IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 December 2024, accepted 8 January 2025, date of publication 17 January 2025, date of current version 23 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3530967

RESEARCH ARTICLE

Robot Simulation on Agri-Field Point Cloud With Centimeter Resolution

SHINTARO NODA, MASAYUKI KOGOSHI, AND WATARU IIJIMA
Research Center for Agricultural Robotics, National Agriculture and Food Research Organization (NARO), Tsukuba 305-0856, Japan

Corresponding author: Shintaro Noda (nodas152@naro.affrc.go.jp)

This work was supported in part by the Public-Private Research and Development Investment Strategic Expansion Program (PRISM) of the Cabinet Office, Government of Japan.

ABSTRACT The need for robotic agricultural automation has been driven by global population growth and climate change. To efficiently evaluate and develop agricultural robots not limited to the growing season, we developed a dynamics simulator that works fast on 3D point-cloud models of agricultural fields. The point-cloud models have been widely used in recent agricultural research thanks to advances in aerial photography technology. Therefore, the simulator can be easily applied to many agricultural fields. To speed up the dynamics calculation on the dense point-cloud model, we developed a method to quickly detect collision points using a grid table, and a method to calculate collision forces between the points and robot meshes. The performance of the simulator was evaluated on an agri-field model (31×14 m^2) represented by 1.7×10^6 points. The computation time of the simulation was 8.8 times faster than real time, and the simulation accuracy compared to actual robot movements was $\sim$ 1 cm in Root Mean Square Error (RMSE). The simulator in this study enables fast computation and accurate prediction of robot movements on centimeter-resolution agri-field point-cloud models, supporting research on agricultural robots not limited to the growing season.

INDEX TERMS Agribot, dynamics simulation, agricultural field, agricultural robot, point cloud.

I. INTRODUCTION

A. BACKGROUND

Global population growth and climate change causing food supply shortages [1] require efficient, data-driven precision agriculture [2]. Especially in Japan, the low food self-sufficiency rate and labor shortage due to the aging of farmers are serious problems [3]. Hence, automated agriculture with robotics is urgently needed.

One reason making agricultural automation difficult is that the growing season is limited. The growing season occurs only a few times per year. Therefore, it takes many years to develop, evaluate, and improve robots repeatedly.

Simulation is an effective tool to study agricultural robots not limited to the growing season. In the field of agricultural research, previous works on simulation include crop growth modeling [4] and robot simulation tools [5], [6], [7].

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

B. PROBLEM

A problem with the existing simulation tools for agricultural robots is the difficulty in directly handling point-cloud field models. In recent agricultural research, the use of the 3D point-cloud data has become common thanks to the popularization of aerial photography technology. However, the existing dynamics engines plugged into the simulation tools require mesh models.

A dedicated dynamics engine is required to efficiently handle the point-cloud data due to the large number of points as well as the different model formats. The field model used in this work contained 1.7×10^6 points. The field size was 31×14 m^2 with less than 2×2 cm^2 resolution.

The 3D point-cloud models can be easily generated by using commercial software [8] implementing Structure-from-Motion (SfM) technology [9]. The technology is based on the principle of triangulation using 2D aerial images. Thus, the robot simulator specialized for the 3D point-cloud model can be easily applied to many agricultural fields.

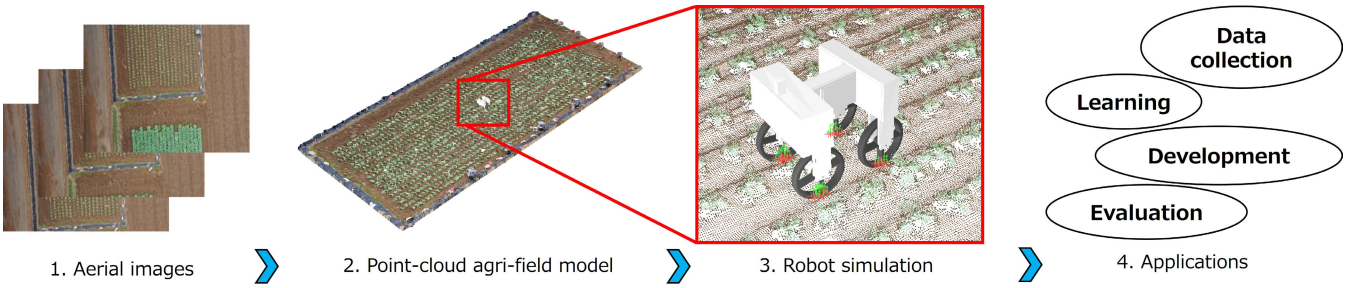

FIGURE 1. High-Speed simulation technology and applications on 3D point-cloud models of agricultural fields generated from aerial images.

C. APPROACH

To efficiently perform simulations on the dense point-cloud data, we developed the following two approaches:

1. Collision point retrieval near around the robot is accelerated by tabling all points in a grid form.
2. Contact force is calculated by integrating the forces between each point and the robot mesh without converting the point-cloud data to mesh data.

To evaluate the simulator, we conducted the following three studies:

1. Evaluation of computational time and memory usage required to compute a unit simulation time.
2. Evaluation of computational stability with respect to the change of the simulation time step.
3. Evaluation of prediction error by comparing actual and simulated robot positions.

Fig. 1 shows a conceptual diagram of this study. The first step is to generate a 3D point-cloud model from aerial images of the field using the SfM technology. Our simulator is designed to work quickly on the 3D model. The applications include data collection, learning, robot development and performance evaluation regardless of the growing season.

The motivation of this research is to establish a fundamental technology to accelerate research on agricultural robots by developing a simulator that operates at high speed on detailed point-cloud models of arbitrary agricultural fields.

D. RELATED WORKS

To automate agriculture, robotization of agricultural machinery (e.g., tractors) has been widely studied [10], [11], [12]. Agricultural machinery plays an important role in existing agricultural systems. Therefore, its automation directly leads to the automation of agriculture.

As a new element besides agricultural machinery, small robots have been researched to perform automatic measurements and weeding [13], [14], [15]. Unlike large agricultural machines, small robots can enter the field late in the growing season. Thus, they are expected to automate precise and localized field management.

Aerial images from Unmanned Aerial Vehicles (UAV) and satellites have been widely used in agriculture [16]. Deep learning for image processing technology was used to predict yields and diagnose soil conditions [17].

Unmanned Ground Vehicles (UGV) have also been used to capture close-up images and to obtain three-dimensional point-cloud data of the field [18]. Applications of UGV include crop measurement using imagery and soil diagnosis using soil sensors (e.g., soil penetration tester) [13], [14], [15].

As an example of robotics research to improve agricultural efficiency, we have developed a system to reduce fertilizer use while maintaining yield through variable fertilization [19] based on multi-point soil measurement by a robot [20].

The use of robotic arms has become widespread in the agricultural sector. Automating fruit harvesting with robotic arms helps save labor in agriculture, as it is a labor-intensive process usually performed manually. Applications of robotic arm include eggplant [21] and apple harvesting [22].

To simulate the movement of these robots, the application of existing robot simulation technology to agricultural robots has been the subject of numerous studies [5]. The existing dynamics engines allow real-time simulation of general multi-joint robots consisting of tens of degrees of freedom [23], [24], [25]. In the research field of agricultural robots, SEARFS [6], CoppeliaSim [7] and Gazebo [26] are the famous simulation tools using the dynamics engines.

These studies lacked the ability to efficiently handle 3D point-cloud field models, which are commonly used in recent agricultural research. In addition, there were few comparative experiments between actual and simulated robot motions. This paper addresses these issues.

The challenge of the simulator in this study is twofold: 1. to achieve high speed operation on dense point-cloud data ($> 10^6$ points) with centimeter resolution; and 2. to handle contact with the environment represented by point-cloud data instead of the triangular mesh data commonly used in existing simulators. Our simulator is a new dynamics engine specialized for robot simulation on point-cloud data.

E. ORGANIZATION OF THIS PAPER

Section II describes a fast robot simulation method on agricultural fields represented by point-cloud data. The method includes a collision detection method between

TABLE 1. Symbols used in the simulation.

i-th joint	
τ_i	Joint torque
f_i^e, n_i^e	External force and moment
q_i	Joint angle
p_i, R_i	Position vector and rotation matrix
a_i	Axis vector
c_i	Center of gravity
I_i	Inertia matrix
m_i	Mass (constant)
P_i, L_i	Momentum and angular momentum
$v_i, \boldsymbol{\omega}_i, u_i$	Velocity, angular velocity and spatial velocity
f_i, n_i	Inner force and moment
p_{ji}	Position vector seen from j-th joint (constant)
R_{ji}	Rotation matrix seen from j-th joint (constant)
a_{ii}	Axis vector seen from i-th joint (constant)
c_{ii}	Centroid seen from i-th joint (constant)
I_{ii}	Inertia matrix seen from i-th joint (constant)
A_i, V_i, Ω_i	Electric current, voltage and resistance of actuator
A_i^{max}, V_i^{max}	Maximum electric current and voltage of actuator
k_i^τ, k_i^ω	DC-motor constant (i.e., $\tau_i = k_i^\tau A_i$, $V_i = k_i^\omega \omega_i$)
$\iota_i, \boldsymbol{\zeta}_i, \eta_i$	Temporary variables required to solve forward dynamics
$\Delta u_i, \Delta\boldsymbol{\omega}_i$	$p_i \times a_i, a_i$
$\ddot{x}_i, y_i, \dot{s}_i$	$[\dot{u}_i; \dot{\boldsymbol{\omega}}_i], [f_i; n_i], [\Delta u_i; \Delta\boldsymbol{\omega}_i]$
M_i, z_i	$y_i - \sum_k y_k = M_i \ddot{x}_i + z_i$
M_i^A, z_i^A	$y_i = M_i^A \ddot{x}_i + z_i^A$

i,j-th point cloud	
b_{ijk}	Vertical distance from contact surface
b_{ijk}^h	Horizontal distance from contact surface
b_{ijk}^0	b_{ijk} before a simulation time step
p_{ij}^c	Position vector
p_{ijk}^c	Position vector seen from k-th joint
p_{ijk}^{c0}	p_{ijk}^c before a simulation time step
v_{ijk}^c	Velocity vector seen from k-th joint
f_{ijk}^v	Vertical contact force to contact surface
f_{ijk}^h	Horizontal contact force on contact surface
k_{ij}	Vertical elasticity of contact (elastic constant)
d_{ij}	Vertical viscosity of contact (viscosity constant)
k_{ij}^h	Horizontal elasticity of contact (elastic constant)
d_{ij}^h	Horizontal viscosity of contact (viscosity constant)
μ_{ij}, μ_{ij}^d	Friction coefficient and dynamic friction coefficient
w-th robot mesh	
n_w	Normal vector
n_{wk}	Normal vector seen from k-th joint
p_w^l	Position vector
Other constants	
Δt	Simulation time step
g	Gravitational acceleration
E	Unit matrix

point-cloud data and robots, and a contact force calculation method. In addition, the section also summarizes the robot kinematics and dynamics calculation.

Section III evaluates the simulator. Three experiments were conducted: 1. computational time and memory usage; 2. stability of the simulation results with respect to changes in simulation time step; and 3. comparison of actual and simulated robot movement.

Section IV discusses, and Section V concludes.

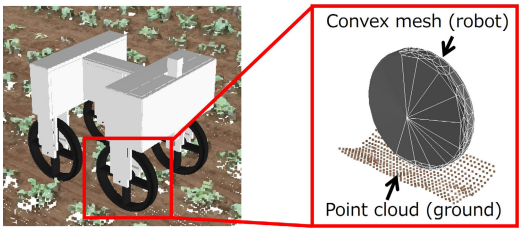

FIGURE 2. Shape model of robot and agricultural field.

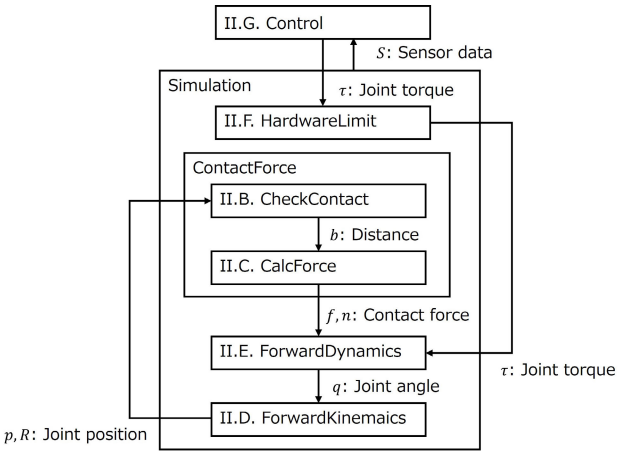

FIGURE 3. Computational flow of simulation.

II. METHOD

This section describes the symbols and algorithms used in this paper. The methods for collision detection and contact force calculation on point-cloud environments are summarized. Furthermore, supplementary information on kinematics and dynamics calculations is provided.

A. SYMBOLS AND ENTIRE ALGORITHM

The symbols used in this paper are summarized in Table 1. $\dot{\boldsymbol{x}}$ is the derivative of $\boldsymbol{x}$ in time and $\ddot{\boldsymbol{x}}$ is the second-order derivative. $\boldsymbol{x}_{ij}$ denotes a j-th joint value seen from the i-th joint. If the viewpoint coordinate i is the world coordinate system (origin), it is written as $\boldsymbol{x}_j$ omitting i.

The robot model is represented by convex triangular meshes and the field model is represented by 3D point-cloud data (Fig. 2). Non-convex shapes can be used by splitting them into convex sub-shapes. Note that although a cylindrical model is more computationally efficient for wheeled robots, the mesh model is applicable to more general shapes and is suitable for simulator evaluation.

Unique IDs are assigned to all joints, faces, and vertices to be distinguished. In the algorithms, these IDs are accessed by the following functions:

Algorithm 1 Pseudo Code of Robot Simulation

Global variable: Δt, ϕ
Δt: simulation time step.
ϕ: root link id.

Procedure: RobotSimulation()
while True **do**
  Control() ## Section II-G
  Simulation()
end while
Procedure: Simulation()
HardwareLimit(ϕ) ## Section II-F
ContactForce(ϕ) ## Section II-B,II-C
ForwardDyanmics(ϕ) ## Section II-E
ForwardKinematics(ϕ) ## Section II-D

Procedure: HardwareLimit(i)
if $\tau_i < -A_i^{max} k_i^{\tau}$ **then** $\tau_i = -A_i^{max} k_i^{\tau}$ **end if**
if $\tau_i > A_i^{max} k_i^{\tau}$ **then** $\tau_i = A_i^{max} k_i^{\tau}$ **end if**
if $\tau_i < (-V_i^{max} - \omega_i k_i^{\omega}) k_i^{\tau} / \Omega_i$ **then**
  $\tau_i = (-V_i^{max} - \omega_i k_i^{\omega}) k_i^{\tau} / \Omega_i$
end if
if $\tau_i > (V_i^{max} - \omega_i k_i^{\omega}) k_i^{\tau} / \Omega_i$ **then**
  $\tau_i = (V_i^{max} - \omega_i k_i^{\omega}) k_i^{\tau} / \Omega_i$
end if
for all k in CHILDREN(i) **do**
  HardwareLimit(k)
end for

Procedure: ContactForce(i)
CheckContact(i) ## Algorithm 2
for all k in CHILDREN(i) **do**
  ContactForce(k)
end for

Procedure: ForwardKinematics(i)
for all k in CHILDREN(i) **do**
  FK(k, i)
end for
Procedure: FK(i, j)
$\boldsymbol{p}_i = \boldsymbol{p}_j + R_j \boldsymbol{p}_{ji}$
$R_i = R_j R_{ji} e^{[\boldsymbol{a}_{ii}\times]q_i}$
$\boldsymbol{a}_i = R_i \boldsymbol{a}_{ii}$
$\boldsymbol{c}_i = R_i \boldsymbol{c}_{ii}$
$I_i = R_i I_{ii} R_i^T + m_i [\boldsymbol{c}_i\times][\boldsymbol{c}_i\times]^T$
$\Delta \boldsymbol{u}_i = \boldsymbol{p}_i \times \boldsymbol{a}_i$
$\Delta \boldsymbol{\omega}_i = \boldsymbol{a}_i$
$\Delta \dot{\boldsymbol{u}}_i = \boldsymbol{u}_j \times \Delta \boldsymbol{\omega}_i + \boldsymbol{\omega}_j \times \Delta \boldsymbol{u}_i$
$\Delta \dot{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_j \times \Delta \boldsymbol{\omega}_i$
$\boldsymbol{u}_i = \boldsymbol{u}_j + \Delta \boldsymbol{u}_i \dot{q}_i$
$\boldsymbol{\omega}_i = \boldsymbol{\omega}_j + \Delta \boldsymbol{\omega}_i \dot{q}_i$
$\dot{\boldsymbol{u}}_i = \dot{\boldsymbol{u}}_j + \Delta \dot{\boldsymbol{u}}_i \dot{q}_i + \Delta \boldsymbol{u}_i \ddot{q}_i$
$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_j + \Delta \dot{\boldsymbol{\omega}}_i \dot{q}_i + \Delta \boldsymbol{\omega}_i \ddot{q}_i$
for all k in CHILDREN(i) **do**
  FK(k, i)
end for

Procedure: ForwardDynamics(i)
FD1(i) ## calculate mass properties
$\ddot{\boldsymbol{x}}_i = -\left(M_i^A\right)^{-1} \boldsymbol{x}_i^A$
for all k in CHILDREN(i) **do**
  FD2(k, i) ## calculate acceleration
end for
FD3(i) ## integrate acceleration
$\boldsymbol{p}_i = \boldsymbol{p}_i + (\boldsymbol{u}_i + \boldsymbol{\omega}_i \times \boldsymbol{p}_i)\Delta t$
$R_i = e^{[\boldsymbol{\omega}_i\times]\Delta t} R_i$
$\boldsymbol{u}_i = \boldsymbol{u}_i + \dot{\boldsymbol{u}}_i \Delta t$
$\boldsymbol{\omega}_i = \boldsymbol{\omega}_i + \dot{\boldsymbol{\omega}}_i \Delta t$
Procedure: FD1(i)
for all k in CHILDREN(i) **do**
  FD1(k)
end for
$\boldsymbol{P}_i = m_i \boldsymbol{u}_i + m_i \boldsymbol{\omega}_i \times \boldsymbol{c}_i$
$\boldsymbol{L}_i = m_i \boldsymbol{c}_i \times \boldsymbol{u}_i + I_i \boldsymbol{w}_i$
$M_i = \begin{pmatrix} m_i E & -m_i[\boldsymbol{c}_i\times] \\ m_i[\boldsymbol{c}_i\times] & I_i \end{pmatrix}$
$\boldsymbol{z}_i = \begin{pmatrix} -m_i \boldsymbol{g} - \boldsymbol{f}_i^e + \boldsymbol{\omega}_i \times \boldsymbol{P}_i \\ -m_i \boldsymbol{c}_i \times \boldsymbol{g} - \boldsymbol{n}_i^e + \boldsymbol{u}_i \times \boldsymbol{P}_i + \boldsymbol{\omega}_i \times \boldsymbol{L}_i \end{pmatrix}$
$\dot{\boldsymbol{s}}_i = \begin{pmatrix} \Delta \boldsymbol{u}_i \\ \Delta \boldsymbol{\omega}_i \end{pmatrix}$
$\ddot{\boldsymbol{s}}_i = \begin{pmatrix} \Delta \dot{\boldsymbol{u}}_i \\ \Delta \dot{\boldsymbol{\omega}}_i \end{pmatrix}$
$\boldsymbol{\zeta}_i^T = \dot{\boldsymbol{s}}_i^T M_i^A$
$\iota_i = \left(\boldsymbol{\zeta}_i^T \dot{\boldsymbol{s}}_i\right)^{-1}$
$M_i^A = M_i$
$\boldsymbol{z}_i^A = \boldsymbol{z}_i$
for all k in CHILDREN(i) **do**
  $M_i^A = M_i^A + \left(M_k^A - M_k^A \dot{\boldsymbol{s}}_k \boldsymbol{\zeta}_k^T \iota_k\right)$
  $\boldsymbol{z}_i^A = \boldsymbol{z}_i^A + \left(\boldsymbol{z}_k^A + M_k^A \ddot{\boldsymbol{s}}_k \dot{q}_k + M_k^A \dot{\boldsymbol{s}}_k (\tau_k - \eta_k) \iota_k\right)$
end for
$\eta_i = \boldsymbol{\zeta}_i^T \ddot{\boldsymbol{s}}_i \dot{q}_i + \dot{\boldsymbol{s}}_i^T \boldsymbol{z}_i^A$
Procedure: FD2(i, j)
$\ddot{q}_i = \left(\tau_i - \boldsymbol{\zeta}_i^T \ddot{\boldsymbol{x}}_j - \eta_i\right) \iota_i$
$\ddot{\boldsymbol{x}}_i = \ddot{\boldsymbol{x}}_j + \ddot{\boldsymbol{s}}_i \dot{q}_i + \dot{\boldsymbol{s}}_i \ddot{q}_i$
for all k in CHILDREN(i) **do**
  FD2(k, i)
end for
Procedure: FD3(i)
$q_i = q_i + \dot{q}_i \Delta t$
$\dot{q}_i = \dot{q}_i + \ddot{q}_i \Delta t$
for all k in CHILDREN(i) **do**
  FD3(k)
end for

1. "CHILDREN": returns the IDs of all child joints.
2. "FACES": returns the IDs of all faces.
3. "VERTICES": returns the IDs of all vertices.

All procedures are shown in Fig. 3 and Algorithm 1. The "Simulation" function is performed for each simulation time step Δt. Its computation is a repetition of the following four functions:

1. "HardwareLimit": limits the joint torques according to the actuator model.
2. "ContactForce": detects collisions with the environment to calculate contact forces.
3. "ForwardDyanmics": calculates the joint angular accelerations of the robot from the contact forces and joint torques by solving the equations of motion of the robot.
4. "ForwardKinematics": calculates the position and attitude of each joint of the robot from the joint angles by kinematical calculation.

The joint torques, which are the control input to the robot, are calculated by the "Control" function. The "Control" and "Simulation" functions are repeated to simulate the robot system.

The following sections explain each of the functions. The formula derivations are summarized in Appendix A.

B. COLLISION DETECTION METHOD

We describe a method to quickly find collision points with a robot in the point-cloud data of agricultural fields. The overall procedure is shown in Algorithm 2. The algorithm is used in the "ContactForce" function in Algorithm 1.

Due to the large number of points involved in a centimeter-accurate field model, it is important to quickly filter out points not colliding with the robot. For that purpose, the field point cloud is divided into a grid form as shown in Fig. 4. Only points within the projected area of the robot are sufficient for collision calculations.

All points are arranged in Δx increments along the x-axis with IDs assigned as $x_0, x_1, x_2, \cdots$. The following equation holds for the i-th point:

$$x_i = x_0 + i\Delta x \tag{1}$$

Let x_{min} be the x-minimum value of the robot mesh vertices, then its neighbor point ID (i) can be calculated as follows:

$$i = \left[\frac{x_{min} - x_0}{\Delta x} \right] \tag{2}$$

Note that $[x]$ is the largest integer not exceeding x (i.e., the Gaussian symbol).

The same calculation can be performed on the y-axis to compute the bounding box of the projected region of the robot mesh. The collision points can be obtained by checking the collision for all points in the bounding box.

C. CONTACT FORCE CALCULATION METHOD

We model the collision between the ground and the robot mesh using viscoelastic deformation to account for the contact forces. As shown in Fig. 5, each point on the ground is subjected to a force proportional to its penetration distance and velocity into the robot link mesh. The integral of the forces is used to calculate the force applied to the robot.

The entire procedure is shown in Algorithm 3. The algorithm is used in Algorithm 2.

Algorithm 2 Collision Calculation Between the k-Th Joint and the Field Point Cloud

Procedure: CheckContact(k)

$\boldsymbol{f}_k^e = \mathbf{0}, \boldsymbol{n}_k^e = \mathbf{0}$
$x_{min}, x_{max}, y_{min}, y_{max} = \text{BoundingBox}(k)$
for all $i \in ([\frac{x_{min}-x_0}{\Delta x}], [\frac{x_{max}-x_0}{\Delta x}])$ **do**
  for all $j \in ([\frac{y_{min}-y_0}{\Delta y}], [\frac{y_{max}-y_0}{\Delta y}])$ **do**
    $b_{ijk}^0 = b_{ijk},\ b_{ijk} = -\infty$
    for all w in FACES(k) **do**
      $\boldsymbol{p}_{ijk}^c = R_k^T \left(\boldsymbol{p}_{ij}^c - \boldsymbol{p}_k \right)$
      $b = \boldsymbol{n}_{wk}^T \left(\boldsymbol{p}_{ijk}^c - \boldsymbol{p}_{wk}^l \right)$
      if $b > b_{ijk}$ **then**
        $b_{ijk} = b$
        $w_{ijk} = w$
        if $b > 0$ **then break end if**
      end if
    end for
    CalcForce(i, j, k, w_{ijk}) ## Algorithm 3
  end for
end for

Procedure: BoundingBox(k)

$x_{min} = \infty,\ y_{min} = \infty,\ x_{max} = -\infty,\ y_{max} = -\infty$
for all w in FACES(k) **do**
  for all v in VERTICES(w) **do**
    if $x_{min} > x_v$ **then** $x_{min} = x_v$ **end if**
    if $y_{min} > y_v$ **then** $y_{min} = y_v$ **end if**
    if $x_{max} < x_v$ **then** $x_{max} = x_v$ **end if**
    if $y_{max} > y_v$ **then** $y_{max} = y_v$ **end if**
  end for
end for
return $x_{min}, x_{max}, y_{min}, y_{max}$

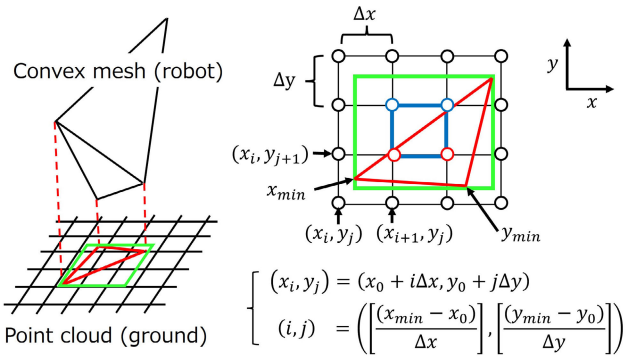

FIGURE 4. Collision detection between robot mesh and point cloud.

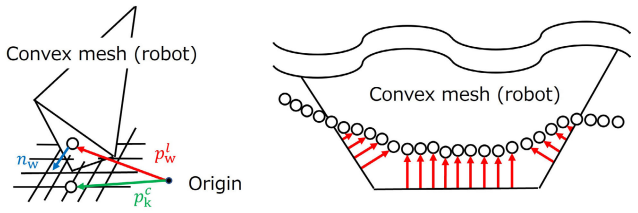

FIGURE 5. Contact force between robot mesh and point-cloud data.

Algorithm 3 Contact Force Calculation Between the w-Th Mesh of the k-Th Joint and the i,j-Th Field Point Cloud

Procedure: CalcForce(i, j, k, w)

if $b_{ijk} > 0$ **then**
  $\boldsymbol{f}^h_{ijk} = \boldsymbol{0}$, $\boldsymbol{f}^v_{ijk} = \boldsymbol{0}$ ## No collision
else
  $\boldsymbol{v}^c_{ijk} = -(\boldsymbol{u}_k + \boldsymbol{\omega}_k \times \boldsymbol{p}^c_{ij})$
  $\boldsymbol{n}_w = R_k \boldsymbol{n}_{wk}$
  $||\boldsymbol{f}^v_{ijk}|| = -k_{ij} b_{ijk} - d_{ij} \boldsymbol{n}_w^T \boldsymbol{v}^c_{ijk}$
  if $||\boldsymbol{f}^v_{ijk}|| < 0$ **then** $||\boldsymbol{f}^v_{ijk}|| = 0$ **end if**
  $\boldsymbol{f}^v_{ijk} = ||\boldsymbol{f}^v_{ijk}|| \boldsymbol{n}_w$
  if $b^0_{ijk} > 0$ **then**
    $\boldsymbol{p}^{c0}_{ijk} = R_k^T (\boldsymbol{p}^c_{ij} - \boldsymbol{p}_k)$
    $\boldsymbol{f}^h_{ijk} = \boldsymbol{0}$ ## No friction
  else
    $\boldsymbol{p}^{c0}_{ij} = \boldsymbol{p}_k + R_k \boldsymbol{p}^{c0}_{ijk}$
    $\boldsymbol{b}^h_{ijk} = (\boldsymbol{p}^c_{ij} - \boldsymbol{p}^{c0}_{ij}) - \boldsymbol{n}_w^T (\boldsymbol{p}^c_{ij} - \boldsymbol{p}^{c0}_{ij}) \boldsymbol{n}_w$
    $\boldsymbol{f}^h_{ijk} = -k^h_{ij} \boldsymbol{b}^h_{ijk} - d^h_{ij} (\boldsymbol{v}^c_{ijk} - \boldsymbol{n}_w^T \boldsymbol{v}^c_{ijk} \boldsymbol{n}_w)$
    if $||\boldsymbol{f}^h_{ijk}|| > ||\boldsymbol{f}^v_{ijk}|| \mu_{ij}$ **then**
      $\boldsymbol{f}^h_{ijk} = ||\boldsymbol{f}^v_{ijk}|| \mu^d_{ij} \frac{\boldsymbol{f}^h_{ijk}}{||\boldsymbol{f}^h_{ijk}||}$
      $\boldsymbol{p}^{c0}_{ijk} = R_k^T (\boldsymbol{p}^c_{ij} - \boldsymbol{p}_k)$
    end if
  end if
end if
$\boldsymbol{f}^e_k = \boldsymbol{f}^e_k + (\boldsymbol{f}^v_{ijk} + \boldsymbol{f}^h_{ijk})$
$\boldsymbol{n}^e_k = \boldsymbol{n}^e_k + \boldsymbol{p}^c_{ij} \times (\boldsymbol{f}^v_{ijk} + \boldsymbol{f}^h_{ijk})$

From the geometrical relationship shown on the left side of Fig. 5, the distance between a ground point and a robot mesh can be calculated using the following equation:

$$b_{ijk} = \boldsymbol{n}^T_{wk} \left(\boldsymbol{p}^c_{ijk} - \boldsymbol{p}^l_{wk} \right) \tag{3}$$

If the distance is less than 0, the contact force due to the collision is calculated. The force acting in the normal direction of the contact surface is calculated as follows:

$$\begin{aligned} \boldsymbol{f}^v_{ijk} &= ||\boldsymbol{f}^v_{ijk}|| \boldsymbol{n}_w \\ ||\boldsymbol{f}^v_{ijk}|| &= -k_{ij} b_{ijk} - d_{ij} \boldsymbol{n}_w^T \boldsymbol{v}^c_{ijk} \end{aligned} \tag{4}$$

Then, we consider the horizontal force on the contact surface (i.e., frictional force). If the distance in a previous calculation (i.e., b^0_{ijk}) is also negative, we can determine that a slip movement is occurring due to successive collisions. The point where the last collision occurred is used as the origin of the frictional force. The calculation of the horizontal force toward the origin is as follows:

$$\begin{aligned} \boldsymbol{f}^h_{ijk} &= -k^h_{ij} \boldsymbol{b}^h_{ijk} - d^h_{ij} (\boldsymbol{v}^c_{ijk} - \boldsymbol{n}_w^T \boldsymbol{v}^c_{ijk} \boldsymbol{n}_w) \\ \boldsymbol{b}^h_{ijk} &= (\boldsymbol{p}^c_{ij} - \boldsymbol{p}^{c0}_{ij}) - \boldsymbol{n}_w^T (\boldsymbol{p}^c_{ij} - \boldsymbol{p}^{c0}_{ij}) \boldsymbol{n}_w \end{aligned} \tag{5}$$

When a frictional force exceeds the static frictional force $||\boldsymbol{f}^v_{ijk}|| \mu_{ij}$, the frictional force is headed by the dynamic frictional force as follows:

$$\boldsymbol{f}^h_{ijk} = ||\boldsymbol{f}^v_{ijk}|| \mu^d_{ij} \frac{\boldsymbol{f}^h_{ijk}}{||\boldsymbol{f}^h_{ijk}||} \tag{6}$$

Furthermore, the origin of the frictional force $\boldsymbol{p}^{c0}_{ijk}$ is updated as follows:

$$\boldsymbol{p}^{c0}_{ijk} = R_k^T (\boldsymbol{p}^c_{ij} - \boldsymbol{p}_k) \tag{7}$$

D. KINEMATICS CALCULATION METHOD

The forward kinematics calculation is a procedure to calculate the position and attitude of each joint from the joint angles. From the geometric relationship shown in Fig. 6, the position and attitude of the i-th joint is calculated as follows:

$$\begin{aligned} \boldsymbol{p}_i &= \boldsymbol{p}_j + R_j \boldsymbol{p}_{ji} \\ R_i &= R_j R_{ji} e^{[\boldsymbol{a}_{ii} \times] q_i} \end{aligned} \tag{8}$$

The overall procedure is shown in the "ForwardKinematics" function of Algorithm 1. By computing (8) recursively from the parent link to the child links, the positions and attitudes of all joints are obtained. The computational time complexity is $\mathcal{O}(N)$ for N-dof joints.

The velocity and acceleration of each joint can be obtained similarly by the derivative of (8). The calculation is supplemented in Appendix A-B.

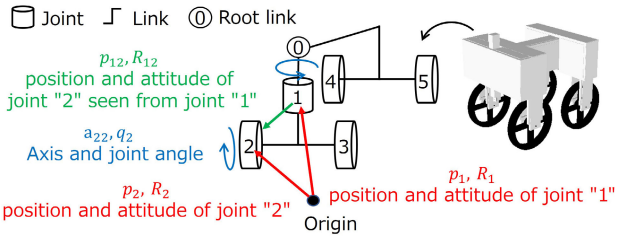

FIGURE 6. Geometric relations between joints.

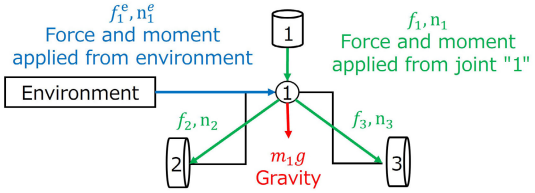

FIGURE 7. Dynamic balance between joints.

E. DYNAMICS CALCULATION METHOD

The forward dynamics calculation is a procedure to calculate the acceleration of each joint from the contact forces and joint torques. The procedure involves solving the equations of motion of a multi-link system.

The equations of motion for rigid bodies are known to be a balanced form of the derivatives of momentum with all forces. As shown in Fig. 7, all forces acting on a multi-link system include: internal forces with all connected joints (i.e., $\boldsymbol{f}_i, \boldsymbol{n}_i, \sum_k -\boldsymbol{f}_k, \sum_k -\boldsymbol{n}_k$); external forces due to contact with the environment (i.e., $\boldsymbol{f}_i^e, \boldsymbol{n}_i^e$); and gravitational forces (i.e., $m_i\boldsymbol{g}, \boldsymbol{c}_i \times m_i\boldsymbol{g}$). Thus, the equations of motion take the following form:

$$\begin{aligned}\dot{\boldsymbol{P}}_i &= \boldsymbol{f}_i - \sum_k \boldsymbol{f}_k + m_i\boldsymbol{g} + \boldsymbol{f}_i^e \\ \dot{\boldsymbol{L}}_i &= \boldsymbol{n}_i - \sum_k \boldsymbol{n}_k + \boldsymbol{c}_i \times m_i\boldsymbol{g} + \boldsymbol{n}_i^e\end{aligned} \tag{9}$$

It is possible to solve the equation for the acceleration in $\mathcal{O}(N)$ for N-dof joints by using Featherstone's method [27]. The procedure is shown in the "ForwardDyanmics" function of Algorithm 1. The formulas needed for the calculation are summarized in Appendix A-C.

F. HARDWARE LIMITATIONS

In the following experiments, we used BrushLess Direct Current (BLDC) motors as the robot actuators. The actuator system includes two limitations on the amount of electric current applied to the motors as follows:

$$\begin{aligned}-A_i^{max} &\le A_i \le A_i^{max} \\ (-V_i^{max} - V_i)/\Omega_i &\le A_i \le (V_i^{max} - V_i)/\Omega_i\end{aligned} \tag{10}$$

The first limitation is the maximum amount of current that can flow through the motor driver. Another limitation comes from the back electromotive force generated by the rotation of the motor.

The torque and angular velocity of each joint are linearly proportional to the current and voltage as follows:

$$\begin{aligned}A_i &= \tau_i / k_i^\tau \\ V_i &= \omega_i k_i^\omega\end{aligned} \tag{11}$$

Thus, the current limitations (10) can be written as the following torque limitations:

$$\begin{aligned}-A_i^{max} k_i^\tau &\le \tau_i \le A_i^{max} k_i^\tau \\ (-V_i^{max} - \omega_i k_i^\omega) k_i^\tau / \Omega_i &\le \tau_i \le (V_i^{max} - \omega_i k_i^\omega) k_i^\tau / \Omega_i\end{aligned} \tag{12}$$

G. CONTROL METHOD

We describe the "Control" function to calculate the actuator input from the robot sensor information. Since the function is independent of the simulation, the implementation of the control system is not limited to the following.

In the following experiments, the weighted sum of the sensor data derivatives is used as the target wheel speed as follows:

$$\dot{q}_i^{ref} = \theta_i^a + \boldsymbol{\psi}^T \boldsymbol{\theta}_i^b + \dot{\boldsymbol{\psi}}^T \boldsymbol{\theta}_i^c + \ddot{\boldsymbol{\psi}}^T \boldsymbol{\theta}_i^d \tag{13}$$

TABLE 2. Computational speed and memory usage..

	Number of points	Time [us]	RSS [MB]
w/ ground	1.7×10^6	114	2311
w/o ground	0	9	158

$\boldsymbol{\psi}$ is a vector consisting of the sensor data. The robot used in the experiment is equipped with two RTK-GNSS antennas to accurately measure the robot position and attitude. As the sensor data, we used the error in position and direction from a given target trajectory. The derivative of the sensor data was calculated by taking the difference at each control cycle. θ_i^{a-d} are the weights to calculate the target rotational speeds of the wheels from the sensor data.

The target speeds are converted to wheel torques by PD control as follows:

$$\Delta\tau_i = w_p(\dot{q}_i^{ref} - \dot{q}_i) - w_d\ddot{q}_i \tag{14}$$

The parameters $(\theta_i^{a-d}, w_p, w_d)$ are constant values adjusted in advance on the simulator.

III. EVALUATION

This section describes three experiments: 1. evaluating the computational performance; 2. confirming the computational stability with respect to the simulation time step; and 3. comparing the robot movement in the simulation with that in an actual field.

The algorithms were implemented in the C++ language. The visualization of the robot and the field model was implemented in Euslisp [28], a Lisp dialect specialized for robotics research.

We used one CPU core (Intel(R) Core(TM) i7-10700K CPU @ 3.80 GHz) and one RAM (DDR4, 32 GB, 3200 MHz). Parallel computing and GPU were not used.

We used Agrisoft's Metashape(R) ver. 1.8 for 3D modeling of the agricultural field. A drone was used to capture images of a 31 × 14 m^2 field from an altitude of 10 m to generate point-cloud data with 2 × 2 cm^2 resolution.

A. COMPUTATIONAL PERFORMANCE

The computation time and memory usage of the simulation are shown in Table 2. We compared two simulations: with and without ground. The latter simulated the free fall motion of the robot.

The computational time was measured as the average time of 10000 runs of the "Simulation" function (Algorithm 1). The time was 114 us with the ground and 9 us without the ground. The difference between the two times (105 us) was the time needed for the collision calculation. Therefore, $100\times 105/114 \sim 92\%$ was used for the collision calculation.

The memory usage was checked against the Resident Set Size (RSS) of Linux. The memory was 2311 MB with the ground and 158 MB without the ground. The difference between the two (2153 MB) was the memory required for the

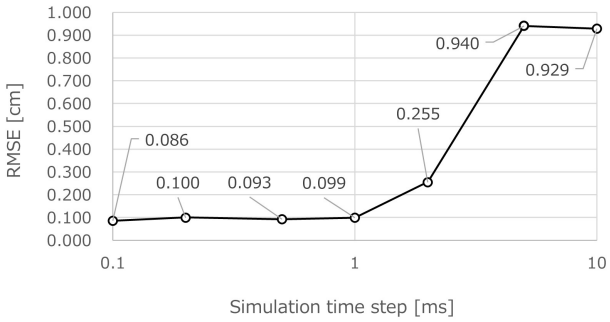

FIGURE 8. Computational stability regarding the simulation time step.

point-cloud data. Therefore, $100 \times 2153/2311 \sim 93\%$ was used for the field model.

B. COMPUTATIONAL STABILITY WITH RESPECT TO THE SIMULATION TIME STEP

The change of the Root Mean Square Error (RMSE) in the simulation results when the simulation time step Δt is varied is shown in Fig. 8. The error was calculated in the following steps:

1. Change the simulation time step $\Delta t \in \{0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10\}$ ms.
2. Simulate the robot moving 2 m by straddling the ridge and planted crops.
3. Sample the position of the moving robot at 20 Hz.
4. Calculate the RMSE of the sampled positions by comparing the results with $\Delta t = 0.01$ ms and the others.

The RMSE was 0.1 cm or less for the simulation time step of less than 1 ms. The RMSE was greater than 0.9 cm for the 10 ms simulation time step. The simulation did not collapse for all time steps.

C. COMPARISON OF ACTUAL DATA AND SIMULATION

To evaluate the predictive performance of the simulation, we compared the results of the robot movements in the actual field and in the simulation. The movements to be compared were straight motions of 2 m. The locations were six areas in the field as shown in Fig. 9 (Area $1 \sim 6$). The same control system and parameters were used in the actual field and in the simulation.

The positions of the actual robot were sampled at 10 Hz by using the RTK-GNSS module [29]. The sampled positions were compared with the simultaneous robot positions in the simulation to calculate the RMSE. With the RTK-GNSS module, the robot position can be measured with centimeter accuracy at a low cost.

The coordinate system expressing the point-cloud model could be associated with latitude and longitude because the drone used to capture images of the agricultural field was equipped with a similar RTK-GNSS module as the robot. Thus, the simulation and the actual robot could use the same coordinate system, allowing easy comparison of positions.

TABLE 3. Simulation accuracy compared to actual robot movements..

Area No.	1	2	3	4	5	6
RMSE [cm]	0.94	0.69	1.00	0.82	0.59	1.06

The latitude and longitude were converted to a rectangular coordinate by using the PROJ library [30].

In order to start the simulation from the same position as the actual robot, the initial position was determined in the following three steps: 1. nine different initial positions were generated by shifting the initial position of the actual robot by 3 cm from left to right and back to front; 2. one second simulation was performed from each initial position; and 3. the one closest to the actual robot was used as the initial position.

The ground contact parameters (e.g., viscoelasticity) were unknown because the field model was generated from aerial images. Therefore, we varied the viscoelasticity parameter to the extent that the simulation did not collapse to investigate the distribution of the RMSE. The minimum RMSE for each area was $\sim$ 1 cm (average was 0.88 cm, minimum was 0.69 cm and maximum was 1.06 cm).

The results are shown in Fig. 10 and Table 3. The vertical axis of Fig. 10 represents the ground stiffness ($k_{ij}^{(h)} \in \{0.1, 0.2, \cdots, 3\}$) and the horizontal axis represents the ground viscosity ($d_{ij}^{(h)} \in \{0.005, 0.01, \cdots, 0.3\}$). The friction coefficient ($\mu_{ij}^{(d)} \in \{0.2, 0.3, \cdots, 1.0\}$) with the lowest RMSE was used to plot Fig. 10.

IV. DISCUSSION

We have developed a dynamics simulator that works fast on 3D point-cloud models of agricultural fields. The simulator has the following advantages:

1. Immediate construction of simulation environments by modeling arbitrary fields from aerial photographs.
2. Evaluation of robot movement not limited to the growing season allowing robot development, evaluation, and data collection at any time.

The memory used by the simulation process was 2.3 GB when a 31×14 m^2 agricultural field was modeled as 3D point-cloud data with 1.7×10^6 points (Section III-A). Therefore, a computer with 32 GB ($>$ 23 GB) of memory can efficiently collect data by running 10 or more simulations in parallel.

Simulation on larger fields will be possible by using virtual memory function. On the Linux system, it is easy to use the "mmap" function to map file into memory. The memory size used by the simulator will increase linearly with the field size (i.e., $31 \times 14/2.3 \sim 189$ m^2/GB $= 18.9$ ha/TB). However, the file access will slow down the computation. Further research on the efficient use of virtual memory is needed.

The computational time for a simulation time step in the same simulation environment was about 114 us (Section III-A). Therefore, 8.8 times faster simulation than real time is possible if the simulation time step is 1 ms.

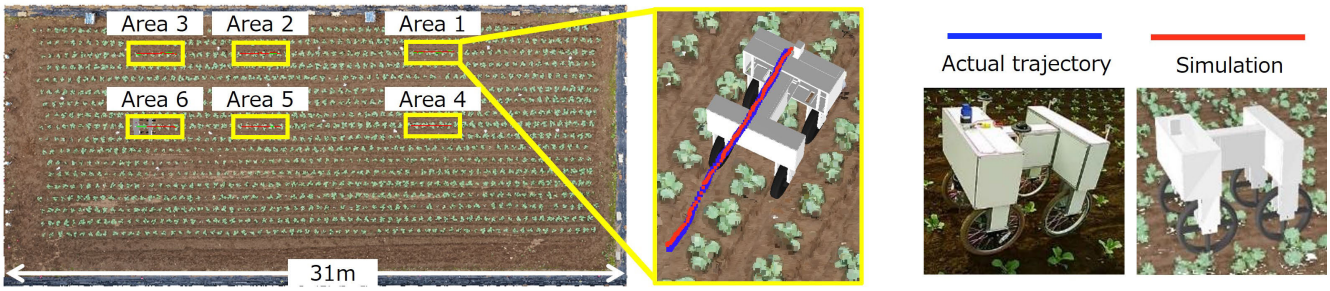

FIGURE 9. Comparison of the robot movement in the actual field and simulation.

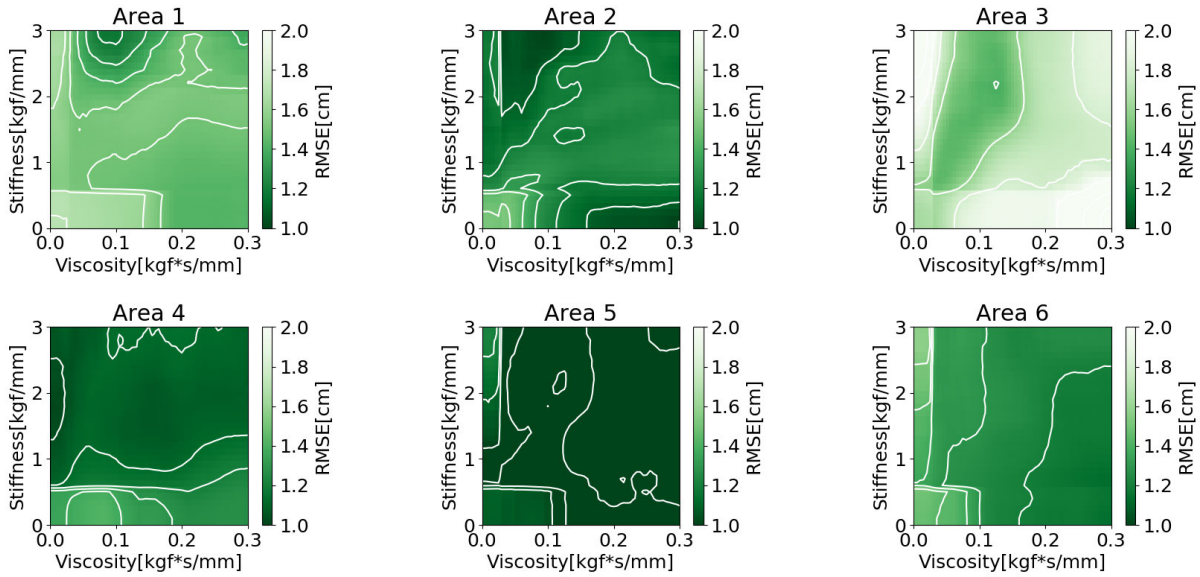

FIGURE 10. RMSE distribution between simulated and actual robot positions when the viscoelasticity of the ground is varied.

The advantage of real-time simulation is that human can interactively check the results of the robot movements. This allows for efficient robot motion generation and evaluation.

The advantage of simulating in more than real time is that it allows for faster collection of teacher data for learning. Our simulator can calculate $24 \times 3600 \times 8.8 = 760320$ seconds of robot movement per day. This means that a 10-second robot movement can be simulated 76032 times.

To speed up the simulation, fast collision calculations are important because 92 % of the computation time was spent on collision calculations as shown in Section III-A. One idea for speeding up is to parallelize the collision calculations. Parallelization is easy because the collision calculation is performed on points distributed on a grid.

In Section III-B, we examined the change in simulation results when the simulation time step Δt was varied. For $\Delta t \leq 1$ ms, the simulation results of the robot position were almost unchanged. Therefore, by using $\Delta t = 1$ ms, we can obtain fast and stable simulation results.

For $\Delta t = 10$ ms, the errors of the simulation results were more than a factor of nine (RMSE$>$ 0.9 cm). However, the simulation calculation did not collapse. By sacrificing accuracy, ten times faster calculations will be possible. Note that more dynamic behavior may be difficult to simulate due to divergence of the integral results.

The prediction error of the actual robot movement was confirmed to be RMSE $\sim$ 1 cm by varying the stiffness and viscosity of the ground (Section III-C). Therefore, by learning driving control on this simulation for various ground parameters, a robust control system can be obtained for real environments.

The ground parameters with the lowest RMSE might reflect the true values. It is an interesting future work to compare the estimated and measured values of the ground hardness. Since soil hardness is an important parameter for crop growth, measurement with a soil penetration tester is often necessary. Simulation of agricultural machines could become a new soil penetration tester.

The RMSE value is a performance baseline allowing easy comparative experiments. The value can vary depending on the experimental settings (e.g., driving distance and trajectory). We evaluated 2 m straight driving movement. However, in actual robot applications, longer driving with non-linear trajectory is necessary.

The reason for using the short driving distance is to obtain the simulation performance independent of ground parameter errors by comprehensively changing the parameters. As shown in Section III-C, we used 14400 parameter combinations (i.e., 30 for $k_{ij}^{(h)}$, 60 for $d_{ij}^{(h)}$ and 8 for $\mu_{ij}^{(d)}$). The computation was completed in < 10 hours.

Steering motion is important as well as linear trajectory because agricultural robot repeatedly moves between furrows. The simulator can calculate the steering behavior including wheel skidding and slipping. The evaluation of the steering simulation is an important future work.

The simulator in this paper is not limited to use with specific robots. Dynamics calculation methods available for general articulated robots are used. Convex triangular mesh structure is used to represent the shape data of the robot. Non-convex shapes can also be expressed by combining convex sub-shapes. Thus, it can be used for wheeled robots, arm robots and legged robots.

Fast calculation method considering ground and crop deformation is a significant future work. Since our simulator modeled the field using rigid bodies, soil deformation and crop movement were ignored. Discrete Element Method (DEM [31]) can be used to account for the movement of sand particles and seeds.

The collision calculation was accelerated by arranging the field point-cloud data into a two-dimensional grid map. Although we used two dimensions because we assumed that the model was generated from aerial photographs, the same method can be extended to three dimensions. OctoMap [32] is well known as a prior study that allows fast retrieval of multiple resolutions of 3D point-cloud data.

Rendering technology is an important future work to simulate robot behavior based on camera recognition. It may be possible to generate an image as seen by the robot camera from the colored point-cloud data.

V. CONCLUSION

In this paper, we have developed a dynamics simulator for agricultural robots. The dynamics calculation was specialized to handle dense 3D point-cloud field models which were widely used in agricultural research thanks to recent advances in aerial photography technology. Thus, the simulator can be easily applied to many actual fields.

To speed up the retrieval of collision points, we used an algorithm to table all points in a grid form. To calculate the contact force, we integrated the force between each colliding point and the robot mesh.

As results of the performance evaluation, the computation time was 8.8 times faster than real time, and the Root Mean Square Error between the driving trajectory of the robot in the simulation and that of the actual field was ~ 1 cm.

The simulation allows fast and accurate prediction of robot movements by taking aerial photographs in any agricultural field, and is available for robot development, modification, and data collection not limited to the growing season.

APPENDIX A
FORMULAS USED IN THE SIMULATION

A. DEFINITION OF FORMULAS

1) DEFINITION OF ROTATION MATRIX

The matrix R that rotates an arbitrary vector around a unit vector $\boldsymbol{b}$ by c radians can be defined as follows:

$$R = e^{[\boldsymbol{b}\times]c} = \sum_k \frac{c^k}{k!}[\boldsymbol{b}\times]^k \tag{15}$$

$[\boldsymbol{b}\times]$ is the outer product matrix of $\boldsymbol{b}$, defined as follows:

$$\boldsymbol{b} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}, [\boldsymbol{b}\times] = \begin{pmatrix} 0 & -b_2 & b_1 \\ b_2 & 0 & -b_0 \\ -b_1 & b_0 & 0 \end{pmatrix} \tag{16}$$

For any vectors $(\boldsymbol{a}, \boldsymbol{b})$, the rotation matrix R has the following form:

$$R(\boldsymbol{a} \times \boldsymbol{b}) = (R\boldsymbol{a}) \times (R\boldsymbol{b}) \tag{17}$$

2) DEFINITION OF VELOCITY

Velocity $\boldsymbol{v}_i$ and angular velocity $\boldsymbol{\omega}_i$ are defined as follows:

$$\begin{aligned} \dot{\boldsymbol{p}}_i &= \boldsymbol{v}_i \\ \dot{R}_i &= [\boldsymbol{\omega}_i\times]R_i \end{aligned} \tag{18}$$

From (15), we can see that the differential of the rotation matrix R is the product of the outer product matrix of a vector and R.

3) DEFINITION OF SPATIAL VELOCITY

To transform the equations of motion concisely, spatial velocity is used. The definition of the spatial velocity is as follows:

$$\begin{aligned} \boldsymbol{u}_i &= \boldsymbol{v}_i - \boldsymbol{\omega}_i \times \boldsymbol{p}_i \\ \dot{\boldsymbol{u}}_i &= \dot{\boldsymbol{v}}_i - \dot{\boldsymbol{\omega}}_i \times \boldsymbol{p}_i - \boldsymbol{\omega}_i \times \dot{\boldsymbol{p}}_i \\ &= \dot{\boldsymbol{v}}_i - \dot{\boldsymbol{\omega}}_i \times \boldsymbol{p}_i - \boldsymbol{\omega}_i \times \left(\boldsymbol{u}_i + \boldsymbol{\omega}_i \times \boldsymbol{p}_i\right) \end{aligned} \tag{19}$$

4) DEFINITION OF CENTER OF GRAVITY AND INERTIA MATRIX

Let $\boldsymbol{r}_j$ be the position vector of any point on a rigid body and ρ_j be the mass of that point. The mass m_i, inertia matrix I_i and center of gravity position $\boldsymbol{c}_i$ of the rigid body link are defined as follows:

$$m_i = \sum_j \rho_j,\ \boldsymbol{c}_i = \frac{\sum_j \rho_j \boldsymbol{r}_j}{\sum_j \rho_j},\ I_i = \sum_j \rho_j [\boldsymbol{r}_j\times][\boldsymbol{r}_j\times]^T \tag{20}$$

A position fixed to a rigid body (e.g., center of gravity) and its differential can be calculated as follows:

$$\boldsymbol{c}_i = \boldsymbol{p}_i + R_i \boldsymbol{c}_{ii}$$

$$\begin{aligned}\dot{\boldsymbol{c}}_i &= \boldsymbol{v}_i + \boldsymbol{\omega}_i \times R_i \boldsymbol{c}_{ii} \\ &= \boldsymbol{u}_i + \boldsymbol{\omega}_i \times \boldsymbol{p}_i + \boldsymbol{\omega}_i \times R_i \boldsymbol{c}_{ii} \\ &= \boldsymbol{u}_i + \boldsymbol{\omega}_i \times \boldsymbol{c}_i \\ \ddot{\boldsymbol{c}}_i &= \dot{\boldsymbol{u}}_i + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{c}_i + \boldsymbol{\omega}_i \times \dot{\boldsymbol{c}}_i \\ &= \dot{\boldsymbol{u}}_i + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{c}_i + \boldsymbol{\omega}_i \times (\boldsymbol{u}_i + \boldsymbol{\omega}_i \times \boldsymbol{c}_i) \end{aligned} \tag{21}$$

5) DEFINITION OF MOMENTUM AND ANGULAR MOMENTUM

The definition of the momentum and angular momentum are as follows:

$$\begin{aligned}\boldsymbol{P}_i &= \sum\nolimits_j \rho_j \dot{\boldsymbol{r}}_j = m_i \dot{\boldsymbol{c}}_i \\ &= m_i \boldsymbol{u}_i + m_i \boldsymbol{\omega}_i \times \boldsymbol{c}_i \\ \boldsymbol{L}_i &= \sum\nolimits_j \rho_j \boldsymbol{r}_j \times \dot{\boldsymbol{r}}_j \\ &= \sum\nolimits_j \rho_j \boldsymbol{r}_j \times \left(\boldsymbol{u}_i + \boldsymbol{\omega}_i \times \boldsymbol{r}_j\right) \\ &= \left(\sum\nolimits_j \rho_j \boldsymbol{r}_j\right) \times \boldsymbol{u}_i + \sum\nolimits_j \rho_j \boldsymbol{r}_j \times \left(\boldsymbol{\omega}_i \times \boldsymbol{r}_j\right) \\ &= \left(\sum\nolimits_j \rho_j \boldsymbol{r}_j\right) \times \boldsymbol{u}_i + \sum\nolimits_j \rho_j \left[\boldsymbol{r}_j \times\right] \left[\boldsymbol{r}_j \times\right]^T \boldsymbol{\omega}_i \\ &= m_i \boldsymbol{c}_i \times \boldsymbol{u}_i + I_i \boldsymbol{\omega}_i \end{aligned} \tag{22}$$

The differential of momentum and angular momentum can be calculated using (20) as follows:

$$\begin{aligned}\dot{\boldsymbol{P}}_i &= m_i \dot{\boldsymbol{u}}_i + m_i \dot{\boldsymbol{\omega}}_i \times \boldsymbol{c}_i + m_i \boldsymbol{\omega}_i \times \dot{\boldsymbol{c}}_i \\ &= m_i \dot{\boldsymbol{u}}_i + m_i \dot{\boldsymbol{\omega}}_i \times \boldsymbol{c}_i + \boldsymbol{\omega}_i \times \boldsymbol{P}_i \\ \dot{\boldsymbol{L}}_i &= m_i \dot{\boldsymbol{c}}_i \times \boldsymbol{u}_i + m_i \boldsymbol{c}_i \times \dot{\boldsymbol{u}}_i + I_i \dot{\boldsymbol{\omega}}_i + \dot{I}_i \boldsymbol{\omega}_i \\ &= m_i \boldsymbol{c}_i \times \dot{\boldsymbol{u}}_i + I_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{u}_i \times \boldsymbol{P}_i + \boldsymbol{\omega}_i \times \boldsymbol{L}_i \end{aligned} \tag{23}$$

B. FORWARD KINEMATICS

1) CALCULATION OF VELOCITY

Velocity and angular velocity can be defined recursively using the values of the parent joint by differentiating (8) as follows:

$$\begin{aligned}\dot{\boldsymbol{p}}_i &= \dot{\boldsymbol{p}}_j + \dot{R}_j \boldsymbol{p}_{ji} \\ &= \boldsymbol{v}_j + \boldsymbol{\omega}_j \times R_j \boldsymbol{p}_{ji} \\ &= \boldsymbol{v}_i \\ \dot{R}_i &= \dot{R}_j R_{ji} e^{[\boldsymbol{a}_{ii} \times] q_i} + R_j R_{ji} \dot{e}^{\boldsymbol{a}_{ii} q_i} \\ &= [\boldsymbol{\omega}_j \times] R_j R_{ji} e^{[\boldsymbol{a}_{ii} \times] q_i} + [\boldsymbol{a}_i \dot{q}_i \times] R_j R_{ji} e^{[\boldsymbol{a}_{ii} \times] q_i} \\ &= [(\boldsymbol{\omega}_j + \boldsymbol{a}_i \dot{q}_i) \times] R_i \\ &= [\boldsymbol{\omega}_i \times] R_i \end{aligned} \tag{24}$$

(24) is based on the following equation:

$$\begin{aligned} R_j R_{ji} \dot{e}^{\boldsymbol{a}_{ii} q_i} &= R_j R_{ji} \left([\boldsymbol{a}_{ii} \dot{q}_i \times] e^{[\boldsymbol{a}_{ii} \times] q_i} \right) \\ &= \left[\left(R_j R_{ji} \boldsymbol{a}_{ii} \dot{q}_i \right) \times \right] \left(R_j R_{ji} e^{[\boldsymbol{a}_{ii} \times] q_i} \right) \\ &= [\boldsymbol{a}_i \dot{q}_i \times] R_j R_{ji} e^{[\boldsymbol{a}_{ii} \times] q_i} \end{aligned} \tag{25}$$

2) CALCULATION OF ACCELERATION

Acceleration and angular acceleration can be defined recursively using the values of the parent joint by differentiating (24) as follows:

$$\begin{aligned} \dot{\boldsymbol{v}}_i &= \dot{\boldsymbol{v}}_j + \dot{\boldsymbol{\omega}}_j \times R_j \boldsymbol{p}_{ji} + \boldsymbol{\omega}_j \times \left(\boldsymbol{\omega}_j \times R_j \boldsymbol{p}_{ji} \right) \\ \dot{\boldsymbol{\omega}}_i &= \dot{\boldsymbol{\omega}}_j + \boldsymbol{\omega}_j \times \boldsymbol{a}_i \dot{q}_i + \boldsymbol{a}_i \ddot{q}_i \end{aligned} \tag{26}$$

(26) is based on the following equation:

$$\dot{\boldsymbol{a}}_i = \dot{R}_j \boldsymbol{a}_{ii} = \boldsymbol{\omega}_j \times R_j \boldsymbol{a}_{ii} = \boldsymbol{\omega}_j \times \boldsymbol{a}_i \tag{27}$$

(26) is transformed using the spatial velocity as follows:

$$\begin{aligned} \boldsymbol{u}_i &= \boldsymbol{v}_i - \boldsymbol{\omega}_i \times \boldsymbol{p}_i \\ &= \boldsymbol{v}_j + \boldsymbol{\omega}_j \times R_j \boldsymbol{p}_{ji} - \left(\boldsymbol{\omega}_j + \boldsymbol{a}_i \dot{q}_i \right) \times \left(\boldsymbol{p}_j + R_j \boldsymbol{p}_{ji} \right) \\ &= \boldsymbol{v}_j - \boldsymbol{\omega}_j \times \boldsymbol{p}_j - \boldsymbol{a}_i \dot{q}_i \times \left(\boldsymbol{p}_j + R_j \boldsymbol{p}_{ji} \right) \\ &= \boldsymbol{u}_j + \left(\boldsymbol{p}_i \times \boldsymbol{a}_i \right) \dot{q}_i \\ \boldsymbol{\omega}_i &= \boldsymbol{\omega}_j + \boldsymbol{a}_i \dot{q}_i \end{aligned} \tag{28}$$

3) ARRANGEMENT OF FORMULAS

To organize the equation, the following symbols are introduced:

$$\begin{aligned} \Delta \boldsymbol{u}_i &= \boldsymbol{p}_i \times \boldsymbol{a}_i \\ \Delta \boldsymbol{\omega}_i &= \boldsymbol{a}_i \end{aligned} \tag{29}$$

(28) can be arranged as follows:

$$\begin{aligned} \boldsymbol{u}_i &= \boldsymbol{u}_j + \Delta \boldsymbol{u}_i \dot{q}_i \qquad (30) \\ \boldsymbol{\omega}_i &= \boldsymbol{\omega}_j + \Delta \boldsymbol{\omega}_i \dot{q}_i \\ \dot{\boldsymbol{u}}_i &= \dot{\boldsymbol{u}}_j + \Delta \dot{\boldsymbol{u}}_i \dot{q}_i + \Delta \boldsymbol{u}_i \ddot{q}_i \qquad (31) \\ \dot{\boldsymbol{\omega}}_i &= \dot{\boldsymbol{\omega}}_j + \Delta \dot{\boldsymbol{\omega}}_i \dot{q}_i + \Delta \boldsymbol{\omega}_i \ddot{q}_i \end{aligned}$$

The differentials of (29) are calculated as follows:

$$\begin{aligned} \Delta \dot{\boldsymbol{u}}_i &= \dot{\boldsymbol{p}}_i \times \boldsymbol{a}_i + \boldsymbol{p}_i \times \dot{\boldsymbol{a}}_i \\ &= \left(\boldsymbol{v}_j + \boldsymbol{\omega}_j \times \left(\boldsymbol{p}_i - \boldsymbol{p}_j \right) \right) \times \boldsymbol{a}_i + \boldsymbol{p}_i \times \left(\boldsymbol{\omega}_j \times \boldsymbol{a}_i \right) \\ &= \left(\boldsymbol{u}_j + \boldsymbol{\omega}_j \times \boldsymbol{p}_i \right) \times \boldsymbol{a}_i + \boldsymbol{p}_i \times \left(\boldsymbol{\omega}_j \times \boldsymbol{a}_i \right) \\ &= \boldsymbol{u}_j \times \boldsymbol{a}_i - \boldsymbol{a}_i \times \left(\boldsymbol{\omega}_j \times \boldsymbol{p}_i \right) - \boldsymbol{p}_i \times \left(\boldsymbol{a}_i \times \boldsymbol{\omega}_j \right) \\ &= \boldsymbol{u}_j \times \boldsymbol{a}_i + \boldsymbol{\omega}_j \times \left(\boldsymbol{p}_i \times \boldsymbol{a}_i \right) \\ &= \boldsymbol{u}_j \times \Delta \boldsymbol{\omega}_i + \boldsymbol{\omega}_j \times \Delta \boldsymbol{u}_i \\ \Delta \dot{\boldsymbol{\omega}}_i &= \dot{\boldsymbol{a}}_i \\ &= \boldsymbol{\omega}_j \times \boldsymbol{a}_i \\ &= \boldsymbol{\omega}_j \times \Delta \boldsymbol{\omega}_i \end{aligned} \tag{32}$$

(32) relies on the following equation to hold for arbitrary vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ as follows:

$$\boldsymbol{a} \times (\boldsymbol{b} \times \boldsymbol{c}) + \boldsymbol{c} \times (\boldsymbol{a} \times \boldsymbol{b}) + \boldsymbol{b} \times (\boldsymbol{c} \times \boldsymbol{a}) = 0 \tag{33}$$

C. FORWARD DYNAMICS

1) CALCULATION OF JOINT TORQUE

The joint torque is equal to the moment around the axis of rotation as follows:

$$\begin{aligned}\tau_i &= \boldsymbol{a}_i^T(\boldsymbol{p}_i \times \boldsymbol{f}_i) + \boldsymbol{a}_i^T \boldsymbol{n}_i \\ &= \Delta \boldsymbol{u}_i^T \boldsymbol{f}_i + \Delta \boldsymbol{\omega}_i^T \boldsymbol{n}_i \end{aligned} \tag{34}$$

(34) relies on the following equation to hold for arbitrary vectors $\boldsymbol{a}, \boldsymbol{b}$ and $\boldsymbol{c}$ as follows:

$$\boldsymbol{a}^T(\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{c}^T(\boldsymbol{a} \times \boldsymbol{b}) = \boldsymbol{b}^T(\boldsymbol{c} \times \boldsymbol{a}) \tag{35}$$

2) ARRANGEMENT OF EQUATIONS OF MOTION

To find the forward dynamics calculation algorithm, (9), (30) and (34) are arranged as follows:

$$\begin{aligned}\boldsymbol{y}_i - \sum\nolimits_k \boldsymbol{y}_k &= M_i \ddot{\boldsymbol{x}}_i + \boldsymbol{z}_i \\ \forall_k \ddot{\boldsymbol{x}}_k &= \ddot{\boldsymbol{x}}_i + \ddot{\boldsymbol{s}}_k \dot{q}_k + \dot{\boldsymbol{s}}_k \ddot{q}_k \\ \tau_i &= \dot{\boldsymbol{s}}_i^T \boldsymbol{y}_i \end{aligned} \tag{36}$$

Here are the newly defined symbols:

$$\begin{aligned}\ddot{\boldsymbol{x}}_i &= \begin{pmatrix} \dot{\boldsymbol{u}}_i \\ \dot{\boldsymbol{\omega}}_i \end{pmatrix},\ \boldsymbol{y}_i = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{n}_i \end{pmatrix},\ \dot{\boldsymbol{s}}_i = \begin{pmatrix} \Delta \boldsymbol{u}_i \\ \Delta \boldsymbol{\omega}_i \end{pmatrix} \\ \boldsymbol{z}_i &= \begin{pmatrix} -m_i \boldsymbol{g} - \boldsymbol{f}_i^e + \boldsymbol{\omega}_i \times \boldsymbol{P}_i \\ -m_i \boldsymbol{c}_i \times \boldsymbol{g} - \boldsymbol{n}_i^e + \boldsymbol{u}_i \times \boldsymbol{P}_i + \boldsymbol{\omega}_i \times \boldsymbol{L}_i \end{pmatrix} \\ M_i &= \begin{pmatrix} m_i E & -m_i[\boldsymbol{c}_i \times] \\ m_i[\boldsymbol{c}_i \times] & I_i \end{pmatrix} \end{aligned} \tag{37}$$

3) CALCULATION OF THE FORWARD DYNAMICS

Featherstone [27] found that forward dynamics calculations can be performed efficiently by formulating the following equation:

$$\boldsymbol{y}_i = M_i^A \ddot{\boldsymbol{x}}_i + \boldsymbol{z}_i^A \tag{38}$$

(38) can be calculated for τ_k as follows:

$$\begin{aligned}\tau_k &= \dot{\boldsymbol{s}}_k^T \boldsymbol{y}_k \\ &= \dot{\boldsymbol{s}}_k^T M_k^A \ddot{\boldsymbol{x}}_k + \dot{\boldsymbol{s}}_k^T \boldsymbol{z}_k^A \\ &= \dot{\boldsymbol{s}}_k^T M_k^A \ddot{\boldsymbol{x}}_i + \dot{\boldsymbol{s}}_k^T M_k^A \ddot{\boldsymbol{s}}_k \dot{q}_k + \dot{\boldsymbol{s}}_k^T I_k^A \dot{\boldsymbol{s}}_k \ddot{q}_k + \dot{\boldsymbol{s}}_k^T \boldsymbol{z}_k^A \end{aligned} \tag{39}$$

(39) can be solved for $\ddot{q}_k$ as follows:

$$\begin{aligned}\ddot{q}_k &= \left(\tau_k - \boldsymbol{\zeta}_k^T \ddot{\boldsymbol{x}}_i - \eta_k\right) \iota_k \\ \boldsymbol{\zeta}_k^T &= \dot{\boldsymbol{s}}_k^T M_k^A,\ \eta_k = \boldsymbol{\zeta}_k^T \ddot{\boldsymbol{s}}_k \dot{q}_k + \dot{\boldsymbol{s}}_k^T \boldsymbol{z}_k^A,\ \iota_k = \left(\boldsymbol{\zeta}_k^T \dot{\boldsymbol{s}}_k\right)^{-1} \end{aligned} \tag{40}$$

From the above, the recursive expression for M_i^A and $\boldsymbol{z}_i^A$ is calculated as follows:

$$\begin{aligned}\boldsymbol{y}_i &= M_i^A \ddot{\boldsymbol{x}}_i + \boldsymbol{z}_i^A \\ &= \boldsymbol{y}_i - \sum\nolimits_k \boldsymbol{y}_k + \sum\nolimits_k \boldsymbol{y}_k \\ &= M_i \ddot{\boldsymbol{x}}_i + \sum\nolimits_k M_k^A \ddot{\boldsymbol{x}}_k + \boldsymbol{z}_i + \sum\nolimits_k \boldsymbol{z}_k^A \\ &= M_i \ddot{\boldsymbol{x}}_i + \sum\nolimits_k M_k^A \left(\ddot{\boldsymbol{x}}_i + \ddot{\boldsymbol{s}}_k \dot{q}_k + \dot{\boldsymbol{s}}_k \ddot{q}_k\right) + \boldsymbol{z}_i + \sum\nolimits_k \boldsymbol{z}_k^A \\ &= \left(M_i + \sum\nolimits_k M_k^A\right) \ddot{\boldsymbol{x}}_i + \boldsymbol{z}_i + \sum\nolimits_k \left(\boldsymbol{z}_k^A + M_k^A \ddot{\boldsymbol{s}}_k \dot{q}_k\right) \\ &\quad + \sum\nolimits_k M_k^A \dot{\boldsymbol{s}}_k \left(\tau_k - \boldsymbol{\zeta}_k^T \ddot{\boldsymbol{x}}_i - \eta_k\right) \iota_k \\ &= \left(M_i + \sum\nolimits_k \left(M_k^A - M_k^A \dot{\boldsymbol{s}}_k \boldsymbol{\zeta}_k^T \iota_k\right)\right) \ddot{\boldsymbol{x}}_i \\ &\quad + \boldsymbol{z}_i + \sum\nolimits_k \left(\boldsymbol{z}_k^A + M_k^A \ddot{\boldsymbol{s}}_k \dot{q}_k + M_k^A \dot{\boldsymbol{s}}_k \left(\tau_k - \eta_k\right) \iota_k\right) \end{aligned} \tag{41}$$

Therefore, the following holds:

$$\begin{aligned}M_i^A &= M_i + \sum\nolimits_k \left(M_k^A - M_k^A \dot{\boldsymbol{s}}_k \boldsymbol{\zeta}_k^T \iota_k\right) \\ \boldsymbol{z}_i^A &= \boldsymbol{z}_i + \sum\nolimits_k \left(\boldsymbol{z}_k^A + M_k^A \ddot{\boldsymbol{s}}_k \dot{q}_k + M_k^A \dot{\boldsymbol{s}}_k \left(\tau_k - \eta_k\right) \iota_k\right) \end{aligned} \tag{42}$$

Forward dynamics calculations can be computed in $\mathcal{O}(N)$ by solving (42) and (40) sequentially.

REFERENCES

[1] H. Charles, J. R. Beddington, I. R. Crute, L. Haddad, D. P. Lawrence, J. Muir, J. Pretty, S. Robinson, S. M. Thomas, and C. Toulmin, "Food security: The challenge of feeding 9 billion people," *Science*, vol. 327, no. 5967, pp. 812–818, Jan. 2010.

[2] R. Gebbers and V. I. Adamchuk, "Precision agriculture and food security," *Science*, vol. 327, no. 5967, pp. 828–831, Feb. 2010.

[3] (Accessed: Feb. 20, 2024). *Agricultural Labor Force Statistics*. Ministry of Agriculture, Forestry and Fisheries, Japan. [Online]. Available: https://www.maff.go.jp/j/tokei/sihyo/data/08.html

[4] J. Huang, J. L. Gómez-Dans, H. Huang, H. Ma, Q. Wu, P. E. Lewis, S. Liang, Z. Chen, J.-H. Xue, Y. Wu, F. Zhao, J. Wang, and X. Xie, "Assimilation of remote sensing into crop growth models: Current status and perspectives," *Agricult. Forest Meteorol.*, vols. 276–277, Oct. 2019, Art. no. 107609.

[5] R. Ramin Shamshiri, I. A. Hameed, L. Pitonakova, C. Weltzien, S. K. Balasundram, I. J. Yule, T. E. Grift, and G. Chowdhary, "Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison," *Int. J. Agricult. Biol. Eng.*, vol. 11, no. 4, pp. 12–20, 2018.

[6] L. Emmi, L. Paredes-Madrid, A. Ribeiro, G. Pajares, and P. Gonzalez-de-Santos, "Fleets of robots for precision agriculture: A simulation environment," *Ind. Robot: Int. J.*, vol. 40, no. 1, pp. 41–58, Jan. 2013.

[7] R. R. Shamshiri, I. A. Hameed, M. Karkee, and C. Weltzien, "Robotic harvesting of fruiting vegetables: A simulation approach in V-REP, ROS and MATLAB," in *Proceedings in Automation in Agriculture-Securing Food Supplies for Future Generations*, vol. 126. IntechOpen, Mar. 2018, pp. 81–105.

[8] (Accessed: Feb. 20, 2024). *Agisoft Metashape(R)*. [Online]. Available: https://www.agisoft.com/

[9] M. J. Westoby, J. Brasington, N. F. Glasser, M. J. Hambrey, and J. M. Reynolds, "'Structure-from-motion' photogrammetry: A low-cost, effective tool for geoscience applications," *Geomorphology*, vol. 179, pp. 300–314, Dec. 2012.

[10] Y. Nagasaka, H. Saito, K. Tamaki, M. Seki, K. Kobayashi, and K. Taniwaki, "An autonomous rice transplanter guided by global positioning system and inertial measurement unit," *J. Field Robot.*, vol. 26, nos. 6–7, pp. 537–548, Jun. 2009.

[11] N. Noguchi, M. Kise, K. Ishii, and H. Terao, "Field automation using robot tractor," in *Automation Technology for Off-Road Equipment Proceedings of the 2002 Conference*. St. Joseph, MI, USA: American Society of Agricultural and Biological Engineers, 2002, p. 239.

[12] (Accessed: Feb. 20, 2024). *Hands Free Farm*. [Online]. Available: https://www.handsfree.farm/

[13] J. Underwood, A. Wendel, B. Schofield, L. Mcmurray, and R. Kimber, "Efficient in-field plant phenomics for row-crops with an autonomous ground vehicle," *J. Field Robot.*, vol. 34, no. 6, pp. 1061–1083, Sep. 2017.

[14] O. Bawden, J. Kulk, R. Russell, C. McCool, A. English, F. Dayoub, C. Lehnert, and T. Perez, "Robot for weed species plant-specific management," *J. Field Robot.*, vol. 34, no. 6, pp. 1179–1199, Sep. 2017.

[15] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1045–1052, Sep. 2017.
[16] J. Kim, S. Kim, C. Ju, and H. I. Son, "Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications," *IEEE Access*, vol. 7, pp. 105100–105115, 2019.
[17] M. Tanaka, A. Hama, Y. Tsurusaki, and Y. Shibato, "Methods of aerial photography using drone and image analyses for evaluation of cabbage growth at individual level," *J. Remote Sens. Soc. Jpn.*, vol. 41, pp. 375–385, Jun. 2021.
[18] A. K. Mortensen, A. Bender, B. Whelan, M. M. Barbour, S. Sukkarieh, H. Karstoft, and R. Gislum, "Segmentation of lettuce in coloured 3D point clouds for fresh weight estimation," *Comput. Electron. Agricult.*, vol. 154, pp. 373–381, Nov. 2018.
[19] Y. Nakano, S. Noda, Y. Miyake, M. Kogoshi, F. Sato, and W. Iijima, "Applicability of variable-rate nitrogen top dressing based on measurement of the within-field variability of soil nutrients for cabbage production," *Horticulturae*, vol. 9, no. 4, p. 506, Apr. 2023.
[20] S. Noda, Y. Miyake, Y. Nakano, M. Kogoshi, W. Iijima, and J. Nakagawa, "A mobile laboratory robot for various and precise measurements of crops and soil in agricultural fields: Development and pilot study," *Agriculture*, vol. 13, no. 7, p. 1419, Jul. 2023.
[21] D. SepúLveda, R. Fernández, E. Navas, M. Armada, and P. González-De-Santos, "Robotic aubergine harvesting using dual-arm manipulation," *IEEE Access*, vol. 8, pp. 121889–121904, 2020.
[22] Y. Onishi, T. Yoshida, H. Kurita, T. Fukao, H. Arihara, and A. Iwai, "An automated fruit harvesting robot by using deep learning," *Robomech J.*, vol. 6, no. 1, pp. 1–8, 2019.
[23] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.
[24] S. Nakaoka, "Choreonoid: Extensible virtual robot environment built on an integrated GUI framework," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Dec. 2012, pp. 79–85.
[25] S. Noda, F. Sugai, K. Kojima, K.-N.-K. Nguyen, Y. Kakiuchi, K. Okada, and M. Inaba, "Semi-passive walk and active walk by one bipedal robot: Mechanism, control and parameter identification," *Int. J. Humanoid Robot.*, vol. 17, no. 2, Apr. 2020, Art. no. 2050012.
[26] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Sep. 2004, pp. 2149–2154.
[27] R. Featherstone, *Rigid Body Dynamics Algorithms*. Secaucus, NJ, USA: Springer-Verlag, 2007.
[28] T. Matsui and M. Inaba, "EusLisp: An object-based implementation of lisp," *J. Inf. Process.*, vol. 13, no. 3, pp. 327–338, Jul. 1991.
[29] (Accessed: Apr. 12, 2024). *ZED-F9P Module: U-blox F9 High Precision GNSS Module*. U-Blox Holding AG. [Online]. Available: https://www.u-blox.com/en/product/zed-f9p-module
[30] PROJ Contributors. *PROJ Coordinate Transformation Software Library*. Springer, 2024. [Online]. Available: https://proj.org/
[31] J. Xu, X. Wang, Z. Zhang, and W. Wu, "Discrete element modeling and simulation of soybean seed using multi-spheres and super-ellipsoids," *IEEE Access*, vol. 8, pp. 222672–222683, 2020.
[32] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.

SHINTARO NODA received the B.E., M.S., and Ph.D. degrees in information science and technology from The University of Tokyo, Japan, in 2013, 2015, and 2018. He was a Project Research Associate with The University of Tokyo, from 2018 to 2021. Currently, he is a Researcher with the Research Center for Agricultural Robotics, National Agriculture and Food Research Organization (NARO), Japan. His research interests include agricultural automation and robotics.

MASAYUKI KOGOSHI received the B.S. and M.S. degrees in agricultural science from Chiba University, Japan, in 2007 and 2009, respectively, and the M.S. and Ph.D. degrees in information science from the Suwa University of Science, Japan, in 2021 and 2024, respectively. He is currently a Researcher with the Research Center for Agricultural Robotics, National Agriculture and Food Research Organization (NARO), Japan. His research interest includes drones and data-driven soil management technology.

WATARU IIJIMA received the B.E., M.S., and Ph.D. degrees in agricultural science from Hokkaido University, in 1994, 1996, and 2007, respectively. He is a Leader of the Field Robotics Unit, Research Center for Agricultural Robotics, National Agricultural Research Organization (NARO). From 2015 to 2017, he was involved in the planning of a national project for research and development on smart agriculture with the Secretariat of Agriculture, Forestry and Fisheries Research Council.