35).
3. Computation of \( x_{\text{ref}}(k) \) from \( k = 0 \) to \( k = N \) by Eqs. (40)–(42) and (49).
4. Computation of \( \lambda(N) \) by Eq. (38).
5. Computation of \( \lambda(k) \) from \( k = N - 1 \) to \( k = 0 \) by Eq. (37).
6. Computation of \( \partial H/\partial u(k) \) from \( k = 0 \) to \( k = N - 1 \) to differentiate partially Eq. (27) with respect to \( u(k) \).
7. Computation of \( \alpha \) using golden section method.
8. \( \Delta u(k) = \alpha \cdot \partial H/\partial u(k) \).
9. \( u(k) \leftarrow u(k) + \Delta u(k) \).
10. If \( ||\Delta u(k)|| > \epsilon_{\text{max}} \), then go to (2).
11. If \( ||C(\tau(s))|| < \epsilon_c \), then quit the optimization and set \( u(0) \) to the inputs.
12. Update \( v(k) \leftarrow v(k) + 2rC(\tau(k)) \).
13. Go to (2).
$u_0 \in \mathbb{R}^7$ is the initial input vector, $\Delta u(k) \in \mathbb{R}^7$ is a variation vector for $u(k)$, $v_0 \in \mathbb{R}^7$ is the initial Lagrange multiplier vector, $\alpha \in \mathbb{R}^*$ is the step range, $\epsilon_{du} \in \mathbb{R}^*$ is the threshold of the unconstrained optimization and $\epsilon_{e} \in \mathbb{R}^*$ is the threshold of the constraint, respectively.

The optimal robot velocity vector $v(t) \in \mathbb{R}^3$ can be computed by

$$v(t) = [v_x(t) \ v_y(t) \ v_\theta(t)]^T$$

where

$$v_x(t) := \left( \frac{dX_r(s)}{ds} \right)_{s=0} \frac{ds}{dt} = \tau_x(0) \cdot V(t)$$

$$v_y(t) := \left( \frac{dY_r(s)}{ds} \right)_{s=0} \frac{ds}{dt} = \tau_y(0) \cdot V(t)$$

$$v_\theta(t) := \left( \frac{dB_r(s)}{ds} \right)_{s=0} \frac{ds}{dt} = (1 + \tan^2(\theta_r(0)))^{-1} \cdot \left( \frac{d\tan(\theta_r(s))}{ds} \right)_{s=0} \frac{ds}{dt} = (1 + \tan^2(\theta_r(0)))^{-1} \cdot \tau_\theta(0) \cdot V(t).$$

3.6. Switching to the optimal regulator in the proximity of the terminal position

Although the usual model predictive control takes $t$ to be the variable for integral interval, the proposed method uses the travel distance $s$ for the variable, which is restricted to $s \in [0, s_h]$ in MPC above, where the $V$ is preferred to be kept to the desired velocity $V_r$. When the travel distance from the current robot position to the terminal position is shorter than $s_h$, in order to achieve the asymptotic convergence to the terminal position, the controller is switched to the optimal regulator to compute $v(t)$ and $y^{(i)}(t)$.

$$\frac{dx_{\text{reg}}}{dt} = Ax_{\text{reg}} - B \cdot K \cdot (x_{\text{reg}} - x_{\text{reg},d})$$

where

$$A = \begin{bmatrix} O_{3 \times 3} & I_3 & O_{3 \times 4} \\ O_{7 \times 3} & O_{7 \times 3} & O_{7 \times 7} \end{bmatrix}, \quad B = \begin{bmatrix} O_{3 \times 7} \\ I_7 \end{bmatrix}, \quad K = \begin{bmatrix} K_r & K_c & O_{3 \times 4} \\ K_r & O_{4 \times 3} & K_c \end{bmatrix},$$

$$x_{\text{reg}} = [X_r(t), Y_r(t), \theta_r(t), v_x(t), v_y(t), v_\theta(t), y_r^{(0)}(t), y_r^{(1)}(t), y_r^{(2)}(t), y_r^{(3)}(t)]^T,$$

$$x_{\text{reg},d} = [X_{rd}, Y_{rd}, \theta_{rd}, 0, 0, 0, y_{u,\text{max}} - y_{u,\text{max}}, y_{u,\text{max}} - y_{u,\text{max}}, y_{u,\text{max}} - y_{u,\text{max}}]^T.$$

$K_r \in \mathbb{R}^{3 \times 3}$, $K_c \in \mathbb{R}^{3 \times 3}$ and $K_w \in \mathbb{R}^{4 \times 4}$ are feedback coefficient matrices, $x_{\text{reg}}$ and $x_{\text{reg},d}$ are the state and the target state of the regulator, and $\theta_{rd}$ is the terminal heading angle, respectively.

4. Experiment

We apply the proposed method to the actual robot system and conduct experiments to verify the feasibility of the real-time optimization using the embedded CPU. To show the advantage over the previous method in terms of mobility and stability, the experimental result based on the algorithm proposed in Ref. (15) is shown for comparison.

4.1. Experimental robot

In this study, we use the leg/wheel robot depicted in Fig. 5, whose specification is shown in Table 1. Figures 6 and 7 depict a top view and a side view of a leg, respectively. Each leg has a set of actuators comprised of three servomotors for joints with 3DOF SCARA motion and a DC motor for a driving wheel equipped with a rotary encoder which realizes the desired angular speed; this robot can move to an arbitrary direction with arbitrary attitude while it can allocate wheels at flexible positions on a flat surface.

This robot is equipped with an embedded CPU for vehicle control, whose specifications are summarized in Table 2. This CPU is widely used for automotive engine control; it has very high reliability and robustness so that it can operate under severe conditions. Since its clock speed is only 80 MHz and its available memory size is limited to 1 MB only, methods which require high computational effort and/or large lookup tables cannot be implemented. In
Fig. 5 Experimental robot.

Table 1 Specifications of the experimental robot.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tread</td>
<td>Maximum 0.2 m</td>
</tr>
<tr>
<td></td>
<td>Minimum 0.7 m</td>
</tr>
<tr>
<td>Wheelbase</td>
<td>Maximum 0.2 m</td>
</tr>
<tr>
<td></td>
<td>Minimum 0.7 m</td>
</tr>
<tr>
<td>Height</td>
<td>0.24 m</td>
</tr>
<tr>
<td>Weight</td>
<td>7.77 kg</td>
</tr>
</tbody>
</table>

order to demonstrate the low computational cost and the practicality of the proposed method, this embedded CPU is used in this study.

As external sensors, this robot has a LRF sensor system consisting of a notebook computer and two laser range finders (LRFs) whose specifications are summarized in Table 3. The robot can scan around 360-degree environment using two LRFs. This sensor system measures the distance to the surrounding environment, estimates the self-location and detects obstacles. Localization is executed based on information for walls, which surround the experimental field, obtained from the LRFs. It should be noted that the notebook computer is used only for the localization and obstacle detection, and not used for vehicle control.

The system configuration of this robot is depicted in Fig. 8. The LRF sensor system estimates the self location from the known map data and measures obstacle positions; this information is sent to the embedded CPU with a period of 100 ms. Obstacles found in the detection area are approximated by polygon shapes and expressed as coordinate data of vertices. The embedded CPU computes wheel speeds and joint angles at every 18 ms, i.e. with 55.6 Hz from the robot velocity and wheel positions through the Quasi-Newton optimization method of the BFGS algorithm.

4.2. Experimental condition

Experiments are conducted with a flat rectangle field with polygonal obstacles. Since the computational effort to calculate the distance to an obstacle is large, the computational time tends to be longer for environments which include many obstacles. Moreover, the effect of the potential field Eq. (20) is small when the obstacle stands away from the robot. Thus, we limit the obstacle detection area using the LRF sensor system to the shadowed area (2 \( \times \) 1.9 m) depicted in Fig. 9. The accuracy of obstacle detection in the experiment is 20 mm.

To conduct a fair comparison, the weights for the objective function of both the proposed method and Ref. (15) are tuned through cut and try experiments of a simple obstacle avoidance problem depicted in Fig. 10 (case I), so that the right side wheels are pulled up to \( y_j = -0.2 \text{ m} \) \((j = 0, 1)\). Then, we conduct a different experiment for a complicated environment depicted in Fig. 11 (case II), which is very similar to the one tested in Ref. (15). We assign four square obstacles \( O_1 \) through \( O_4 \) with 0.3 m on a side so that the optimal path includes both bottleneck
and double lane change which often appear in cluttered environments. We also demonstrate that the computational speed of the proposed method is high enough. In both cases, the initial position is \((X_r,0,Y_r,0) = (0.5,2.5)\) and the terminal position is \((X_r,d,Y_r,d) = (6.5,2.5)\) and the controller is switched to the optimal regulator for \(X \geq 5.5\) m.

### 4.3. Indices for mobility and stability

The experimental results of case II are quantitatively evaluated for the indices defined below which represent mobility and stability performance, respectively.

To obtain high mobility in the environment with obstacles, it is preferred to shrink wheel position to its minimum, while keeping enough safety distance from the obstacles. Thus we consider \(y\)-directional distance between each wheel position and the nearest obstacle given by the absolute value of \(D_{ij}^{\text{margin}}\), represented by Eq. (59), for the mobility index:

\[
D_{ij}^{\text{margin}} = D_{ij}(0) - y_{ij}^{\text{w}}(0). \tag{59}
\]

In addition, to ensure the safety distance, we also keep

\[
\min_{ij} |D_{ij}^{\text{margin}}|, \tag{60}
\]

to be larger than the safety margin \(\rho\) appeared in Eq. (49).

To evaluate the stability, it is natural to use the area of the support polygon, \(a_s\). In addition, to keep the balanced allocation of right and left wheels, the weight of chassis should be uniformly distributed to each wheel. Thus we consider the weight distribution ratio, \(r_{ij}\), defined in Eq. (A.6) in the appendix, and use the minimum weight distribution ratio given by

\[
\min_{ij} r_{ij}, \tag{61}
\]

for another index of stability. It is noted that the ideal value of \(r_{ij}\) is 0.25. Since the height of the gravity center of this robot is low (90 mm), load shift due to the acceleration was ignored.

### 4.4. Experimental result

#### 4.4.1. Case I

Figures 12 through 17 and Figs. 24 through 25 depict the experimental results of the proposed method with parameters shown in Table 4. Figures 18 through 23 and Figs. 30 through 31 depict those of the method in Ref. (15) with parameters shown in Table 5.
To compare these results in terms of the position, the X-position of the mobile robot is used as the horizontal axis. Figures 12 and 18 depict the trajectories of gravity center and four wheels, where the gravity center, the two front wheels, and the two rear wheels are colored by red, green, and blue, respectively. The vertices and crossing diagonal lines of orange colored quadrangle indicates those of the position of the gravity center and positions. Figures 13 and 19 depict the y-directional distance of the wheels. Figures 12 and 18 indicate that both the proposed method and the method of Ref. (15) realized the obstacle avoidance by pulling the wheel position up to −0.2 m. Figures 14 through 17, 20 through 23, and 30, indicate the heading angle \( \theta \), input vector \( r \), translational and rotational velocity \( v_x, v_y, v_\theta \) and the variation of wheel position \( \mu \) of the proposed method, and the heading angle \( \theta \), parameters of the collision free path \( k_1, k_2, k_3, k_4 \), translational and rotational velocity \( v_x, v_y, v_\theta \) and the variation of wheel position \( \mu \) of the method in Ref. (15). Figures 25 and 31 depict the \( y \)-directional distance between the wheel and the nearest obstacle \( D_{\text{avg}} \) represented by Eq. (59) which indicates that the distance was larger than \( \rho = 0.2 \) m, and the two methods took similar safety distance between the wheel and the obstacle. Thus, the weight parameters for the proposed method resulted in similar obstacle avoidance distance with that of Ref. (15); we used the same weight parameters in the following experiment under a complex environment.

4.4.2. Case II Using weight parameters for case I, we conduct experiments in the environment depicted in Fig. 11. Figures 26 through 29 and Fig. 36 through 40 depict the experimental results of the proposed method, and Fig. 32 through 35 and Fig. 42 through 46 depict those of the method in Ref. (15). Figure 26, 27, 32 and 33 depict the trajectories of gravity center and four wheels, and \( y \)-directional wheel position. The proposed method achieved the
Table 6 Minimum absolute passage margin at each wheel position.

| Obstacle | Method | \( |D_{\text{margin}}^{ij}| \) [m] |
|----------|--------|------------------|
|          |        | 00   | 01   | 10   | 11   | Minimum |
| \( O_1 \) | Proposed | 0.208 | 0.154 | 0.207 | 0.200 | 0.208 |
|          | Previous | 0.167 | 0.172 | 0.207 | 0.154 | 0.172 |

Table 7 Minimum weight distribution ratio.

<table>
<thead>
<tr>
<th>Method</th>
<th>( \rho_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00</td>
</tr>
<tr>
<td>0.5 ( \leq X \leq 2.5 )</td>
<td>Proposed</td>
</tr>
<tr>
<td></td>
<td>Previous</td>
</tr>
<tr>
<td>2.5 ( \leq X \leq 4.0 )</td>
<td>Proposed</td>
</tr>
<tr>
<td></td>
<td>Previous</td>
</tr>
<tr>
<td>4.0 ( \leq X \leq 6.5 )</td>
<td>Proposed</td>
</tr>
<tr>
<td></td>
<td>Previous</td>
</tr>
</tbody>
</table>

obstacle avoidance control with obviously larger deviation from the desired path and smaller variation of wheel positions than those of the method in Ref. (15). Figs. 36, 37 and 38 indicate that the velocity of the proposed method varied continuously by the usage of the passage width smoothed by LPF in Eq. (2), although it varied discontinuously in the method of Ref. (15) as depicted in Figs. 42, 43 and 44.

Figures 39 and 45 represent \( D_{\text{margin}}^{ij} \), the margin of the passage width between each wheel and the obstacle. The double-headed arrows at \( Y = 0.5, 0.55, -0.5, -0.55 \) indicate the ranges where the obstacle exists in a direction toward \( y \)-axis. Additionally, as the typical index for the mobility, the minimum of \( \rho_j \) for each obstacle is summarized in Table 6 at the corresponding point indicated in Figs. 39 and 45 by black arrows. The blanks of Table 6 indicate that the obstacle does not exist in a direction toward \( y \)-axis. Figures 40 and 46 represent the weight distribution ratio \( \rho_j \) for each method. The minimum of weight distribution ratio and the index of the stability (61) in three range \( 0.5 \leq X \leq 2.5, 2.5 \leq X \leq 4.0 \) and \( 4.0 \leq X \leq 6.5 \) is summarized in Tables 7 at the corresponding point indicated in Figs. 40 and 46 by black arrows. Figure 41 depicts the area of the support polygon, \( A_s \), and its numerical integral with respect to \( X, A_s \), for both the proposed method and Ref. (15).

In the bottleneck \( (O_1 \) and \( O_2 \)), \( A_s \) of the proposed method was smaller than that of Ref. (15). But \( |D_{\text{margin}}^{ij}| \) of Ref. (15) was partially smaller than the safety margin \( \rho = 0.2 \) m; Ref. (15) achieved a larger \( A_s \) at the cost of safety. On the other hand, the proposed method always satisfied the safety margin, because \( |D_{\text{margin}}^{ij}| \) was kept larger than \( \rho \). In addition, the minimum of \( \rho_j \) was larger than that of Ref. (15); it achieved the balanced allocation of right and left wheels to pass through the bottleneck.

During the double lane change \( (O_3 \) and \( O_4) \), the proposed method achieved improved stability and mobility, because it resulted in larger \( A_s \) and \( \min |\rho_j| \) comparing with Ref. (15), while \( |D_{\text{margin}}^{ij}| \) was almost larger than \( \rho \). The minimum of \( D_{\text{margin}}^{ij} \) was 0.199 m which was slightly smaller than \( \rho = 0.2 \) m, but it was within the range of the distance measurement error of approximately 20 mm.

Throughout the path, the mobility of the proposed method was similar to Ref. (15), while the second stability index (61) was better. Only the area of the support polygon, \( A_s \), was inferior at the bottleneck in order to satisfy safety margin, but the integral of the support polygon, \( A_s \), in Fig. 41 indicates that the overall performance was similar, while the proposed method always kept the safety margin which the method in Ref. (15) did not. Thus, it is concluded that the balanced stability and mobility was achieved in the proposed method.
Fig. 12 Experimental trajectory of proposed method (case I).

Fig. 13 Experimental $y_{ij}(0)$ of proposed method (case I).

Fig. 14 Experimental $\theta$ of proposed method (case I).

Fig. 15 Experimental tangent vector of path of proposed method (case I).

Fig. 16 Experimental velocity of proposed method (case I).

Fig. 17 Experimental $\mu_{ij}(0)$ of proposed method (case I).

Fig. 18 Experimental trajectory of previous method (case I).

Fig. 19 Experimental $y_{ij}(0)$ of previous method (case I).

Fig. 20 Experimental $\theta$ of previous method (case I).

Fig. 21 Experimental parameters of previous method (case I).

Fig. 22 Experimental velocity of previous method (case I).

Fig. 23 Experimental $\mu_{ij}(0)$ of previous method (case I).
Fig. 24 Experimental $\mu_j^{(0)}$ of proposed method (case I).

Fig. 25 Experimental $D_{\text{margin}}^{j}$ of proposed method (case I).

Fig. 26 Experimental trajectory of proposed method (case II).

Fig. 27 Experimental $y_{ij}^{(0)}$ of proposed method (case II).

Fig. 28 Experimental $\theta$ of proposed method (case II).

Fig. 29 Experimental tangent vector of path of proposed method (case II).

Fig. 30 Experimental $\mu_j^{(0)}$ of previous method (case I).

Fig. 31 Experimental $D_{\text{margin}}^{j}$ of previous method (case I).

Fig. 32 Experimental trajectory of previous method (case II).

Fig. 33 Experimental $y_{ij}^{(0)}$ of previous method (case II).

Fig. 34 Experimental $\theta$ of previous method (case II).

Fig. 35 Experimental parameters of previous method (case II).
Fig. 36 Experimental velocity of proposed method (case II).

Fig. 37 Experimental $\mu^0_j(0)$ of proposed method (case II).

Fig. 38 Experimental $\mu^1_j(0)$ of proposed method (case II).

Fig. 39 Experimental $D_{ij}^{\text{main}}$ of proposed method (case II).

Fig. 40 Experimental $r_{ij}^{\text{main}}$ of proposed method (case II).

Fig. 41 Experimental $\alpha_j$ and $A_j$ (case II).

Fig. 42 Experimental velocity of previous method (case II).

Fig. 43 Experimental $\mu^0_j(0)$ of previous method (case II).

Fig. 44 Experimental $\mu^1_j(0)$ of previous method (case II).

Fig. 45 Experimental $D_{ij}^{\text{main}}$ of previous method (case II).

Fig. 46 Experimental $r_{ij}^{\text{main}}$ of previous method (case II).

Fig. 47 Experimental optimization clock of proposed method (case II).
Figure 47 depicts the computational clock of the proposed method. After the controller is switched to the optimal regulator for $X > 5.5$ m, the optimization is skipped and its clock becomes zero. When the real-time optimization was running for $X = 0.5$ m through $5.5$ m, the slowest clock was about 150 Hz around $X = 2.0$ m. Thus, the optimization clock is sufficiently faster than the period of the control cycle of the actuators ($18 \text{ms} = 55.6$ Hz), and the real-time implementation of the proposed model predictive obstacle avoidance method is achieved.

5. Conclusion

In this study, we proposed a simultaneous obstacle avoidance and wheel allocation control method for leg/wheel mobile robots. Although model predictive obstacle avoidance control and model predictive wheel allocation control have been studied separately in our past research\(^{(14),(15)}\), the proposed method integrates model-predictive optimal path planning with optimal allocation of wheels and controls of the heading angle depending on the slope of the predicted trajectory. Moreover the computational effort has been reduced by deleting repetitive usage of mathematical function logics to treat the tangent of the heading angle as the state variable and approximating the trigonometric functions using 3rd degree MacLaurin polynomials.

Experiments have been conducted to verify both the feasibility of the real-time implementation using the embedded CPU and the advantage over the previous result\(^{(15)}\). To show the performance in the practical situation, the experiments were conducted in the environment including both bottleneck and double lane change, which was the one of the common scenarios in obstacle avoidance. The proposed method achieved the real-time implementation using an embedded CPU. Furthermore, we compared the proposed method with the method of Ref.\(^{(15)}\) using the distance between each wheel and the nearest obstacle, and the weight distribution ratio as the indices for mobility and stability. Experimental results indicate that the balanced mobility and stability was achieved in the proposed method.

Like many of other MPCs, measurement error strongly affects the performance of the obstacle avoidance control. For the experimental results in this paper, the target running speed of the robot is only about 0.5 m/s which might give the robot a sufficient number of measurements of obstacle position and localization. Robustness against measurement error will be further investigated in our future research. In addition, with the proposed method, the robot may stuck in a local minimum which is generated by the potential field Eq.\((20)\) and the wheels may collide with the obstacle. Therefore, we must assign the obstacle so that the potential field does not generate local minimum points, and adjust the weights of the objective function so that wheels do not collide. Moreover, the optimization of the joint angles is considered separately from the proposed method. Thus, the future task is to resolve these issues.

Acknowledgment

The authors gratefully acknowledge the support of Grant-in-Aid for Scientific Research (C) No. 21560475. In this research, we gratefully acknowledge and thank Prof. Katsumasa Suzuki and Prof. Yoshikazu Kanamiya of Tokyo City University for their advices and Mr. Isao Okawa for his support on the experimental system. The authors are also deeply grateful to the valuable comments from the reviewers.

Appendix

Let us assume that the robot body is approximated by a rigid flat plate which is supported by four springs representing elasticity of tire located at wheel position as depicted in Fig. 48. $\delta_{\text{pitch}}$ is the pitch angle and $\delta_{\text{roll}}$ is the roll angle, respectively. $m$ is the mass, $h$ is the height of the gravity center, $(x_{ijw}, y_{ijw}, z_{ijw}) (i = 0, 1, j = 0, 1)$ is the wheel positions, $k$ is the spring constant, $z_0$ is the natural length of the spring, $F_{ij} (i = 0, 1, j = 0, 1)$ is the elastic restoring force from the spring, and $g$ is the acceleration of gravity, respectively. When both $\delta_{\text{pitch}}$ and
δ_{roll} are small, the following equations are obtained using geometric relations, Hooke’s law of elasticity, and the static equations of equilibrium on vertical force and moments:

\[ z_{ij} = x_{ij}^{w} \delta_{pitch} + y_{ij}^{w} \delta_{roll} + h \]  
(A.1)

\[ F^{ij} = -\kappa (z_{ij}^{w} - z_{0}) \]  
(A.2)

\[ \sum_{ij} F^{ij} = mg \]  
(A.3)

\[ \sum_{ij} F^{ij} x_{ij}^{w} = 0 \]  
(A.4)

\[ \sum_{ij} F^{ij} y_{ij}^{w} = 0 \]  
(A.5)

The weight distribution ratio \( r^{ij} \) is given as follows:

\[ r^{ij} = \frac{F^{ij}}{mg} \]  
(A.6)

Eliminating \( \delta_{pitch} \), \( \delta_{roll} \) and \( h - z_{0} \) from Eqs (A.1) through (A.5), and substituting \( F^{ij} \) into Eq. (A.6), \( r^{ij} \) is represented by the following equation which is independent of \( \kappa \), \( m \), and \( g \):

\[ r^{ij} = \left[ x_{ij}^{w} \quad y_{ij}^{w} \right] \begin{bmatrix} \sum_{ij} (x_{ij}^{w})^2 & \sum_{ij} x_{ij}^{w} y_{ij}^{w} & \sum_{ij} y_{ij}^{w} \\ \sum_{ij} x_{ij}^{w} y_{ij}^{w} & \sum_{ij} (y_{ij}^{w})^2 & \sum_{ij} y_{ij}^{w} \\ \sum_{ij} x_{ij}^{w} & \sum_{ij} y_{ij}^{w} & 4 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \]  
(A.7)

References


