2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

Torque-Thrust Integrated Inverse Dynamics for Rotor Distributed Manipulator toward Dynamic Deformation

Kazuki Sugihara[1] and Kei Okada[1]

***Abstract*— In recent years, manipulation tasks in high place by aerial robots have been realized. In particular, rotor distributed manipulators equipped with rotors on each link of articulated structure have attracted attention for their ability to perform high DoF tasks through whole-body deformation. However, in many conventional rotor distributed robots, joint control and thrust control were separated for model simplification, or the dynamics of joint motion were ignored and robot body was approximated as a rigid multirotor at each control cycle. As a result, they are mainly applied to tasks comprising static motion, and its application to tasks requiring dynamic deformation is limited. Therefore, in this work, we aim to enable dynamic deformation by a rotor distributed manipulator. We clarify the whole-body dynamics model of a rotor distributed manipulator that integrate joint torque and thrust, and develop its control system. In the proposed inverse dynamics method, we formulate a problem based on quadratic programming to minimize the exerted thrust and joint torque while achieving the desired whole-body motion. Then, we implement a control system for a rotor distributed manipulator incorporating the proposed inverse dynamics method. We verify the effectiveness of the proposed method through trajectory generation of actuator inputs and dynamic motion simulation. To the best of our knowledge, this is the first time to establish a control system for dynamic deformation by integrating joint torque and thrust for rotor distributed robots.**

I. Introduction

In recent years, aerial robot have been used not only for tasks that utilize their three-dimensional mobility, such as exploration [1] and guidance [2], but also for tasks involving environmental contact, such as valve opening [3] and object transportation [4], [5]. There are two types of aerial robot configurations for aerial manipulation: rotor centralized type and rotor distributed type. Rotor centralized configurations attach manipulators to a conventional multirotor [4], [6], while rotor distributed configurations are deployed rotors on each links of the manipulator's skeleton, allowing for whole-body deformation [7], [8], [9]. Rotor distributed configurations have the advantage of being able to easily keep the CoG with in the support area of the rotors even when the body deforms, and to distribute and reduce the load on each joint by using thrust, compared to rotor centralized configurations.

However, many conventional rotor distributed robots simplified the nonlinear dynamics model by separating joint control and thrust control, or approximated the robot body as a rigid multirotor at each moment by ignoring the dynamics of joint motion. In such control system, kinematics based joint torque compensation by thrust is only possible, so there are limitations in performing tasks which require dynamic deformation. Therefore, to address this problem, this work proposes a control system for rotor distributed manipulators that integrates the dynamics of joint motion and exerted thrust. The control inputs, joint torque and thrust, are optimized simultaneously to achieve the desired motion. Since rotor distributed manipulators have redundant control DoF for thrust compared to the number of joints, this redundancy can be utilized to minimize the required joint torque. We formulate this inverse dynamics problem based on quadratic programming and establish a control system that incorporates it. And we verify the effectiveness of the proposed inverse dynamics by trajectory generation of joint torque and thrust, and further realize dynamic deformation using a dynamics simulator. This dynamic deformation is expected to be applied to practical applications such as tap testing in infrastructure inspection.

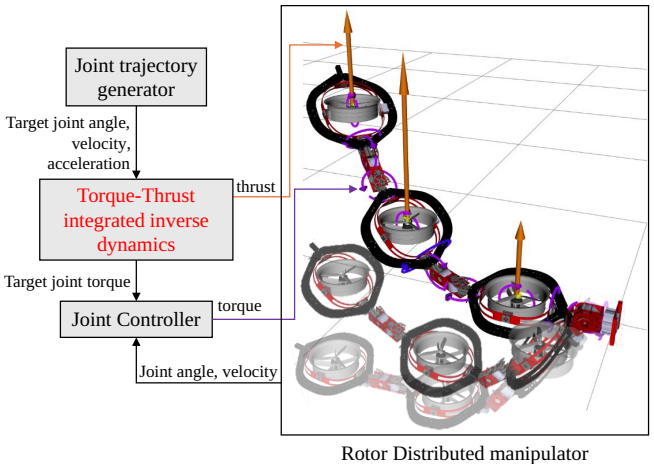


Fig. 1: Dynamic deformation by rotor-distributed manipulator integrating torque and thrust.

II. Related works and our contribution

A. Related works

In [10], an optimal control method to achieve fast deformation by a two-dimensional transformable multilink multirotor was proposed. However, the thrust does not affect joint motion, so the actuators in whole-body was not integrated in control. On the other hand, there is a rotor distributed humanoid robot that fly by controlling joint with fixed thrust [12]. It has stability issues due to low responsiveness of the

This work was supported by JSPS KAKENHI Grant Number JP25KJ0776.

[1]Kazuki Sugihara and Kei Okada are with Department of Mechano-Informatics, Graduate School of Information Science and Technology, The University of Tokyo. sugihara@jsk.imi.i.u-tokyo.ac.jp

	MPC for two dimentional transformable multilink multirotor [10]	LASDRA [7]: Rotor equipped links are connected by passive joint	DRAGON [11]: Three dimentional transformable multilink multirotor	Ours
Consider Joint Dynamics	✓: State equation of MPC consider joint velocity and acceleration.	✓: Acceleration-level control is applied to each link	⊘: Ignore joint velocity and acceleration in modeling	✓
Torque-Thrust Integration	⊘: Control for joint and thrust are separated	⊘: Only consider external actuation by distributed rotors	✓: Use thrust to decrease joint load	✓

TABLE I: Comparison of control strategies for rotor-distributed robots. In previous works, these robots ignored joint dynamics or separated control of joint and thrust. We incorporate dynamics with integrated torque and thrust control.

joints used for feedback control. In LASDRA [7] and Hiryu-II [13], thrust modules are connected by passive joints to realize deformation using only thrust, however, the control system does not consider the internal forces which can be generated on joint by actuators. In addition, a control method that uses thrust to reduce the load on position-controlled joints has been proposed [11], [14]. However, it assumes quasi-static deformation by ignoring the influence of joint angular velocity and acceleration, so dynamic deformation is difficult with this method.

Thus, conventional control of rotor distributed robots have the problem of separating joint control and thrust control, or assuming quasi-static deformation, making it difficult to consider dynamic deformation with integrated whole-body actuation. Therefore, in this work, we propose a whole-body inverse dynamics which integrates the exerted thrust and torque considering dynamics of joints. The comparison of these previous works and our work is shown in Table I.

B. Our contribution

Key contributions of this work are summarized as follows:

1) We clarify the dynamics model of rotor distributed manipulator that integrates thrust and joint torque.
2) We propose a torque-thrust integrated inverse dynamics method to achieve desired dynamic motion, and show that dynamic motion trajectory can be generated.
3) We establish a control system for rotor distributed manipulator, and demonstrate that dynamic deformation is possible through simulation.

This work is the first to establish a control system for dynamic deformation by integrating joint torque and thrust for rotor distributed robots.

III. Dynamics model of rotor distributed manipulator

In this section, we clarify the dynamics model of rotor distributed manipulator, and propose a torque-thrust integrated inverse dynamics method.

A. Model of rotor with rotating propeller

In this work, we use rotating propellers for thrusters, and assume that thrust along to the rotational axis and counter torque proportional to the thrust around the rotational axis are generated. Moreover, we assume that rotational axis of rotor is corresponding to z-axis of rotor frame. So the wrench generated in the frame of the i-th rotor is as follows,

$$\begin{bmatrix} \boldsymbol{b}_3 \\ \sigma_i \boldsymbol{b}_3 \end{bmatrix} \lambda_i. \tag{1}$$

In (1), $\boldsymbol{b}_3 = \begin{bmatrix} 0, 0, 1 \end{bmatrix}^T$. $\lambda_i \in \mathbb{R}$ and $\sigma_i \in \mathbb{R}$ are exerted thrust and counter torque coefficient (signed) of i-th rotor, respectively. We assume that the direction of the rotor's rotation is unidirectional, and thrust can only be generated along the positive direction of the rotational axis, so $\lambda_i > 0$.

B. Dynamics model of rotor-distributed robot

To verify whether dynamic joint deformation is possible based on a whole-body control method that integrates torque and thrust, we simplify the model by considering a fixed-base system with the root link fixed at the origin of the inertial frame. In this work, we consider the dynamics model of a rotor distributed manipulator as shown in Fig. 2. The equations of motion for a multilink system can be expressed by considering the wrench generated by the exerted thrust in addition to the joint torque as follows,

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} + \sum_{i=1}^{N_r} J_{\lambda_i}^T \begin{bmatrix} \boldsymbol{b}_3 \\ \sigma_i \boldsymbol{b}_3 \end{bmatrix} \lambda_i. \tag{2}$$

In (2), $\boldsymbol{q} \in \mathbb{R}^{N_q}$ is configuration vector and N_q is the number of the joints. $M(\boldsymbol{q})$ is the inertia matrix, $C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is the Coriolis and centrifugal force, $\boldsymbol{g}(\boldsymbol{q})$ is the gravity, $\boldsymbol{\tau}$ is the joint torque, and J_{λ_i} is the jacobian of the i-th rotor frame. N_r is the number of rotors.

C. Torque-Thrust integrated inverse dynamics

Inverse dynamics is an algorithm that calculate target generalized forces from generalized coordinates, generalized velocities, and generalized accelerations. In rotor distributed manipulator, exerted thrust is also an input to the actuators in addition to joint torque, so we calculate them simultaneously. Therefore, we propose the following torque-thrust integrated inverse dynamics.

The target variables are the target joint torque and exerted thrust, so the number of variables is $N_q + N_r$. And there are equality constraints as the motion equation (2) that expresses the desired motion. Therefore, there are N_q equality constraints and $N_q + N_r$ decision variables, so there are redundant DoF of the number of rotors. Although the pseudo-inverse matrix can be used to calculate the solution that satisfy (2), these solution does not necessarily satisfy the upper and lower limits defined by each actuators. Hence, we use quadratic programming to optimize all actuator inputs considering the inequality constraints of the actuator inputs in addition to dynamics constraints. We formulate the optimization problem to minimize exerted torque and thrust as follows,

$$\text{minimize} \qquad \boldsymbol{\tau}^T W_1 \boldsymbol{\tau} + \boldsymbol{\lambda}^T W_2 \boldsymbol{\lambda}, \tag{3}$$

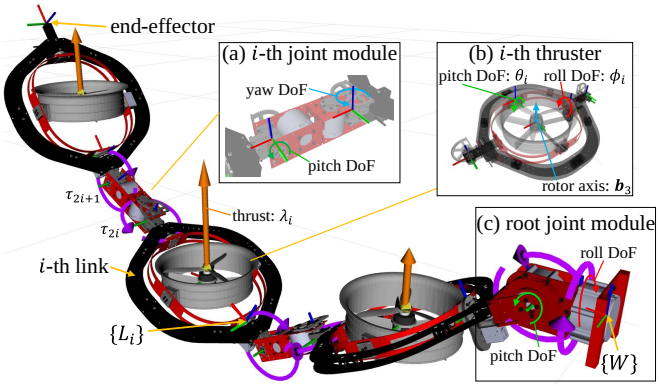


Fig. 2: Dynamics model of rotor-distributed manipulator in this work. (a) link joint module with 2 DoF. (b) thruster with 2 DoF vectoring freedom. (c) root joint module which is fixed to world frame $\{W\}$.

$$\text{subject to} \quad M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} + \sum_{i=1}^{N_r} J_{\lambda_i}^T \begin{bmatrix} \boldsymbol{b}_3 \\ \sigma_i \boldsymbol{b}_3 \end{bmatrix} \lambda_i, \tag{4}$$

$$\tau_{min,i} < \tau_i < \tau_{max,i} \quad (i = 1, \ldots, N_q), \tag{5}$$

$$\lambda_{min,i} < \lambda_i < \lambda_{max,i} \quad (i = 1, \ldots, N_r). \tag{6}$$

In (3), $\boldsymbol{\lambda} = \left[\lambda_1, \cdots, \lambda_{N_r}\right]^T$ is thrust vector. W_1 and W_2 are diagonal matrices representing the weights for the optimization of joint torque and thrust, respectively. The larger the value of this matrix, the smaller the optimal solution for the corresponding joint torque and thrust is obtained. Also, $\tau_{min,i}$, $\tau_{max,i}$, $\lambda_{min,i}$, and $\lambda_{max,i}$ in (5) and (6) are the lower and upper limits of i-th joint torque and thrust, respectively.

IV. Joints control for manipulator

In this section, we describe the motion control framework that incorporates the proposed inverse dynamics method in Sec.III. The control framework is shown in Fig. 3, which consists of joint angle planning, calculation of nominal actuator inputs by inverse dynamics, and joint PD control.

A. Joint trajectory generator

The robot model used in this work has two types of joints. One is skeleton joint that can control the pose of the end-effector, and the other is thrust vectoring joint that only control the direction of thruster. We use different planner for these joints.

1) Target Trajectory generation for skelton joint: In this work, we perform a task of commanding the three dimensional position, velocity and acceleration of the end-effector. By solving the inverse kinematics at each control cycle, we obtain the target joint angles $\boldsymbol{q}_{des}$. Target joint angular velocities $\dot{\boldsymbol{q}}_{des}$ are calculated from the end-effector's target velocity and actual joint angles in each control cycle. For feedback control using joint torque and thrust, target angular acceleration are calculated as follows,

$$\ddot{\boldsymbol{q}}_{des} = \ddot{\boldsymbol{q}}_{ff} + K_p(\boldsymbol{q}_{des} - \boldsymbol{q}) + K_d(\dot{\boldsymbol{q}}_{des} - \dot{\boldsymbol{q}}). \tag{7}$$

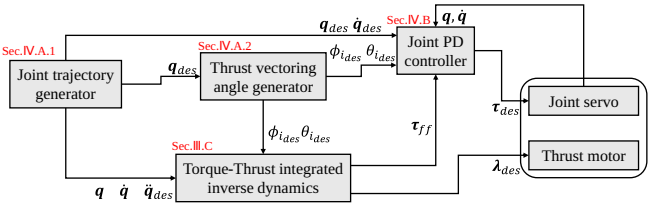


Fig. 3: Control framework for rotor-distributed manipulator including proposed torque-thrust integrated inverse dynamics.

In (7), $\ddot{\boldsymbol{q}}_{ff}$ is feed-forward joint acclerations calculated from target end-effector acceleration and current robot state. Target joint accelerations $\ddot{\boldsymbol{q}}_{des}$ is calculated by PD feedback control with gain matrices K_p and K_d. $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}_{des}$ are used for inverse dynamics to obtain target thrust and joint torque.

2) Target angle formulation for thrust vectoring joint: The robot model used in this work has a 2 DoF thrust vectoring mechanism, as shown in Fig. 2(b). The i-th rotor can rotate around the parent link's x axis by ϕ_i and in the direction orthogonal to it by θ_i. This allows the thrust to be directed in any direction.

These joint angles do not affect the end-effector's task space, but it is important to orient the thrust direction in a way that makes it easier to move the joints. Therefore, in this work, we set the thrust direction to be aligned with the gravity direction so that it can easily counteract the influence of gravity. The target thrust vectoring angles are calculated from the parent link's pose as follows,

$$\phi_i = \operatorname{atan}\left(\frac{{}^{\{W\}}_{\{L_i\}}R(3,2)}{{}^{\{W\}}_{\{L_i\}}R(3,3)}\right), \qquad \theta_i = \operatorname{asin}\left(-{}^{\{W\}}_{\{L_i\}}R(3,1)\right). \tag{8}$$

In (8), ${}^{\{W\}}_{\{L_i\}}R$ is the rotation matrix of the parent link's pose in the inertial frame, and ${}^{\{W\}}_{\{L_i\}}R(i,j)$ is the (i,j)-th element of this matrix. We calculate these target thrust vectoring angles at each control cycle from the target joint angles, and input them to the joint controller.

B. Joint torque controller

In this work, each joint is torque controlled. For skeleton joints, in addition to the target torque obtained by inverse dynamics, PD control term is added to command joint torque. On the other hand, thrust vectoring joints are controlled by joint torque determined by only PD beedback control.

V. Verification with trajectory generation and simulation

In this section, we apply the proposed torque-thrust integrated inverse dynamics and control framework to a rotor distributed manipulator and verify dynamic deformation. First, we describe the robot model and software system. Then, we verify whether actuator inputs for dynamic deformation can be obtained by trajectory generation, and perform the motion in a dynamics simulator.

TABLE II: Tables of specifications of robot for simulation.

Physical quantity

Attribute	Value
mass of root joint module	2.0 kg
mass of link joint module	0.78 kg
mass of thruster integrated link	1.1 kg
link length	0.5 m
total length	1.9 m
propeller size	9 inch

Actuator spec

Attribute	Value
root joint velocity	18.8 rad/s (rated)
root joint torque	20 Nm (rated) / 60 Nm (peak)
link joint velocity	6.9 m/s (rated)
link joint torque	20 Nm (rated) / 56 Nm (peak)
max thrust	38 N

Actuators

Attribute	Value
root joint servo	robstride03
link joint servo	robstride00 (x4 decelerated)
thrust motor	T-Motor AT3520 KV720
thrust vectoring servo	Dynamixel XL430-W250

TABLE III: Weights of optimization for inverse dynamics

case	A	B	C	D
W_1	$1.0E_{N_q}$	$1.0E_{N_q}$	$1.0E_{N_q}$	$1.0E_{N_q}$
W_2	$10.0E_{N_r}$	$1.0E_{N_r}$	$0.1E_{N_r}$	$0.01E_{N_r}$

A. System architecture

1) Robot model: The robot model used for trajectory generation and simulation in this work is shown in Fig. 2. The skeleton joint modules are shown in Fig. 2(a) and Fig. 2(c), both of them have 2 DoF. Three links shown in Fig. 2(b) are connected in series by these joint modules. Each link is equipped with a thruster with 2 DoF thrust vectoring apparatus as shown in Fig. 2(b). These are designed using actual actuators, and have the specifications shown in Table II. Except for the root joint module, the joints are designed to be able to exert a torque of 20 Nm at rated load, while also being able to achieve a speed of 6 rad/s or more by additional reducer.

2) Software system: In this work, we build a software system for simulation using ROS [15] on a Thinkpad T15g (Intel Core i7-10850@2.7 GHz, RAM: 32 GB), and use Gazebo [16] as the dynamics simulator. For the torque control of the joints described in Sec.IV-B, we implemented a controller which allows feed-forward torque commands based on the `JointPositionController` of ros_control [17]. For the linear quadratic programming solver to solve the inverse dynamics described in Sec.III-C, we used OSQP [18]. The quadratic programming problem shown in (3)–(6) can be solved in about 50 µs for the robot model used in this work. The Pinocchio library [19] is used to calculate physical quantities such as the inertia matrix and jacobians of robot. Pinocchio is also used to implement the joint motion generator described in Sec.IV-A. This integrated simulation system is depicted in Fig. 4. This system is publicly available at `https://github.com/sugikazu75/jsk_dynarm_robot/tree/SII2026`.

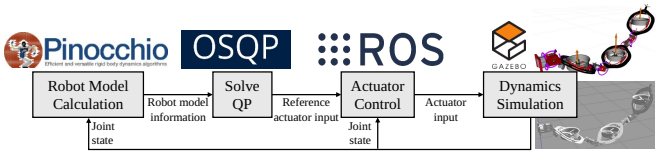
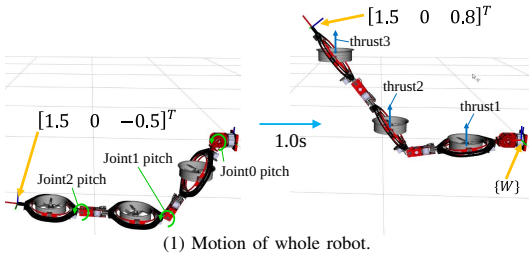
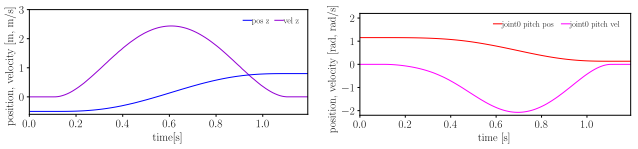


Fig. 4: Integrated software system for dynamics simulation.

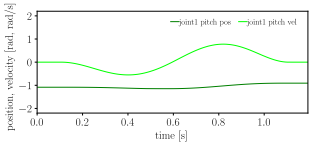


(1) Motion of whole robot.

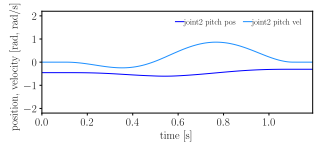


(2) End-effector z coordinate and velocity in world frame.

(3) Position and velocity of joint0_pitch joint



(4) Position and velocity of joint1_pitch joint



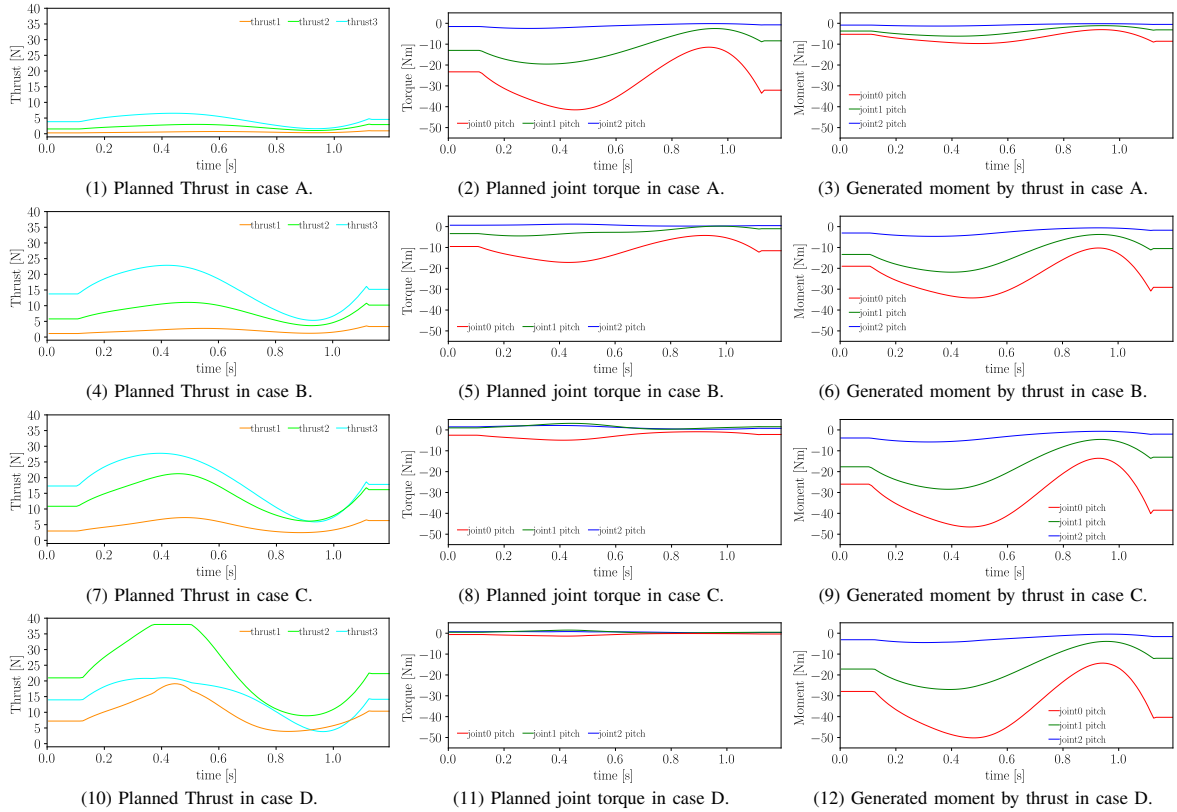
(5) Position and velocity of joint2_pitch joint

Fig. 5: (1): Planned dynamic motion. (2): Trajectory of position and velocity of end-effector in z direction. (3)–(5): Trajectory of joint position and velocity of which contribute to this motion.

B. Trajectory generation for dynamic deformation

1) Setting: We verify whether the target torque and thrust can be generated to achieve the dynamic motion shown in Fig. 5(1). This is a trajecotry that move the end-effector from $\begin{bmatrix}1.5 & 0 & -0.5\end{bmatrix}^T$ to $\begin{bmatrix}1.5 & 0 & 0.8\end{bmatrix}^T$ in the inertial frame $\{W\}$ shown in Fig. 5(1), along the minimum jerk trajectory in 1.0 s. The target position and velocity of end-effector in z direction are shown in Fig. 5(2). In Fig. 5(3)–Fig. 5(5), target angle and velocities of each joint that contribute this motion are plotted. We compare the combination of weight matrices in (3) for this deformation with four cases shown in Table III. E_* is $*$-dimensional identity matrix. We assume that the weight determines whether to use more torque or thrust to achieve the desired motion.

2) Result: In all cases A to D, we were able to obtain the nominal actuator inputs required to achieve the planned end-effector trajectory by the proposed inverse dynamics. We were able to plan a trajectory with a maximum end-effector velocity of 2.5 m/s and a maximum joint angle velocity of 2.1 rad/s. In Fig. 6, we show the target thrust, target joint torque, and moment generated at each joint by the thrust (the second term of the right-hand side of (4)) for each case.



(1) Planned Thrust in case A. (2) Planned joint torque in case A. (3) Generated moment by thrust in case A.

(4) Planned Thrust in case B. (5) Planned joint torque in case B. (6) Generated moment by thrust in case B.

(7) Planned Thrust in case C. (8) Planned joint torque in case C. (9) Generated moment by thrust in case C.

(10) Planned Thrust in case D. (11) Planned joint torque in case D. (12) Generated moment by thrust in case D.

Fig. 6: Plots related to trajectory generation in Fig. 5. Left column: Planned thrust for each case. Middle column: Planned joint torque for each case. Right column: Moment generated at each joint by thrust.

It can be seen that the larger the weight in the optimization for thrust, the smaller thrust and the larger joint torque were obtained. In case A, where the weight for thrust was largest, the maximum thrust was about 5 N and the target torque at the joints was about 40 Nm. On the other hand, in case D, where the weight for thrust was smallest, the planned maximum joint torque was only about 2 Nm, and the moment generated by the thrust at the joints was more than 40 Nm. In this case, target thrust for second rotor was saturated. Also, when comparing the planned thrust, it can be seen that in cases A to C, the thrust of the end-effector side rotors is larger. This is because the thrust of the end-effector side rotors can contribute to the motion of all joints, and its moment arm is longer than that of other rotors. On the other hand, in case D, where the weight for thrust was too small, the thrust of thrust2, which is the central link, became larger. This is because the thrust of the end-effector side rotors was suppressed to reduce the joint torque, resulting in a larger thrust at the central link.

From the above, with appropriate optimization weights, torque required at the joints can be made smaller than the rated torque by utilizing thrust. This is expected to reduce the required specifications for the joints, allowing the entire system to be made lighter and enabling dynamic deformation in the air. It has also been clarified that depending on the optimization weights, the thrust may generate joint load and energy may be wasted.

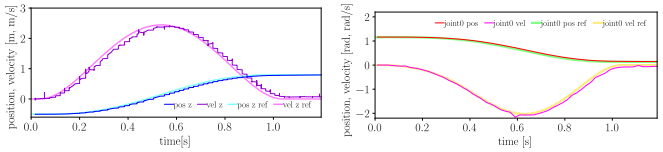
C. Dynamics simulation

1) Dynamic deformation: We verify whether the dynamic deformation motion can be actually performed in a dynamics simulator with the trajectory generated in Sec.V-B. Here, the weight matrix for the optimization in (3) was set to case C in Table III. The thrust and feed-forward joint torque obtained by solving the inverse dynamics was input to actuators.

Plots in Fig. 7 show the z component of the end-effector position and velocity, and their target values. The maximum deviation of the end-effector position from the target was under 0.05 m. The tracking error in joint angular velocity reached a maximum of approximately 0.1 rad/s, but tracking of joint angle using a larger gain achieved an accuracy of 0.05 rad or better.

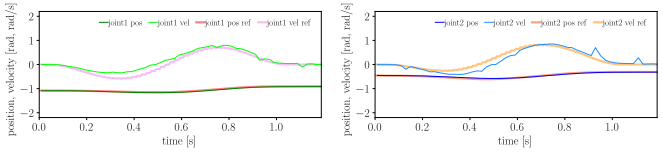
2) Continuous deformation: To verify deformation with coninuous trajectory with varying end-effector speed, we performed a simulation to follow a eight-curve trajectory consisting of two circles shown in Fig. 8 at constant angular velocity. The radius of the circular trajectory was set to 0.4 m, and the angular velocity of the end-effector was set to 2.5 m/s.

The plots in Fig. 9 show the target and actual values of the end-effector position in the yz plane, the actual joint torque, joint velocity, and thrust. Both the joint torque and thrust are within the upper limits shown in Table II and it can be seen that the inverse dynamics can be solved continuously for long trajectory. The end-effector position accuracy was achieved to



(1) Target and actual z coordinate of end-effector in simulation.

(2) Target and actual z velocity of end-effector in simulation.



(3) Target and actual z velocity of end-effector in simulation.

(4) Target and actual z velocity of end-effector in simulation.

Fig. 7: Plots of end-effector trajectory during dynamic deformation simulation.

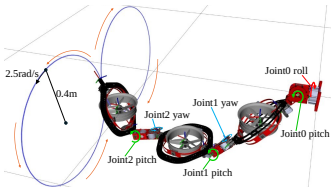


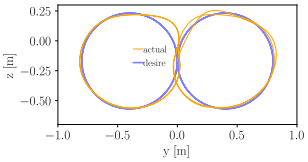
Fig. 8: Reference of circle trajectory.

about 0.05 m, and the end-effector velocity was also achieved with a tracking accuracy similar to that in Sec.V-C.1. The joint velocity was able to achieve a maximum motion of about 2 rad/s.

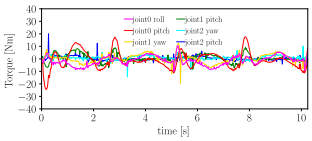
VI. Conclusion and Future Work

In this work, we proposed a torque-thrust integrated inverse dynamics method for dynamic deformation by rotor distributed manipulators. We constructed a control system and verified that dynamic deformation can be achieved by compensating joint torque with thrust through trajectory generation and simulation. We also clarified that the optimization parameters in the cost function of the proposed inverse dynamics method have a significant impact on the actuator inputs. Excessive values may lead to large inputs of either torque or thrust.

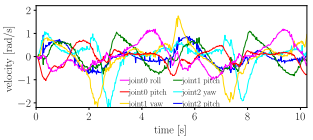
In the future, it is expected to make a system that automatically determines the optimization parameters for a given robot model. Moreover, we will verify the effectiveness of the proposed dynamics on a real robot and apply to various real world tasks. We also aim to extend our method to a floating-base system, enabling dynamic motion in the air. By including the pose of the root link in the configuration vector, high DoF of motion will be realized.



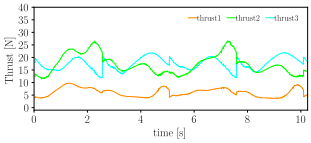
(1) Target and actual position of end-effector on yz plane.



(2) Joint torque



(3) Joint velocity



(4) Thrust

Fig. 9: Plots of circle trajectory tracking simulation.

References

[1] Mihir Dharmadhikari, et al. Motion primitives-based path planning for fast and agile exploration using aerial robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 179–185. IEEE, 2020.

[2] Mario Garzón, et al. An aerial–ground robotic system for navigation and obstacle mapping in large outdoor areas. *Sensors*, Vol. 13, No. 1, pp. 1247–1267, 2013.

[3] Matko Orsag, et al. Valve turning using a dual-arm aerial manipulator. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 836–841, 2014.

[4] Suseong Kim, et al. Aerial manipulation using a quadrotor with a two dof robotic arm. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4990–4995. IEEE, 2013.

[5] Jonathan Fink, et al. Planning and control for cooperative manipulation and transportation with aerial robots. *The International Journal of Robotics Research*, Vol. 30, No. 3, pp. 324–334, 2011.

[6] Karen Bodie, et al. An Omnidirectional Aerial Manipulation Platform for Contact-Based Inspection. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[7] Hyunsoo Yang, et al. Lasdra: Large-size aerial skeleton system with distributed rotor actuation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7017–7023. IEEE, 2018.

[8] Moju Zhao, et al. Transformable multirotor with two-dimensional multilinks: Modeling, control, and whole-body aerial manipulation. *The International Journal of Robotics Research*, Vol. 37, No. 9, pp. 1085–1112, 2018.

[9] Takuzumi Nishio, et al. Design, control, and motion planning for a root-perching rotor-distributed manipulator. *IEEE Transactions on Robotics*, Vol. 40, pp. 660–676, 2023.

[10] Fan Shi, et al. Multi-rigid-body dynamics and online model predictive control for transformable multi-links aerial robot. *Advanced Robotics*, Vol. 33, No. 19, pp. 971–984, 2019.

[11] Moju Zhao, et al. Versatile articulated aerial robot dragon: Aerial manipulation and grasping by vectorable thrust control. *The International Journal of Robotics Research*, Vol. 42, No. 4-5, pp. 214–248, 2023.

[12] Bo Xu, et al. System design and flight control of a flying wheel-legged humanoid robot (fwlr). *Industrial Robot: the international journal of robotics research and application*, 2025.

[13] Yusuke Ueno, et al. Development of hiryu-ii: A long-reach articulated modular manipulator driven by thrusters. *IEEE Robotics and Automation Letters*, Vol. 5, No. 3, pp. 4963–4969, 2020.

[14] Moju Zhao, et al. Design, modeling, and control of a quadruped robot spidar: Spherically vectorable and distributed rotors assisted air-ground quadruped robot. *IEEE Robotics and Automation Letters*, Vol. 8, No. 7, pp. 3923–3930, 2023.

[15] M. Quigley, et al. ROS: an open-source Robot Operating System. In *Open-Source Software Workshop of the 2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009.

[16] N. Koenig, et al. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 2149–2154 vol.3. IEEE, 2004.

[17] Sachin Chitta, et al. ros_control: A generic and simple control framework for ros. *The Journal of Open Source Software*, 2017.

[18] Bartolomeo Stellato, et al. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, Vol. 12, No. 4, pp. 637–672, 2020.

[19] Justin Carpentier, et al. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019.