

Article

Navigation Path Based Universal Mobile Manipulator Integrated Controller (NUMMIC)

Taehyeon Kim ¹, Myunghyun Kim ², Sungwoo Yang ² and Donghan Kim ^{2,*}¹ Department of Electronic Engineering, Kyung Hee University, Yongin 17104, Korea² AgeTech-Service Convergence Major, Department of Electronic Engineering, Kyung Hee University, Yongin 17104, Korea

* Correspondence: donghani@khu.ac.kr

Abstract: As the demand for service robots increases, a mobile manipulator robot which can perform various tasks in a dynamic environment attracts great attention. There are some controllers that control mobile platform and manipulator arm simultaneously for efficient performance, but most of them are difficult to apply universally since they are based on only one mobile manipulator model. This lack of versatility can be a big problem because most mobile manipulator robots are made by connecting a mobile platform and manipulator from different companies. To overcome this problem, this paper proposes a simultaneous controller which can be applied not only to one model but also to various types of mobile manipulator robots. The proposed controller has three main characteristics, which are as follows: (1) establishing a pose that motion planning can be carried out in any position, avoiding obstacles and stopping in a stable manner at the target coordinates, (2) preventing the robot from collision with surrounding obstacles while driving, (3) defining a safety area where the manipulator does not hit the obstacles while driving and executing the manipulation accordingly. Our controller is fully compatible with Robot Operating System (ROS) and has been used successfully with three different types of mobile manipulator robots. In addition, we conduct motion planning experiments on five targets, each in two simulation worlds, and two motion planning scenarios using real robots in real-world environments. The result shows a significant improvement in time compared to existing control methods in various types of mobile manipulator and demonstrates that the controller works successfully in the real environment. The proposed controller is available on GitHub.

Keywords: mobile manipulator; motion planning; simultaneous control; path analysis; ROS

Citation: Kim, T.; Kim, M.; Yang, S.; Kim, D. Navigation Path Based Universal Mobile Manipulator Integrated Controller (NUMMIC). *Sensors* **2022**, *22*, 7369. <https://doi.org/10.3390/s22197369>

Academic Editors: Luige Vladareanu, Hongnian Yu, Hongbo Wang and Yongfei Feng

Received: 13 September 2022

Accepted: 24 September 2022

Published: 28 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the development of artificial intelligence and control technology, the era has come when the service robot directly interacts with humans. The service robot is expected to have an important role in our daily life [1]. However, a current service robot is mostly only used in the same way as a mobile robot, or manipulator robot. A converged platform, in which the manipulator is loaded onto the mobile robot, is called a mobile manipulator, and since it can accomplish unstructured tasks in dynamic environments, it maximizes the use of the robots and is of especially great value in the service area [2]. This is the reason for the current research on mobile manipulators.

There are two types of mobile manipulator controllers, one for sequential control and the other for simultaneous control. Today, most of the mobile manipulator controllers in a real environment such as the field of industry use sequential control. This control method, which is shown in the Figure 1a, controls the manipulator and moves its end-effector to the goal pose after it moves the mobile robot to the manipulable location in a single goal pose. This is due not only to legal reasons, including safety concerns, but also to maintain precision and dexterity [3,4]. On the other hand, the simultaneous control method, which is

shown in the Figure 1b, calculates the desired pose of a mobile robot to a single goal pose, as well as calculating how to manipulate the end-effector to reach the goal pose while the robot moves to the corresponding desired pose. After that, it executes the algorithm controlling both the mobile robot and the manipulator at the same time. Simultaneous control of the mobile manipulator has a great advantage for efficiency, and work is continuously being carried out in this area [5–7].

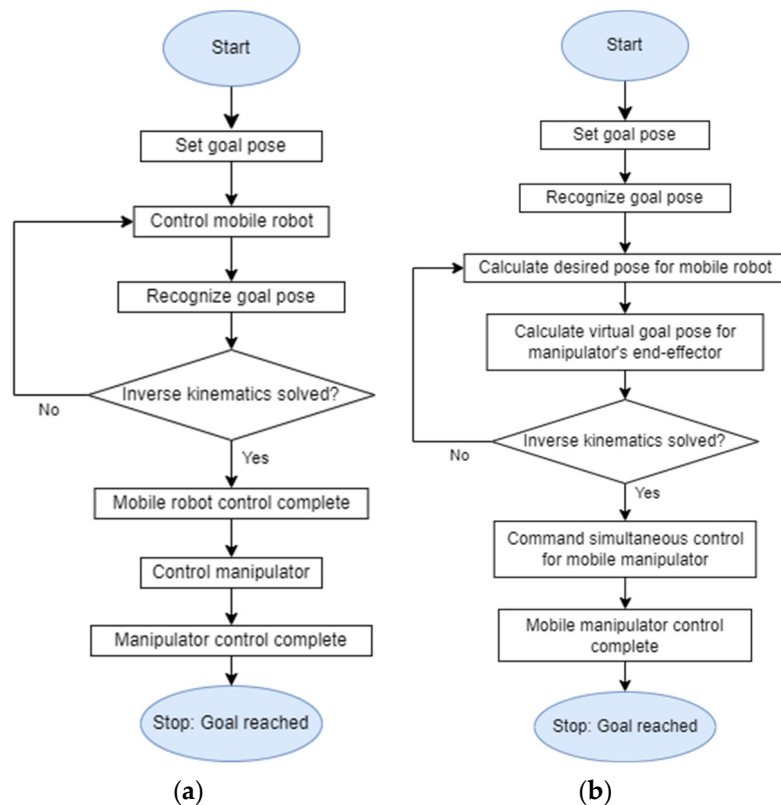


Figure 1. (a) Flowchart of the algorithm controlling mobile manipulator sequentially. (b) Flowchart of the algorithm controlling the mobile manipulator simultaneously.

Meanwhile, because of its complexity, the typical simultaneous controller is designed to control one fixed structured mobile manipulator [8]. However, most of the mobile manipulator robots currently used do not come from one company in the form of a finished product, but are used by combining the mobile platform and the manipulator arm of different companies. This increases the possibility of each user using different mobile manipulator platforms and this situation makes it hard to apply existing simultaneous controllers built on a single model. This paper proposes the Navigation path-based Universal Mobile Manipulator Integrated Controller (NUMMIC), which is a type of kinematic controller. NUMMIC is able to control the mobile platform and manipulator simultaneously by simply changing a few parameters to overcome the existing kinematic controller's limitations. Since most of the existing kinematic controllers need as detailed as possible characteristics of models for precise operation, it is challenging to use it with various types of mobile manipulators. However, in NUMMIC, the parameters required by the algorithm are minimized for stable simultaneous control. Accordingly, it can be applied to various types of mobile manipulators. Here, the only hardware it requires is a mobile manipulator which has a single arm on a mobile platform with Light Detection and Ranging (LiDAR) and Inertial Measurement Unit (IMU) sensor, and the software required is the Unified Robot Description Format (URDF) [9], available on ROS [10] for each part. Any mobile manipulator that satisfies these conditions can be used by the proposed controller with a few parameter revisions. This is possible because the controller performance is based on the platform's navigation control and does not need other sensors except for LiDAR and

IMU sensors during motion planning. Also, it tunes the lower-level kinematic controller rather than directly controlling the hardware at the lower end.

NUMMIC, which is suggested in this paper, is composed of three substructures: a Manipulation Enabled Pose (MEP) setting based on a LiDAR sensor, an optimized end-effector default pose, and a Manipulation Safety Section (MSS) setting based on path curvature. These substructures elaborately control the `move_base` package [11] and the `MoveIt` package [12], which are publicly provided by the ROS as mentioned, and make simultaneous control of the mobile manipulator of the target object possible in different environments and positions. Since the entire controller is operated by coordinating the lower kinematic controller packages, the controller is easily compatible with various mobile manipulators which have different specifications.

The specific structure of NUMMIC can be found in Figure 2. For a detailed explanation of the substructures, the desired pose, which is called Manipulation Enabled Pose (MEP), is needed. This pose which allows the mobile manipulator to be controlled simultaneously using global path is found by MEP setting based on LiDAR sensor. The second substructure, the optimized end-effector default pose, derives the location in which there is least deviation of time, regardless of the position of the target object and its designation as a manipulator's default pose. This idea is based on the fact that, in the operating environment, the manipulator's manipulation is restricted to limited directions. The last substructure, the Manipulation Safety Section (MSS) setting based on path curvature, supports the safe simultaneous control of the mobile manipulator. According to its control algorithm, manipulation is operated at the same time as the mobile platform moves. If the manipulation is being executed or has already completed its execution and the platform moves with the stretched manipulator, the possibility of colliding with obstacles near the path will become much higher. To prevent this, an algorithm from this substructure locates the area which is in no danger of colliding with nearby objects and only performs the manipulation when the mobile platform passes through the area. This area is called the Manipulation Safety Section (MSS).

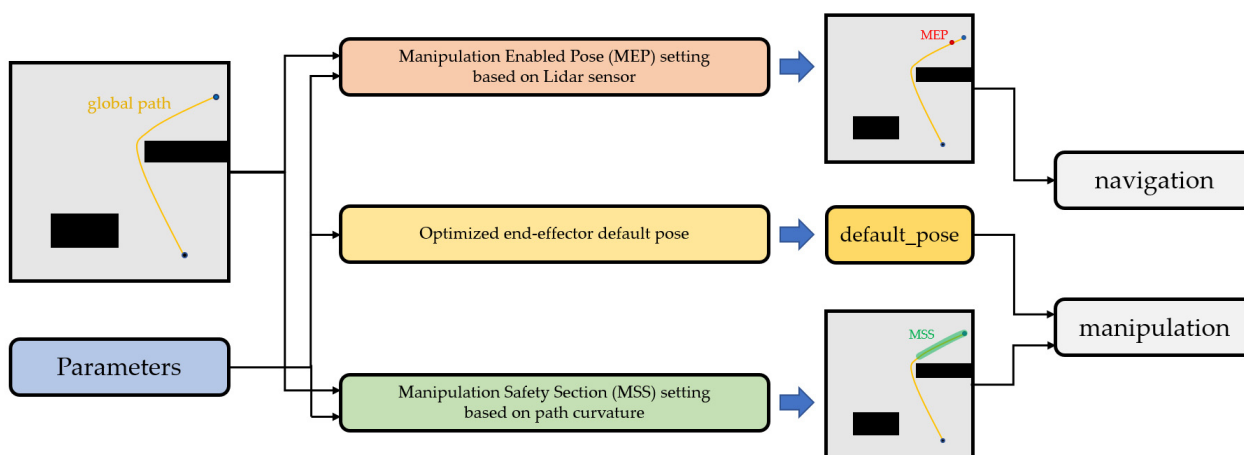


Figure 2. Diagram of NUMMIC's substructures.

In summary, this work suggests the controller, NUMMIC, for simultaneous control of the mobile manipulator. NUMMIC is able to perform in a stable manner the motion planning towards the target object in any environment when its target coordinates are given. This process takes less time compared to the sequential control algorithm, due to the simultaneous control of the mobile platform and the manipulator. Since this NUMMIC controller coordinates the `move_base` and `MoveIt` packages, which can be used publicly at ROS, it can be applied to various manipulator robots with different specifications if some parameters in the controller are modified.

To validate our proposal, we present an experiment for motion planning towards given target coordinates using the mobile manipulator robot in a simulation environment.

To show the motion planning in different environments we use two different maps with various obstacles for the simulation and designate various points in the map as target coordinates. Also, to show that NUMMIC can be used in mobile manipulators with different specifications, we run the simulation with three different types of mobile manipulator: one in which UR3 [13] of universal robot is attached to Husky UGV [14] from Clearpath corporation, another one which has a UR5 [15] manipulator with the same mobile platform, and a third that has Kinova corporation's Gen3 lite [16] manipulator on Jackal UGV [17]. Finally, we conduct an experiment using NUMMIC with a mobile manipulator, in which Husky UGV is attached to a UR3 manipulator in a real environment.

In Section 2 we explain existing approaches to mobile manipulator control. In Section 3 we describe the structure of NUMMIC in detail, including (1) its three substructures (2) the architecture based on ROS using these substructures. We present the experiments in both simulation and within a real environment, discussing the results obtained in Sections 4 and 5. Lastly, the conclusions of the evaluation and future works are presented in Section 6.

2. Related Work

Because of the mobile manipulator's capability, various related studies are conducted on the premise that it is used at different fields. The many fields of application include industry [18,19], which is mentioned above, but also construction [20], agriculture [21–23], disaster [24], and healthcare [25,26].

For a robot to execute complex works in these different fields and environments, it needs a highly qualified manipulation ability. In particular, the efficient control of the mobile manipulator in unstructured dynamic environments is important, but this issue is not completely solved. In Reference [27], it minimizes execution time for the pick-up task by deriving the optimal trajectory of joint space, applying a random profile approach (RPA). There are some works whose methods of control focus on the manipulator's end-effector. [28]. One of those works carries out sampling of the waypoint with the end-effector's target position and orientation trajectory and implements an optimized planner using the Genetic Algorithm for continuous movement of the mobile manipulator. The work in [29] efficiently generates a path without collision in complex environments by suggesting the Optimized Hierarchical Mobile Manipulator Planner (OHMP), which is composed of two steps: two-dimensional mobile motion planning and three-dimensional manipulator motion planning. Recently, due to the development of deep learning, research into mobile manipulator control using reinforcement learning is also being conducted. In [30], the authors suggest a system in which the mobile manipulator robot learns action-related places through experience-based learning with the environment. In Reference [31], they offer a mobile manipulator system with a more efficient framework by decoupling the state-of-the-art deep reinforcement learning control and visual perception.

The controller suggested in this research works as a controlling `move_base`, which is a package for navigation and `MoveIt`, the manipulator control package, on the upper level. There are some studies to find improved action by the selection in which it uses an existing controller in normal situations and replaces the motion with a new controller or adjusts the existing one in particular situations. In Reference [32], the controller normally uses `move_base` to drive the mobile robot and then when the robot faces the narrow space, it regenerates appropriate waypoints for passing the space. Also, in the work of Reference [33], the author suggests an algorithm which executes a more efficient exploration using `Gmapping` [34] and the `move_base` package. Moreover, in the same context, other studies are conducted which extend the function of `move_base` with a higher-level controller [35–37] or uses the `MoveIt` package [38–40] for operation in specific higher-level environments. Although it is not related to replacing or tuning the controller itself, as with reference [41], there are also some studies on user-friendly interfaces, replacing `RViz` [42], the 3D visualization tool which gives order to the existing `move_base`, and `MoveIt` packages, to a VR-based interface. In the case of the controllers suggested in the studies mentioned, they are activated by replacing

or tuning the existing ROS packages in certain environments but have the limitation that they have to be controlled with RViz or terminals. However, in this paper, the authors make a Virtual Reality-based control interface in which the user has access intuitively at higher level. This seems consistent with the approach direction that suggests a higher-level controller which gives order to a lower one without directly handling the lower kinematic controller of the move-base and MoveIt packages.

3. Controller Description

The objective of the controller is to control the mobile manipulator simultaneously in various environments. For its successful operation, when the mobile manipulator receives the position of the target object, it should firstly be able to calculate the pose where it can execute the manipulation towards the target and move to the corresponding pose. The flow chart of NUMMIC for this operation can be found in Figure 3.

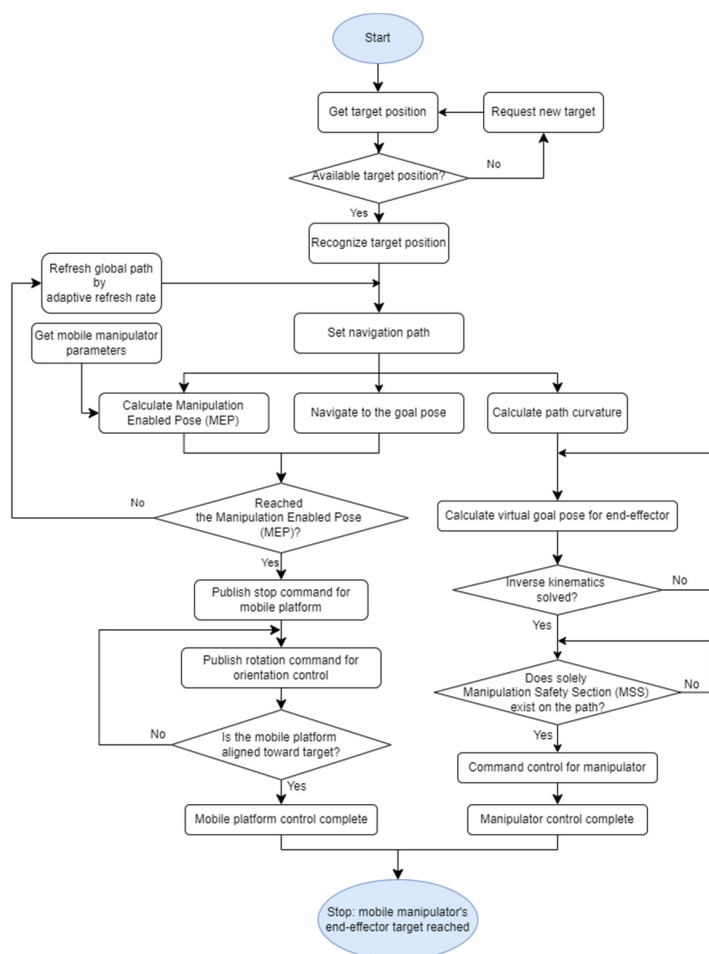


Figure 3. Execution flowchart of NUMMIC.

Following the order of the flow chart, after receiving the available position of the target, NUMMIC sets the navigation path through the global path planner [43] in the move-base package with scan data from the LiDAR sensor attached at the mobile platform. If MEP, a position where the manipulation for the target on navigation path is successfully executed, is specified, the mobile platform of the mobile manipulator can be located on the MEP controlled using move-base. This process is based on some parameters of the mobile manipulator and the information about the position of the target.

At the same time, two additional processes are required for the efficient and stable control of the mobile manipulator, which are the designation of the manipulator’s default pose and the setting for MSS using path’s curvature analysis. The designation of the

suggested default pose is based on the idea that the generated trajectory at manipulation is likely to be a straight line since the mobile manipulator is aligned to face the target object when it stops at MEP. The default pose proposed in this paper is a point where, no matter what value the z-coordinate of the target object has, deviation of the distance from default pose's coordinate to the target, i.e., the trajectory of end-effector during manipulation, is minimized and the manipulator does not hit the surrounding obstacles. Since the state of the manipulator not performing any action while the mobile manipulator drives is defined as the default pose, there is no part related to the calculation of default pose in NUMMIC flow chart of Figure 3. As seen in Figure 2, although the part suggesting an optimized default pose exists in parallel with other substructures, it is decided in advance by the parameters of mobile manipulator and used in NUMMIC rather than being performed in the main operation of NUMMIC. This default pose is calculated with the manipulator's workspace and some parameters including mobile robot's footprint, and by using it to control the manipulator it can reduce the average planning time for target objects in various locations.

The setting for MSS using path's curvature analysis, like MEP generating algorithm, uses a navigation path to find the area with a high risk of collision between the manipulator and surroundings while the mobile robot drives. This operation can also be seen in the flow chart of Figure 3 through the *Calculate path curvature* block on the rightmost side following the *Set navigation path*. The global path generated by the global path planner from move-base is split into steps according to the granularity value, and it analyzes the curvature at each path's interval using the difference of radian values at these steps. It distinguishes the area where there is a possibility of the mobile manipulator colliding with nearby obstacles or not depending on the analyzed curvature, the width of mobile manipulator, and the manipulation distance. Finally, it performs the manipulation only if there is MSS, the area which has no risk for the manipulator to collide with the surroundings on its path between the mobile manipulator and the target object according to the path analysis.

3.1. Manipulation Enabled Pose (MEP) Setting Based on LiDAR Sensor

In this part, the executed operations are as follows: (1) setting the MEP for the manipulation towards the given target and control the mobile platform for the movement to the pose, (2) stopping the MEP using the algorithm which controls the velocity proportional to distance and (3) rotation control based on P-controller to perfectly align the target and the mobile manipulator. These three operations are executed sequentially.

To set the MEP, the controller calculates the Manipulable Area. Manipulable Area is an area in which the mobile manipulator can manipulate the target. The controller then generates a global path to the target using move-base and makes the mobile manipulator drive along the path. Here, the calculation of the Manipulable Area is as follows. A pose is assumed in which the mobile manipulator stopping at MEP performs the manipulation towards the target. Figure 4 shows the pose in the simulation. Here, the first joint from the manipulator's base link becomes the central axis of the recommended workspace. Using the characteristics of NUMMIC algorithm, only values of x and z coordinates in 3-dimensional space during the manipulation are needed since the coordinates of the base link of the mobile manipulator and the target are aligned after the control of the mobile platform. Assuming the radius of the manipulator's recommended workspace to be d_m , z-coordinates of target be t_z , and z-coordinates of manipulator's first joint from the ground be m_z , the angle between the line from the target, which is perpendicular to the ground, and the line extended from target to manipulator's first joint is as follows:

$$\theta = \cos^{-1} \left(\frac{|t_z - m_z|}{d_m} \right) \quad (1)$$

The definition of the distance on the x-axis from the base link of the mobile platform to the first joint of the manipulator as d_{bm} , r_m , a radius of Manipulable Area is as follows:

$$r_m = d_m \sin \theta + d_{bm} \quad (2)$$

Thus r_m , the radius of Manipulable Area is a distance between the perpendicular foot of a point where the target meets the recommended workspace to the ground, and the perpendicular foot of a base link to the ground.

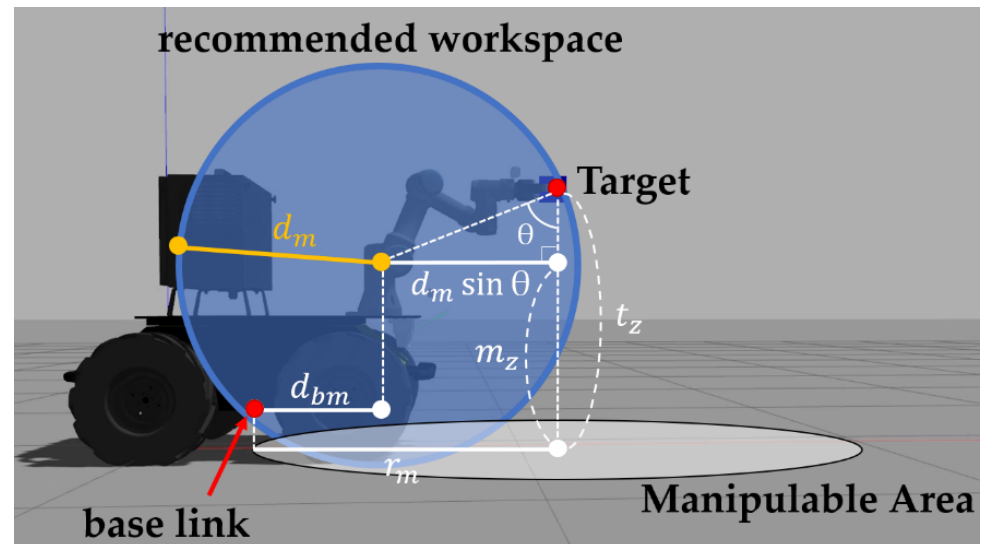


Figure 4. Schematic representation for the correlation between mobile manipulator's recommended workspace, target, and r_m .

Second, the controller stops the mobile manipulator when it is determined that the mobile manipulator has entered the Manipulable Area while driving along the path. If a signal is sent instantaneously, a significant error occurs since the controller controls the mobile robot with move-base. To prevent this, the suggested controller in this paper defines the area equivalent to twice the radius of the Manipulable Area as sigmoid distance proportional speed control area and adjusts the velocity of the mobile platform in this area. The velocity decelerates along the shape of the sigmoid function in proportion to the distance from the target point. When v_x is the translational velocity of mobile platform, v_{max} is the maximum velocity of mobile platform, v_{stop} is the velocity when mobile platform is stopped, and d is the distance between the mobile platform and the MEP, then the formula of the sigmoid function is as follows.

$$v_x = \frac{v_{max} - v_{stop}}{e^{-d} + 1} + v_{stop} \quad (3)$$

To converge to the velocity v_{stop} when it reaches the target point, the function is shifted to the positive direction as follows:

$$v_x = \frac{v_{max} - v_{stop}}{e^{-d+6} + 1} + v_{stop} \quad (4)$$

Finally, assuming the radius of predefined sigmoid distance proportional speed control area to be r_{sa} , the coefficient to decelerate after entering the area is revised.

$$v_x = \frac{v_{max} - v_{stop}}{e^{-\frac{12}{r_{sa}}d+6} + 1} + v_{stop} \quad (5)$$

In order to verify how the mobile platform of the mobile manipulator is controlled through the above Formula (5), a graph is drawn assuming the specific situation. Because the maximum speed of Husky UGV which is used in the experiment of the paper is 1 m/s, v_{max} is set at 1 m/s and the targetting velocity at stop is set at 0 m/s. Then, assuming that r_{sa} about particular target A is 1.2 m, a graph of the corresponding sigmoid function is drawn as Figure 5. From this graph, it can be expected that as the mobile manipulator goes

beyond the sigmoid distance proportional speed control area and approaches the target, its speed will decrease and stops in a stable manner at MEP.

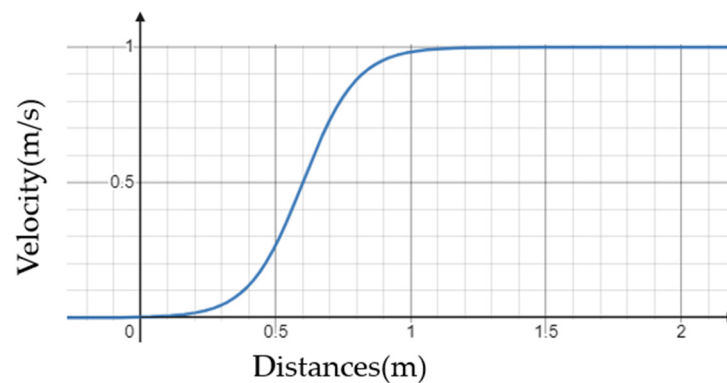


Figure 5. A distance-velocity graph for applying sigmoid distance proportional speed control at a point 1.2 m away from the target with a mobile platform which has a maximum speed of 1 m/s.

After the mobile platform is stopped by the sigmoid distance proportional speed control algorithm, it rotates the platform to face toward the coordinates of target object. This rotational control is composed of simple P-control which multiplies the gain value to the difference of mobile platform's yaw value and target one, and this gain can be modified according to the needs of the user at NUMMIC's configuration, even though it is specified to 0.25 as default.

The final pose of the mobile manipulator becomes the MEP through a series of control processes for detailed alignment of the mobile platform including the designation of MEP by target coordinates and specification of the mobile manipulator, stop at the position, and rotational control. The graphical explanation about MEP can be found in Figure 6. Figure 6a shows where the MEP is located on the path, based on the creation of a navigation path which has the position of target object from the mobile manipulator as goal on RViz. Figure 6b shows the Manipulable Area, a circular area with radius r_m that has the target object as a center when the mobile manipulator is located closer to the target object, and the sigmoid distance proportional speed control area with radius $2r_m$ which also has target object as a center. In this case, the intersection of the navigation path from the mobile manipulator to target object and the Manipulable Area becomes the position of the MEP.

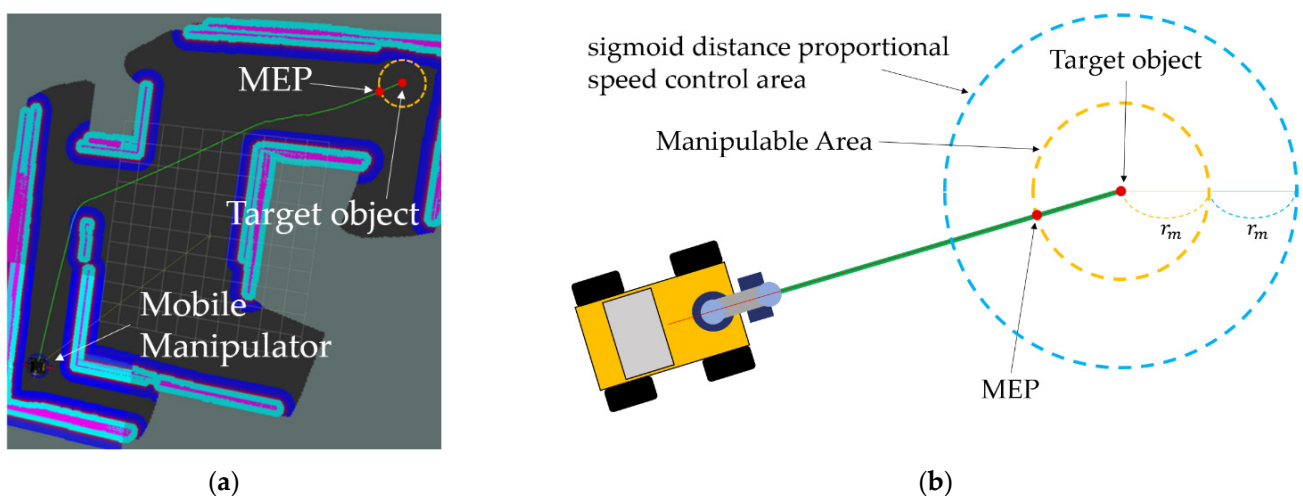


Figure 6. Schematic description of the Manipulation Enabled Pose (MEP). (a) shows where the MEP is located on the navigation path. (b) shows the Manipulable Area and the sigmoid distance proportional speed control area according to r_m .

3.2. Optimized End-Effector Default Pose

The default pose of the end-effector suggested in this paper is a pose in which there is no collision when the mobile manipulator drives, and it does not lose much time in the planning of the manipulation, whatever value the target's z -coordinate has. The z -coordinate of the suggested pose is assumed to be k , and distance at x -axis from the base link of manipulator to the front footprint value on move-base of the mobile platform to be d_{fm} , which can be described as Figure 7. Figure 7a represents d_m , the radius of mobile manipulator's recommended workspace, d_{fm} , the distance on the x -axis from the manipulator's base link to the front footprint of the mobile platform, and d_{bm} , the distance on x -axis from the mobile manipulator's base link to the manipulator arm's base link. Figure 7b represents an arbitrary position of the target (blue point) and the expected position of default pose (green point on the green line) through a circle with the first joint from the manipulator's base link as the origin and d_m as the radius. A red point is the target which has a minimum z value among the target, and a purple point is the target which has a maximum z value. This range can be modified by controller's parameter values. X , expected planning trajectory distance value from default pose to the target object, can be derived as follows:

$$X = \sqrt{(x - d_{fm})^2 + (\sqrt{d_m^2 - x^2} - k)^2} \quad (6)$$

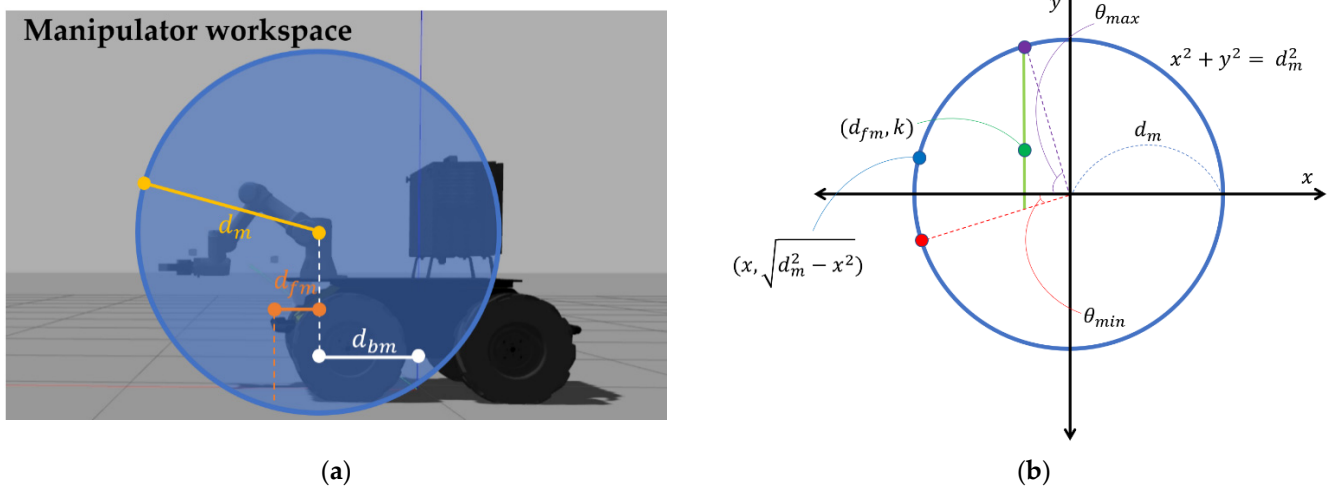


Figure 7. Schematic description of the position for the default pose. (a) represents d_m , d_{fm} , d_{bm} on the simulation mobile manipulator model. (b) represents the expected position of default pose (green point) in a simplified drawing.

To find a default pose required by the controller, the formula should be expressed as the mean value and variance of continuous probability variable as follows:

$$E_{distance}(X) = \int_{-d_m}^{-d_{fm}} x \sqrt{(x - d_{fm})^2 + (\sqrt{d_m^2 - x^2} - k)^2} dx = m \quad (7)$$

$$V_{distance}(X) = \int_{-d_m}^{-d_{fm}} (x - m)^2 \sqrt{(x - d_{fm})^2 + (\sqrt{d_m^2 - x^2} - k)^2} dx \quad (8)$$

When the $V_{distance}(X)$ is minimum, k becomes the z -coordinate of manipulator's default pose. Although the above expression is suitable for understanding the concept of suggested default pose, it is difficult to calculate the desired k value due to its complexity. Thus, for ease of calculation, we exchanged the values of x and y -axis, and the values in the range of integrals are replaced by discrete ones which take into consideration the manipulator's operating range and limits in control. The revised formula is given below.

Here, z_{max} is the value in which the maximum height for the mobile manipulator to perform manipulation is subtracted by the z -coordinate value of manipulator's first joint from the base link, and z_{min} is the value in which the minimum height for manipulation is subtracted by the z -coordinate value of manipulator's first joint.

$$X_2 = \sqrt{(x - k)^2 + \left(\sqrt{d_m^2 - x^2} - d_{fm}\right)^2} \quad (9)$$

$$E_{distance}(X_2) = \int_{z_{min}}^{z_{max}} x \sqrt{(x - k)^2 + \left(\sqrt{d_m^2 - x^2} - d_{fm}\right)^2} dx = m_2 \quad (10)$$

$$V_{distance}(X_2) = \int_{z_{min}}^{z_{max}} (x - m_2)^2 \sqrt{(x - k)^2 + \left(\sqrt{d_m^2 - x^2} - d_{fm}\right)^2} dx \quad (11)$$

As in Expressions (6)–(8), when the $V_{distance}(X_2)$ is minimum, k becomes the z -coordinate of manipulator's default pose ($z_{min} \leq k \leq z_{max}$). Finally, the k value which is derived from the above calculation is designated as the z -coordinate of the default pose and d_{fm} , the distance to the front footprint of the mobile platform based on the manipulator's base link is designated as the x -coordinate of the default pose. After defining the y -coordinate as the nearest point to 0 where the manipulator can move satisfying these two values, and inputting the orientation value so that the end-effector looks at the front of the mobile manipulator in the corresponding position, the default pose is set.

3.3. Manipulation Safety Section (MSS) Setting Based on Path Curvature

For successful simultaneous control of the mobile manipulator in various environments which include obstacles, it is necessary to prevent the manipulator from colliding with its surroundings. For this reason, in this paper, we analyze the global path where the mobile manipulator drives and make the manipulator decide whether there is a large curve that it might bump into when it is stretched forward along the path from the current position to the target. Assuming the radius of Manipulable Area about specific target of mobile manipulator to be r_m , the width of the mobile platform to be w_m , and the mobile manipulator to be a material particle that moves along the global path, the curvature radius ρ about the differential length of the path is required.

Generally, this curvature radius ρ , which is decided by the arc between two adjacent points of the trace, is expressed as follows:

$$\rho = \frac{ds}{d\theta} \quad (12)$$

Meanwhile, the global path from move-base is in the form of an array composed of numerous x and y coordinates. Differential length, which is called path step in this paper, can be decided using the difference between the sequential values of these x and y coordinates. Each of these path steps has a certain distance value and they are preset in advance which can be checked at the configuration of move-base package. This differential length is called path granularity, which refers to ds of above Formula (12). $d\theta$ can also be derived from the angular difference of each continuous path step. The equation for calculating the curvature radius by applying the above is as follows:

$$\rho = \frac{\text{path granularity}}{\text{difference of path step angular}} \quad (13)$$

Using curvature radius ρ and width of mobile platform w_m , the area where the manipulator does not collide with nearby obstacles while the mobile platform passes specific path step with its stretched arm can be calculated. This area corresponds to the space between the arc of a circle whose radius is equal to the curvature radius ρ plus half of w_m , and the arc of a circle whose radius is same as the value, ρ is subtracted by half of w_m . This is

described as the green space at Figure 8. If the curvature of the path is large enough for the end-effector to be outside the green space when the manipulation is executed at that moment, the path step, which includes the base link (a green dot inside the mobile platform in Figure 8) of the mobile manipulator, is treated as not belonging to MSS. Algorithm 1 is a pseudo-code based on Python for deciding the MSS by global path.

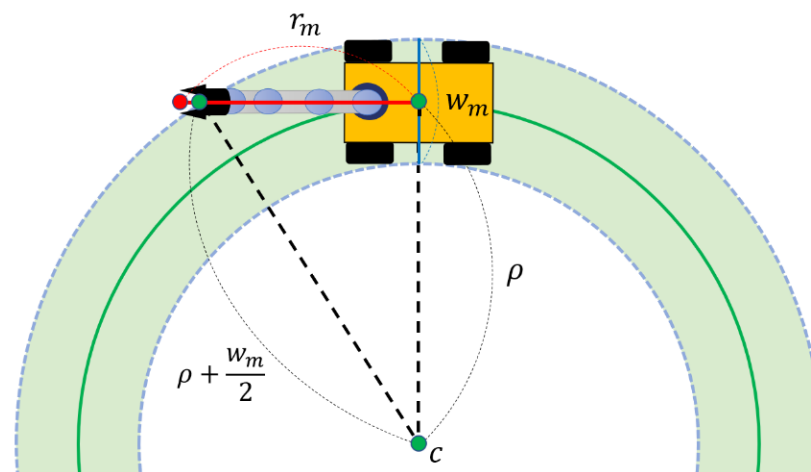


Figure 8. Schematic description for the calculation of Manipulation Safety Section (MSS).

After distinguishing the MSS and the section which is not on the global path from the mobile manipulator's start point towards the target, if only the MSS exists on the remaining path from the path step, which is closest to the location where the mobile manipulator is driving, to the target, manipulator's planning will be executed. Through the process, mobile manipulator can be controlled by simultaneous control without collision in complex environments including obstacles.

3.4. ROS-Based Architecture

NUMMIC is implemented through the Python scripts based on ROS, and it controls a whole mobile manipulator with `move_base`, which controls the mobile platform, and `MoveIt`, which controls the manipulator, solving inverse kinematics. The structure of NUMMIC packages on ROS is shown as Figure 9.

`Controller.launch` executes the nodes at the same time which are required for operating NUMMIC package through python scripts inside the script folder. `default_pose_cal.launch` executes the python script which calculates the position of proposed default pose using the value inside `controller_param.yaml` for the user to control the default pose that is proposed in this paper before operating NUMMIC. `controller_param.yaml` file obtains the specification of the mobile manipulator and additional setting parameters related to the operation of the controller and sends them to each node through `rospy.get_param` function. A brief description about input and additional parameters at `controller_param.yaml` file can be found in Table 1. `goal_nav.py`, `goal_nav_stop_combine.py`, and `orientation_check.py` are the Python scripts for the MEP setting, stop at MEP, and detailed rotational control after stopping. `curvature_check.py` sends the decision about MSS and its result to `manipulation.py`. `manipulation.py` executes the manipulation to locate at the proposed default pose, which is calculated in advance by the manipulator based on MSS, or at virtual goal pose, which is to reach the target coordinates for the end-effector or manipulator when the mobile platform is stopped at MEP. `default_pose_calculator.py` calculates the position of default pose which will be used with this controller according to the parameter values set by the user. Figure 10 represents how the communication using the topics works in the process in which NUMMIC controls the mobile manipulator through the external `move-base` package and `MoveIt` package on ROS. The proposed controller is released as an open-source repository on GitHub [44].

Algorithm 1: Determining the Manipulation Safety Section (MSS) from curvature of global path

Data: Path data: path_msg $[[\text{path_msg}[0]_x, \text{path_msg}[0]_y], [\text{path_msg}[1]_x, \text{path_msg}[1]_y], \dots, [\text{path_msg}[N-1]_x, \text{path_msg}[N-1]_y]]$; Path granularity: pg ; Width of mobile robot: w_m ; Radius of Manipulable Area: r_m

Result: Manipulation Safety Section (MSS) on the global path

```

1  path = []
2  path_angular = []
3  path_angular_modified = []
4  MSS = []
5  path_step = N - 1
6  path_offset = round( $r_m/pg$ )
7  # Get path data
8  for i = 0 to path_step do:
9      path.append([path_msg[i]x, path_msg[i]y])
10 end for
11 # Calculate path step angular
12 for i = 0 to path_step - 1 do:
13     yawx = path[i + 1][0] - path[i][0]
14     yawy = path[i + 1][1] - path[i][1]
15     angular = atan2(yawy, yawx)
16     if angular < 0:                                     # Match the signs of the angular
17         angular = 2π + angular
18     end if
19     path_angular.append(angular)
20 end for
21 # Get difference of path step's angular
22 for i = 0 to path_step - 2 do:
23     angular_modified = path_angular[i + 1] - path_angular[i]
24     if angular_modified < -π:
25         angular_modified = 2π + angular_modified
26     elif angular_modified > π:
27         angular_modified = 2π - angular_modified
28     end if
29     if i ≥ ((path_step - 2) - path_offset):             # Exclude the path step belonging to Manipulable Area
30         angular_modified = 0
31     end if
32     path_angular_modified.append(|angular_modified|)
33 end for
34 # Get MSS
35 for i = 0 to path_step - 2 do:
36     if path_angular_modified[i] == 0:
37         radius_of_curvature = 0
38     else:
39         radius_of_curvature = (pg/path_angular_modified[i])
40     end if
41     if radius_of_curvature == 0:                         # Position with True values on this array is MSS
42         MSS.append(True)
43     elif sqrt(radius_of_curvature + (wm/2)2 - (radius_of_curvature)2) > rm:
44         MSS.append(True)
45     else:
46         MSS.append(False)
47     end if
48 end for

```

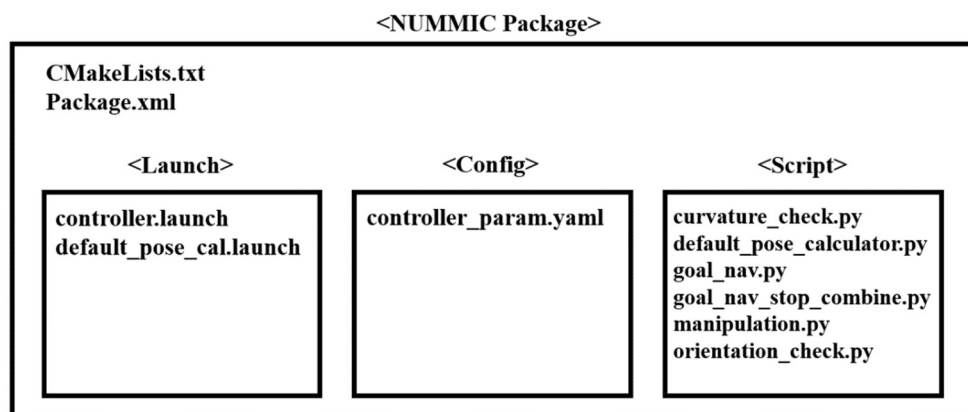


Figure 9. Structure of NUMMIC Package based on ROS.

Table 1. Parameters that can be set in controller_param.yaml of NUMMIC Package, and brief description about them.

Parameter	Description
/first_joint_height	z-axis distance from ground to manipulator’s first joint (m).
/between_base_link	x-axis distance from mobile robot’s base_link to manipulator’s base_link (m).
/recommended_reach	Manipulator’s recommended workspace (m).
/mobile_robot_width	Width of the mobile robot (m).
/base_link_offset	Height of the mobile robot’s base_link (m).
/orientation_kP_value	kP value in orientation control.
/refresh_cycle	Global path refresh cycle (s).
/granularity	The step size to take between points on a given navigation trajectory (m).
/m_bl_to_ft	x-axis distance from manipulator’s base link to front footprint (m).
/z_max	Maximum z value in manipulation (m).
/z_min	Minimum z value in manipulation (m).

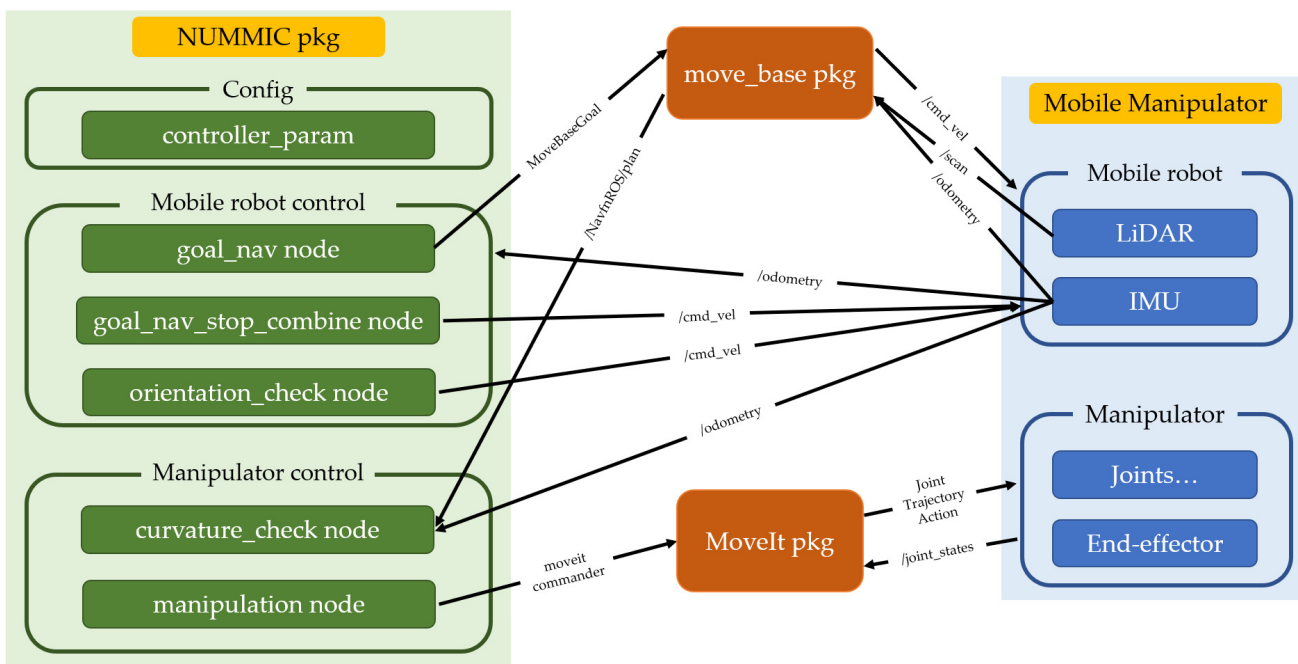


Figure 10. Topic exchanges between nodes in NUMMIC package based on ROS.

4. Experimental Setup

Prior to the experiment evaluating the suggested controller, we would like to explain the simulation world for the experiment, give a detailed specification of mobile manipulator, and the parameter values of NUMMIC and navigation package.

4.1. Simulation Worlds and Robotics Platforms

The simulation is performed for three different mobile manipulator platforms in two different worlds. Each simulation world, corresponding map files, and target coordinates are described at Figure 11. Among them, the simulation experiment in the first world involves the mobile manipulator, which always moves from the origin when motion planning for each target. Throughout this experiment, we can evaluate the mobile manipulator with the suggested controller on whether it can perform the motion planning from a fixed point to different places, avoiding the obstacles. The experiment in the second world involves the mobile manipulator that moves from MEP of previous target when it executes the motion planning for each target. Throughout the experiment, it is possible to verify that mobile manipulators existing in various locations can once again carry out motion planning towards different locations while avoiding obstacles. Also, since Test world #2 has a room-like structure in the middle, the mobile manipulator should pass through a narrow gap and exit the structure for motion planning toward Target 3 after Target 2. The experiment in such an environment is required to determine whether a collision exists during driving, which is important in a word of mobile manipulator using simultaneous control, and also it will be possible to verify the usefulness of the decision for MSS at the controller.

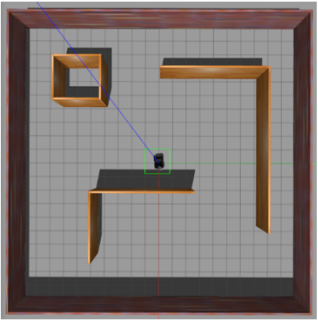
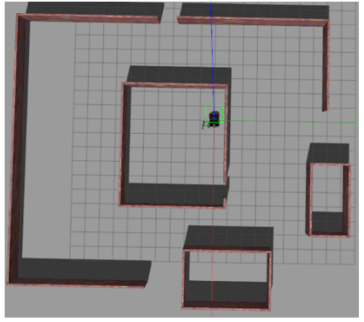
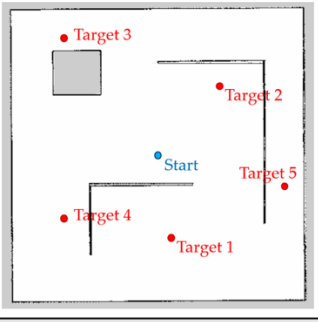

World	Test world #1	Test world #2
Appearance		
Map		
Target #	[x, y, z] coordinates (m)	
1	[6, 1, 0.4]	[4, -5, 0.35]
2	[-4, 5, 0.533]	[1, -1, 0.6]
3	[-8.5, -7, 0.6]	[0, 6, 0.533]
4	[4.5, -7, 0.7]	[-5, -9, 0.533]
5	[2, 9, 0.5]	[1, -11, 0.8]

Figure 11. Appearance of Test worlds for each simulation tests, their map files, and [x, y, z] coordinates of target points.

As mentioned previously, the mobile manipulators used for the experiments are three different types: Husky UGV with UR3 (Husky_UR3), Husky UGV with UR5 (Husky_UR5), and Jackal UGV with Kinova Gen3 lite (Jackal_Kinova). The hardware specifications of each mobile manipulator can be found in Figure 12. In addition, there are some remarks for each mobile manipulator used in the experiment. First, Husky_UR3 and Husky_UR5 share the same mobile platform but since Husky_UR3 has a bracket structure between the mobile platform and the manipulator, and Husky_UR5 does not. The height of the first joint from the manipulator's base link is higher at Husky_UR3. Second, for the UR series provided by Universal Robots, there is a recommended reach separate from the maximum reach, but the Gen3 lite model from Kinova does not have the recommended reach described in the manual. Thus, when we used NUMMIC in the case of the UR series, the recommended reach value described in the manual was used for the parameters, but in the case of Gen3 lite, we reduced the appropriate value in the maximum reach and used it as recommended reach.




Model	Husky_UR3	Husky_UR5	Jackal_Kinova
Appearance			
Mobile platform	Husky UGV External dimensions: 990 × 670 × 390 mm Internal dimensions: 296 × 411 × 155 mm Weight: 50 kg Payload: 75 kg Max speed: 1.0 m/s LiDAR: SICK-TIM571	Husky UGV External dimensions: 990 × 670 × 390 mm Internal dimensions: 296 × 411 × 155 mm Weight: 50 kg Payload: 75 kg Max speed: 1.0 m/s LiDAR: SICK-TIM571	Jackal UGV External dimensions: 508 × 430 × 250 mm Internal dimensions: 250 × 100 × 85 mm Weight: 17 kg Payload: 20 kg Max speed: 2.0 m/s LiDAR: SICK-TIM571
Manipulator	UR3 Weight: 11 kg Payload: 3 kg Reach: 500 mm Degrees of freedom: 6 rotating joints Recommended reach: 450 mm	UR5 Weight: 18.4 kg Payload: 5 kg Reach: 850 mm Degrees of freedom: 6 rotating joints Recommended reach: 750 mm	Gen3 lite Weight: 5.4 kg Payload: 0.5 kg Reach: 760 mm Degrees of freedom: 6 rotating joints

Figure 12. Hardware specifications of three mobile manipulators.

4.2. NUMMIC Parameters

In order to perform simultaneous control successfully for the mobile manipulator, it is necessary to put a proper parameter value according to the specifications of the mobile manipulator to be used. Although each parameter value is described sufficiently in the previous Section 3.4, this section will further explain why these values are added by the specifications. Parameters for each mobile manipulator used for experiments are represented at Table 2.

In the case of */first_joint_height*, Husky_UR3 and Husky_UR5, using similar manipulator and the same mobile platform, should have almost same value, but as described above, Husky_UR3 has a larger value due to the difference in bracket structure. The Jackal_Kinova model has the lowest platform height but its value is higher because of the structural characteristics of the manipulator. For the */between_base_link* parameter, Husky_UR3 and Husky_UR5 have the same value because the bracket structure mentioned earlier only affects the height, and Jackal_Kinova's is close to 0 since the manipulator is attached to the center of the mobile platform. The values of */recommended_reach*, */mobile_robot_width*, and */base_link_offset* are input based on the hardware specification manual, and since only for Jackal_Kinova model */recommended_reach* value is not listed on the manual, the parameter which is obtained by subtracting the arbitrary value from the maximum value is used. High */orientation_KP_value* will reduce the time to control rotation but reduce the accuracy of

motion planning, and vice versa. In this experiment, the same value is applied to all mobile manipulator models for variable control. Also, the `/refresh_cycle` value is the same for each model due to variable control, even though reducing the value can affect better driving with more system loading. `/granularity` and `/m_bl_to_ft` are written based on the parameter values in the `move_base` package of mobile platforms used for each mobile manipulator. Therefore, for models using the same Husky UGV, the same parameter value is applied on the NUMMIC. In the case of `/z_max` and `/z_min`, the user can enter any value as long as the mobile manipulator specification allows it, but here, the value corresponding to the limit value of the specification is added. `/z_max` is the z value that the manipulator can extend beyond the footprint of the mobile platform as much as possible, and `/z_min` is the height value of LiDAR sensor attached to the mobile platform. The reason why `/z_min` is set in this way is that if `/z_min` is smaller than the height of LiDAR, the LiDAR sensor may be covered by manipulator and interfere with driving.

Table 2. Parameter values of NUMMIC package for each mobile manipulator used in experiments.

Parameters	Husky_UR3	Husky_UR5	Jackal_Kinova
<code>/first_joint_height</code>	0.533	0.474	0.52
<code>/between_base_link</code>	0.331	0.331	0.01
<code>/recommended_reach</code>	0.45	0.75	0.71
<code>/mobile_robot_width</code>	0.67	0.67	0.43
<code>/base_link_offset</code>	0.13	0.13	0.065
<code>/orientation_KP_value</code>	0.25	0.25	0.25
<code>/refresh_cycle</code>	3	3	3
<code>/granularity</code>	0.025	0.025	0.02
<code>/m_bl_to_ft</code>	0.169	0.169	0.25
<code>/z_max</code>	0.95	1.204	1.184
<code>/z_min</code>	0.312	0.312	0.25

4.3. Other Parameters

Since NUMMIC is operated at upper level of the `move_base` stack and `MoveIt` package, fine-tuning of these packages is significant for successful operation. In particular, the adjustment of parameters related to navigation plays a major role for the mobile manipulator to move successfully and efficiently to the MEP. In fact, this process should be delicately determined according to different specifications of each mobile manipulator and the configuration of the map, but in this paper, only the minimum parameter values are modified based on the default value in GitHub code supporting for each mobile platform from their companies. Because a duration of time, the time spent on Manipulation for evaluating algorithms in this paper is measured from the moment when the mobile manipulator enters the sigmoid distance proportional speed control area until the manipulation is completed. Fine-tuning of the parameter values might improve the total driving time or position error of end-effector but considering that the results of each algorithm are compared in this paper, the overall conclusion will be maintained.

Thus, we use Navfn [45] for the global path planner, and DWA planner [46] for the local path planner [47] based on navigation parameters in the repository for Husky UGV [48] and the repository for Jackal UGV [49]. `Inflation_radius` and `min_vel_x` values are modified slightly. For successful manipulation without collision at any points near the wall, the `inflation_radius` value is fixed at 0.1. Also, at Test world #2, to check whether designation of MSS is successfully carried out, the mobile manipulator should pass through the narrow gap between walls. Thus, we modify `min_vel_x` value to -0.1 and make the mobile platform temporarily move backward to pass the gap easily. The pose estimate of the mobile manipulator robot is performed by AMCL [50] that the most commonly

used Monte-Carlo Localization (MCL) algorithm implemented on ROS. With the MoveIt package, the basic setting is performed based on the URDF of each manipulator arm using the supported MoveIt Setup Assistant [51]. Here, KDL Kinematics Plugin [52] was chosen for the kinematics solver.

5. Experimental Results

This section covers the analysis of the results of experiment for verifying our proposed controller's performance. In Section 5.1 we analyze the results for a total of 45 times of the manipulation test with three different mobile manipulators from Target 1 to 5 in the Test world #1 simulation environment. In Section 5.2 we repeat the previous motion planning test in Test world #2 and analyze the results. Lastly, in Section 5.3, we conduct two different motion planning scenarios four times each with an actual Husky_UR3 mobile manipulator robot in real environments and analyze the results for this experiment.

5.1. Experiment #1: Simulated Robots in Test World #1

Table 3 shows the comparison of the average duration of time value with respect to the result of motion planning for each target using Conventional algorithms and the NUMMIC suggested in this paper. The first algorithm for comparison is the sequential controller (SC) which controls the mobile manipulator in a sequential manner. This controller executes the manipulation after its mobile platform has completely finished moving. Because it is possible to compare performance with NUMMIC when it can do motion planning for various points in the map, only the idea of MEP of NUMMIC algorithm is applied in the process of driving the mobile platform. The second algorithm, simply combined only with move_base and MoveIt (OMM), controls only the mobile platform and manipulator arm simultaneously and does not contain the velocity control which is used with NUMMIC for stability in the motion planning process and MSS decision. Duration is a measure of the time between when the mobile manipulator enters the sigmoid distance proportional speed control area and when the manipulation is completed. The reason why duration is defined in this way is that when the overall motion planning time is calculated, the duration of time varies significantly, depending on the distance from the origin to target, thus it is difficult to compare each time for different control methods. Table 4 represents the average values of the Euclidean distance error of x -axis, y -axis, and x - y plane for target coordinates and the mobile manipulator's end-effector at the end of motion planning separately for each pre-mentioned algorithm. The reason why the error value about z -axis is not in the table is that the error on z -axis is always maintained at 0 compared to the errors on x -axis and y -axis which changed significantly by each algorithm.

Table 3. The result comparing the average values of duration using different algorithms for each mobile manipulator in Test world #1.

Average Duration of Time (S)	SC	OMM	NUMMIC
Husky_UR3	26.2	10.97	16.22
Husky_UR5	25.01	9.2	14.99
Jackal_Kinova	23.12	8.59	12.68

Table 4. The average values of error of end-effector after motion planning using different algorithms for each mobile manipulator in Test world #1.

Average Error (cm)	SC			OMM			NUMMIC		
	x	y	ED *	x	y	ED	x	y	ED
Husky_UR3	0.7	0.54	0.95	3.32	3.34	5.28	0.68	0.46	0.83
Husky_UR5	0.9	0.42	1.03	6.42	4.06	8.18	0.62	0.72	0.99
Jackal_Kinova	1.14	1.04	1.68	6.2	4.66	8.05	1.58	1.18	2.17

* Euclidean distance.

The result shows that the duration was reduced by about 41% for every mobile manipulator robot when we use NUMMIC for motion planning compared to the motion planning with the existing Sequential Controller. The simultaneous control of only move_base and MoveIt can reduce the time more drastically than the method proposed in this paper, but accuracy of motion planning becomes significantly lower. In fact, the control method using only move_base and MoveIt is about +400 to +700% which is a huge surge compared to Sequential Controller. On the other hand, NUMMIC has fewer errors than that of the controller with only move_base and MoveIt, which has −12% of decreasing error or +29% increasing error with Sequential Controller.

The reason for this result seems to be closely related to which process the errors in this experiment mainly come from. Since every error is shown on the x -axis and the y -axis, not on the z -axis, it is estimated that the errors in the end-effector position are caused mostly by the navigation process. There are various factors for the cause of errors in the navigation process including the error from the localization between the map file and driving environment, the error by the slip of wheel, and the error that occurs when starting and stopping. These navigation errors affect the motion planning of the mobile manipulator to make errors, and in the case of OMM, which has a particularly large error, it seems that a larger error has accumulated during the stopping at the MEP. Here, it suddenly stopped without velocity control in the sigmoid distance proportional speed control area, which is one of the NUMMIC's internal algorithm. Therefore, it takes less duration than NUMMIC, but a large slip occurs during a sudden stop and causes a large error in the end-effector position. On the other hand, in the case of NUMMIC, when entering a preset area, it stops in a stable manner at the MEP by sigmoid distance proportional speed control. This process makes a little time loss but it is still faster than Sequential Controller and, in terms of accuracy, it is not significantly different from Sequential Controller.

5.2. Experiment #2: Simulated Robots in Test World #2

The overall procedure is similar with previous Section 5.1. Table 5 compares the average value of duration for each target by algorithm, and Table 6 represents the average value of the Euclidean distance of the x -axis, y -axis, and x - y plane for the target coordinates and the mobile manipulator's end-effector after motion planning is completed by each algorithm. However, there is a noticeable point which differs between Tables 3 and 4 in Section 5.1. In the case of the control method with OMM applied to Husky_UR5, when the motion planning toward Target 3 from the position of Target 2 is executed, the manipulator always hits the wall and fails to execute. Thus, the tables below show the average value except for that result.

Table 5. The result comparing the average values of duration by different algorithms for each mobile manipulator in Test world #2.

Average Duration of Time (S)	SC	OMM	NUMMIC
Husky_UR3	23.92	9.98	15.25
Husky_UR5	24.73	11.37	15.5
Jackal_Kinova	22.72	8.19	13.49

Table 6. The average values of error of end-effector after motion planning by different algorithms for each mobile manipulator in Test world #2.

Average Error (cm)	SC			OMM			NUMMIC		
	x	y	ED *	x	y	ED	x	y	ED
Husky_UR3	0.36	0.66	0.79	5.3	5.68	8.12	0.32	0.62	0.72
Husky_UR5	0.38	0.58	0.77	1.88	2.58	3.62	0.62	0.84	1.15
Jackal_Kinova	1.7	1.52	2.48	4.22	4.86	6.77	1.52	1.08	2.14

* Euclidean distance.

Through the table, the experiment conducted in the Test world #2 shows similar results to 5.1 for all mobile manipulator robots. For NUMMIC, the duration is reduced by about 38% compared to the existing Sequential Controller. And for the method with only move_base and MoveIt, the duration greatly decreases compared to NUMMIC but there is a problem that the error on the end-effector position becomes too large.

Additionally, there are several results that can be established with this experiment. One of them is the collision between the manipulator and the wall during the motion planning from Target 2 to Target 3 with Husky_UR5 and the controller with only move_base and MoveIt, which are mentioned previously. This controller simply controls the mobile platform and the manipulator arm simultaneously without any additional tuning, and it might cause serious problem in the motion planning process. In the case of the manipulator of Husky_UR3 and Jackal_Kinova, there isn't any collision because of their short reach, but Husky_UR5's manipulator, which has relatively long reach, can hit the obstacles. This situation can be seen in Figure 13a. In the second scene of the figure, the mobile manipulator reaches out its arm and tries to pass through a narrow gap in the third scene, resulting in a collision, which causes a completely distorted costmap, as in fourth and fifth scene. With NUMMIC, as shown in Figure 13b, this collision does not occur because it goes through the process of determining the MSS through the global path. In fact, it does not implement the manipulation because there is an area with the risk of collision on the remaining global path until the second scene of Figure 13b. However, from the third scene, which depicts the mobile robot passing a narrow gap, it starts the manipulation and successfully completes the motion planning for the target point in the fifth scene.

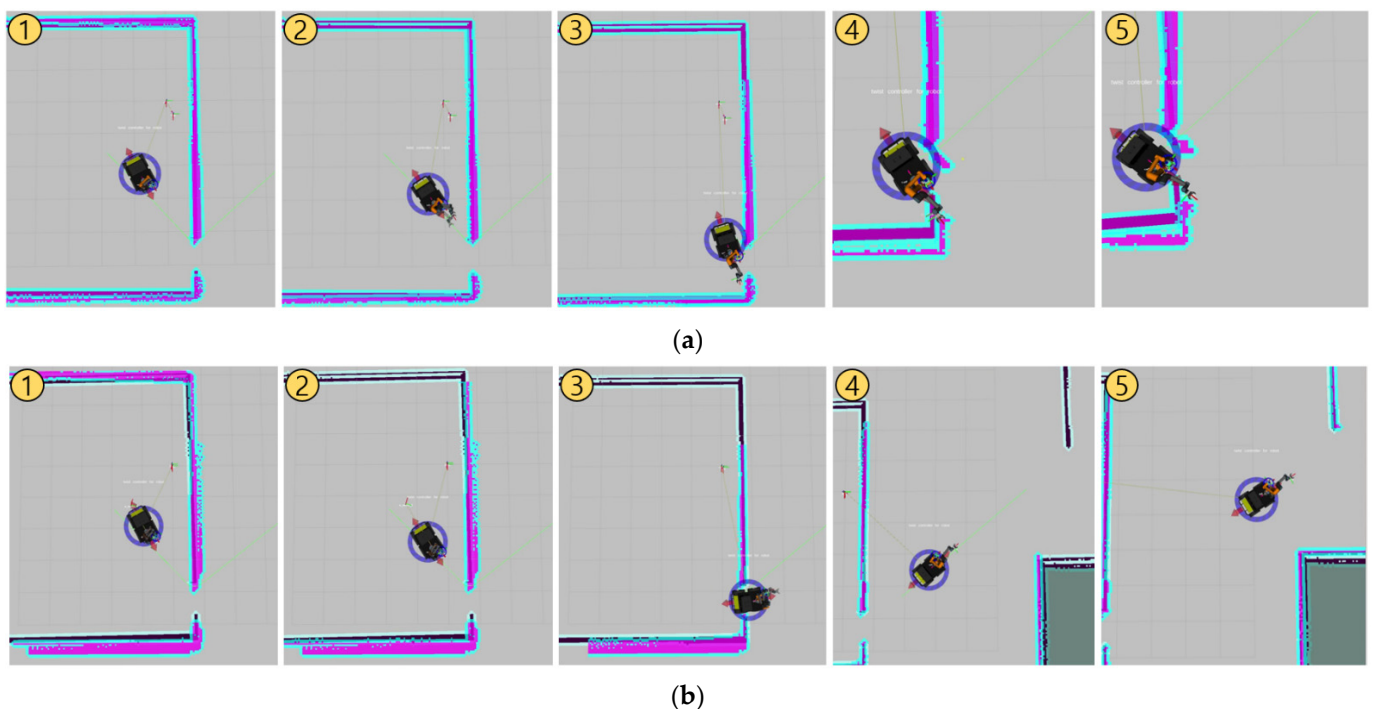


Figure 13. Both snapshots (a,b) show the process of motion planning from Target 2 to Target 3 in Test world #2 using Husky_UR5. (a) uses only move_base and MoveIt, and (b) uses the proposed NUMMIC controller.

In addition, although the tendency for the result values of each controller is maintained, the case with Jackal UGV always shows less duration and greater error compared with Husky UGV. This result seems to be because Jackal UGV itself has a higher maximum speed than Husky UGV, and the linear and angular accelerations in [48,49] GitHub repository which are referred to in this paper are set according to this property.

5.3. Experiment #3: Real Robot in a Real Environment

Lastly, we conducted Experiment #3 with a real Husky_UR3 robot to check whether NUMMIC works in real-world environment without any problems. The hardware specification of Husky_UR3 and the NUMMIC's parameter values are identical to the ones from the previous simulation. The map file of an actual environment is shown in Figure 14. This experiment is composed of two scenarios. In the first scenario, T01_Target [3, 0, 0.533] is designated as target coordinates based on the T01_Start point and the motion planning with NUMMIC is executed. Four motion planning experiments are repeated for the same point, and after each planning is completed, a laser point is applied to the end-effector to lower the foot of the perpendicular to measure which point is manipulated. In the second scenario, T02_Target [2.795, -7, 0.587] is set as target coordinates based on the T02_Start point and the motion planning with NUMMIC is done. In this scenario, the mobile manipulator must turn a corner before motion planning the target because of the structure of the hallway. In the same manner, four motion plannings are repeated for the same target, and the error between the position of end-effector and target coordinates is measured by the foot of the perpendicular in which a laser point is applied to the end-effector.

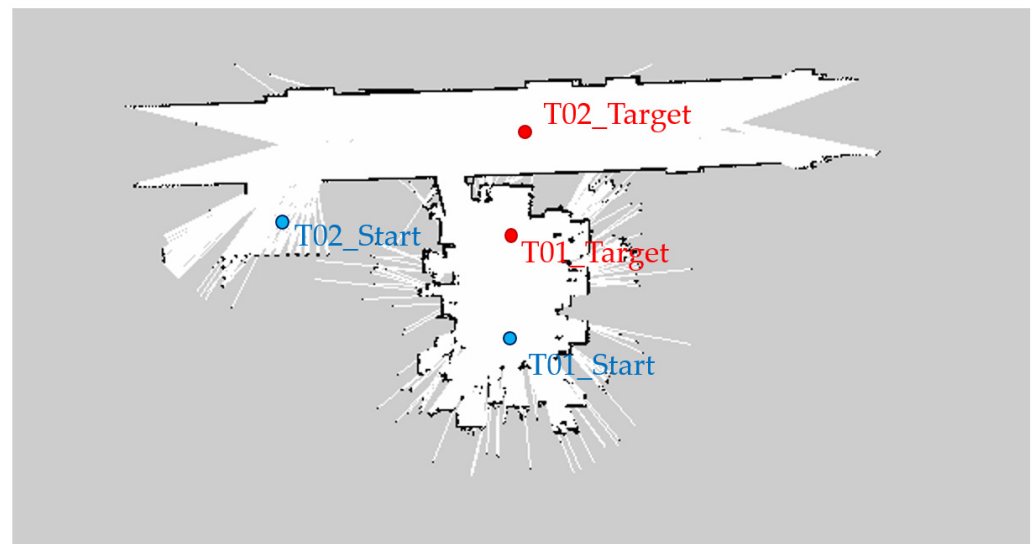


Figure 14. Map file of a real environment for the experiment. Each blue point, named T01_Start and T02_Start, represents the starting point of the mobile manipulator in the first and second scenarios. Each red point, named T01_Start and T02_Start, represents the target point of the mobile manipulator in the first and second scenarios.

Figure 15 shows the motion planning from T01_Start to T01_Target point sequentially. Since there is no collision risk zone on the path to the target, manipulation is performed at the same time as driving (Second scene of Figure 15). After that, when it enters within a specific distance from the target point, the motion planning is successfully performed by the distance proportional speed control using sigmoid function and stopping at the MEP. Figure 16 depicts motion planning from T02_Start to T02_Target point. In this scenario, there is a high-curvature point on the path, which is an area at risk of collision, thus before passing the section, the manipulator maintains the default pose (second scene of Figure 16) and the moment it passes the section, it starts manipulation (third scene of Figure 16).

In this way, the NUMMIC operates as designed, but in terms of accuracy, the result is clearly worse than that of the simulation. Table 7 compares the average values of the error between the target coordinates and the mobile manipulator's end-effector position after finishing each motion planning experiments: T01 and T02. In an actual experiment, since there is an uneven floor or an error on the manipulation that does not appear in the simulation, the error in the z-axis occurred even if it is not large compared to that in the x-y

plane. Therefore, errors are also stated based on all x , y , z axes and Euclidean distance in three dimensions.



Figure 15. Snapshots of the experiment in the first scenario at real environment.



Figure 16. Snapshots of the experiment in the second scenario at real environment.

Table 7. The error values of end-effector position after motion planning in a real environment. T01 is the result of the first scenario, and T02 is the result of the second scenario.

Average Error (cm)	x	y	z	Euclidean Distance
T01	2.3	2.1	0.075	3.62
T02	3.08	8.05	0.15	9.2

However, similar to the previous simulation, this result is estimated to have accumulated larger errors in the navigation process rather than derived from NUMMIC itself. In fact, prior experiments show that a significant number of errors are caused by problems from the navigation process. Since more varied non-systematic errors are easier to intervene in in real environments, the increase in errors in real environments compared to the simulation is a phenomenon that occurs in most navigation-related experiments. In the experiment of T02, which has particularly larger errors than in the simulation, most of the causes seem to have been due to the localization process between the map and the real environment, and this can be estimated in relation to the direction in which the error is large. T01 shows a relatively uniform error in the x -axis and y -axis, but the error of the y -axis in T02 is much larger than that of the x -axis. This is because in the T02 scenario the robot drives on a straight hallway and this environment lacks a point to grasp the feature, which creates a large error in the direction due to the characteristics of the localization algorithm. Thus, we can conclude that although NUMMIC operates in the same way as the simulation, due to the characteristic of the real environment, a larger error is accumulated in the navigation stage, which focuses on localization, and the manipulation stage. We expect

that the errors in the entire motion planning process can also be overcome by installing additional sensors or algorithms to reduce errors from the localization stage.

6. Conclusions

In this paper, we proposed the NUMMIC for the mobile manipulator robot, which is able to execute in a stable manner motion planning in various environments with simultaneous control and has more time benefits than the traditional sequential control method. Here, the various environments mean that the mobile manipulator can work under various surroundings including obstacles, while also meaning that the controller can be used in different types of mobile manipulator model. What makes it possible are the structural characteristics of NUMMIC, which is composed of the designation of the MEP that selects a stable position for manipulation at different environments, the designation of the default pose position of the manipulator to help manipulation efficiently, and the safety check for the manipulator during the motion planning toward target. These three aspects control the mobile platform and the manipulator arm by tuning the `move_base` and `MoveIt` packages. Thus, any mobile manipulator platform can use this controller only if it is a mobile manipulator, the mobile platform is attached to a single manipulator arm with LiDAR and IMU sensor, and it has a URDF about the robot. This controller was tested in simulations by two different maps and three mobile manipulator platforms which have different specifications and confirmed that it operated in any environment in a stable manner. Also, in the test with real robots, the overall operation was similar to the simulation results, even though there were some errors caused by the localization process.

In the future, we will focus on reducing errors in motion planning which are inferred from the localization process and the grasping algorithm. For those, we suppose that the process for obtaining data about the grasped object using an additional camera on the wrist near the manipulator's end effector is needed. Using NUMMIC, it will be able to reduce the error in motion planning with the method continuously checking whether it can grasp the target object at current MEP or not when the camera captures the target object while the robot moves to MEP for grasping. Also, if the grasping algorithm is added to the final process of NUMMIC using the data, this simultaneous controller for mobile manipulator can be used instantly for work such as pick and place in various environments.

Author Contributions: Conceptualization, T.K.; methodology, T.K. and M.K.; software, T.K.; validation, T.K. and D.K.; formal analysis, T.K.; investigation, T.K. and S.Y.; resources, T.K., M.K. and S.Y.; writing—original draft preparation, T.K.; writing—review and editing, T.K., M.K., S.Y. and D.K.; visualization, T.K.; supervision, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Trade, Industry and Energy (MOTIE), South Korea, under the Industrial Technology Innovation Program under Grant 20004315, 20009008 and the Korea Institute for Advancement of Technology, Grant P0017033 and the BK21 plus program through the National Research Foundation (NRF) funded by the Ministry of Education of Korea (No.5120200313836) and partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.RS-2022-00150000, Artificial Intelligence Convergence Innovation Human Resources Development (Kyung Hee University)).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank the anonymous reviewers and academic editor for their comments and suggestions. T.K. would like to thank his brother, Gyeongmin, for his support and help.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Belanche, D.; Casaló, L.V.; Flavián, C.; Schepers, J. Service robot implementation: A theoretical framework and research agenda. *Serv. Ind. J.* **2020**, *40*, 203–205. [CrossRef]
2. Jae-Bong, Y.; Seung-Joon, Y. Mobile Manipulation for the HSR Intelligent Home Service Robot. In Proceedings of the 16th International Conference on Ubiquitous Robots (UR), Jeju, Korea, 24–27 June 2019.
3. Chen, F.; Selvaggio, M.; Caldwell, G.D. Dexterous Grasping by Manipulability Selection for Mobile Manipulator with Visual Guidance. *IEEE Trans. Ind. Inform.* **2019**, *15*, 1202–1210. [CrossRef]
4. Zhou, K.; Ebenhofer, G.; Eitzinger, C.; Zimmermann, U.; Walter, C.; Saenz, J.; Castaño, P.L.; Hernández, A.F.M.; Oriol, N.J. Mobile Manipulator Is Coming to Aerospace Manufacturing Industry. In Proceedings of the IEEE International Symposium on Robotic and Sensors Environments (ROSE), Timisoara, Romania, 16–18 October 2014.
5. Andaluz, V.H.; Sásig, E.R.; Chicaiza, W.D.; Velasco, P.M. Linear Algebra Applied to Kinematic Control of Mobile Manipulators. *IT Converg. Secur.* **2017**, *449*, 297–306.
6. Mashali, M.; Wu, L.; Alqasemi, R.; Dubey, R. Controlling a Non-Holonomic Mobile Manipulator in a Constrained Floor Space. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018.
7. Zhou, S.; Yazhini, C.P.; Peter, C.Y.C. Simultaneous Base and End-Effector Motion Control of a Nonholonomic Mobile Manipulator. In Proceedings of the 6th International Conference on Automation, Robotics and Applications, Queenstown, New Zealand, 17–19 February 2015.
8. Sandakalum, T.; Ang, M.H., Jr. Motion Planning for Mobile Manipulators—A Systematic Review. *Machines* **2022**, *10*, 97. [CrossRef]
9. Sucas, I.; Kay, J. Urdf. Available online: <https://wiki.ros.org/urdf> (accessed on 25 May 2022).
10. Garage, W.; Stanford Artificial Intelligence Laboratory; Open Robotics. Robotic Operating System. Available online: www.ros.org (accessed on 6 December 2021).
11. Marder-Eppstein, E. Move_Base. Available online: https://wiki.ros.org/move_base (accessed on 6 December 2021).
12. Loan, A.S.; Chitta, S. MoveIt. Available online: <https://moveit.ros.org> (accessed on 6 December 2021).
13. Universal Robot Ur3. Available online: <https://www.universal-robots.com/products/ur3-robot/> (accessed on 2 May 2022).
14. Husky Unmanned Ground Vehicle. Available online: <https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/> (accessed on 2 May 2022).
15. Universal Robot Ur5. Available online: <https://www.universal-robots.com/products/ur5-robot/> (accessed on 2 May 2022).
16. Discover Our Gen3 Lite Robot. Available online: www.kinovarobotics.com/product/gen3-lite-robots (accessed on 23 June 2022).
17. Jackal Unmanned Ground Vehicle. Available online: <https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/> (accessed on 23 June 2022).
18. Ventator, E.; Lee, G.S.; Newman, W. Hardware and software architecture of ABBY: An industrial mobile manipulator. In Proceedings of the 2013 IEEE International Conference on Automation Science and Engineering (CASE), Madison, WI, USA, 17–20 August 2013.
19. Meng, J.; Wang, S.; Li, G.; Jiang, L.; Zhang, X.; Lui, C.; Xie, Y. Iterative-learning error compensation for autonomous parking of mobile manipulator in harsh industrial environment. *Robot. Comput.-Integr. Manuf.* **2021**, *68*, 102077. [CrossRef]
20. Štibinger, P.; Broughton, G.; Majer, F.; Rozsypálek, Z.; Wang, A.; Jindal, K.; Zhou, A.; Thakur, D. Mobile Manipulator for Autonomous Localization, Grasping and Precise Placement of Construction Material in a Semi-Structured Environment. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2595–2602. [CrossRef]
21. Vatauvuk, I.; Vasiljević, G.; Kovačić, Z. Task Space Model Predictive Control for Vineyard Spraying with a Mobile Manipulator. *Agriculture* **2022**, *12*, 381. [CrossRef]
22. Colucci, G.; Botta, A.; Tagliavini, L.; Cavallone, P.; Baglieri, L.; Quaglia, G. Kinematic Modeling and Motion Planning of the Mobile Manipulator Agri.Q for Precision Agriculture. *Machines* **2022**, *10*, 321. [CrossRef]
23. Vineet, S.; Deshmukh, D.; Pratihari, D.K.; Deb, A.K.; Ray, H.; Bhattacharyya, H. Dynamic Analysis of Tracked Mobile Manipulator Used in Agriculture. In Proceedings of the 2021 IEEE 18th India Council International Conference, Guwahati, India, 19–21 December 2021.
24. Naazare, M.; Rosas, F.G.; Schulz, D. Online Next-Best-View Planner for 3D-Exploration and Inspection With a Mobile Manipulator Robot. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3779–3786. [CrossRef]
25. Colucci, G.; Tagliavini, L.; Carbonari, L.; Cavallone, P.; Botta, A.; Quaglia, G. Paquitop.arm, a Mobile Manipulator for Assessing Emerging Challenges in the COVID-19 Pandemic Scenario. *Robotics* **2021**, *10*, 102. [CrossRef]
26. Li, Z.; Moran, P.; Dong, Q.; Shaw, R.J.; Hauser, K. Development of a tele-nursing mobile manipulator for remote care-giving in quarantine areas. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3581–3586.
27. Akli, I.; Bouzouia, B.; Achour, N. Motion analysis of a mobile manipulator executing pick-up tasks. *Comput. Electr. Eng.* **2015**, *43*, 257–269. [CrossRef]
28. Vazquez-Santiago, K.; Goh, C.F.; Shimada, K. Motion Planning for Kinematically Redundant Mobile Manipulators with Genetic Algorithm, Pose Interpolation, and Inverse Kinematics. In Proceedings of the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), Lyon, France, 23–27 August 2021; pp. 1167–1174.
29. Li, Q.; Mu, Y.; You, Y.; Zhang, Z.; Feng, C. A Hierarchical Motion Planning for Mobile Manipulator. *IEEE Trans. Electr. Electron. Eng.* **2020**, *15*, 1390–1399. [CrossRef]

30. Stulp, F.; Fedrizzi, A.; Mösenlechner, L.; Beetz, M. Learning and Reasoning with Action-Related Places for Robust Mobile Manipulation. *J. Artif. Intell. Res.* **2012**, *43*, 1–42. [[CrossRef](#)]
31. Wang, C.; Zhang, Q.; Tian, Q.; Li, S.; Wang, X.; Lane, D.; Petillot, Y.; Wang, S. Learning Mobile Manipulation through Deep Reinforcement Learning. *Sensors* **2020**, *20*, 939. [[CrossRef](#)] [[PubMed](#)]
32. Moreno, F.-A.; Monroy, J.; Ruiz-Sarmiento, J.-R.; Galindo, C.; Gonzalez-Jimenez, J. Automatic Waypoint Generation to Improve Robot Navigation Through Narrow Spaces. *Sensors* **2020**, *20*, 240. [[CrossRef](#)] [[PubMed](#)]
33. da Silva Lubanco, D.L.; Pichler-Scheder, M.; Schlechter, T. A Novel Frontier-Based Exploration Algorithm for Mobile Robots. In Proceedings of the 2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE), Barcelona, Spain, 12–15 February 2020.
34. Gerkey, B. Gmapping. Available online: <https://wiki.ros.org/gmapping> (accessed on 2 May 2022).
35. Conner, D.C.; Willis, J. Flexible Navigation: Finite state machine-based integrated navigation and control for ROS enabled robots. In Proceedings of the SoutheastCon 2017, Concord, NC, USA, 30 March–2 April 2017; pp. 1–8.
36. Choi, Y.J.; Ramatryana, I.N.A.; Shin, S.Y. Cellular Communication-Based Autonomous UAV Navigation with Obstacle Avoidance for Unknown Indoor Environments. *Int. J. Intell. Eng. Syst.* **2012**, *14*, 344–352. [[CrossRef](#)]
37. Fernandez Carmona, M.; Parekh, T.; Hanheide, M. Making the Case for Human-Aware Navigation in Warehouses. In *Towards Autonomous Robotic Systems*; Lecture Notes in Computer Science; Althoefer, K., Konstantinova, J., Zhang, K., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 449–453.
38. Chen, L.; Wei, Z.; Zhao, F.; Tao, T. Development of a virtual teaching pendant system for serial robots based on ROS-I. In Proceedings of the 2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), Ningbo, China, 19–21 November 2017; pp. 720–724.
39. Wang, Z.; Gong, L.; Chen, Q.; Li, Y.; Liu, C.; Huang, Y. Rapid Developing the Simulation and Control Systems for a Multifunctional Autonomous Agricultural Robot with ROS. In *Intelligent Robotics and Applications*; Lecture Notes in Computer Science; Silva, M., Luís Lima, J., Reis, L., Sanfeliu, A., Tardioli, D., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 26–39.
40. Sepúlveda, D.; Fernández, R.; Navas, E.; González-de-Santos, P.; Armada, M. ROS Framework for Perception and Dual-Arm Manipulation in Unstructured Environments. In *Robot 2019: Fourth Iberian Robotics Conference*; Advances in Intelligent Systems and Computing; Silva, M., Luís Lima, J., Reis, L., Sanfeliu, A., Tardioli, D., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 137–147.
41. Roldán, J.J.; Peña-Tapia, E.; Garcia-Aunon, P.; Del Cerro, J.; Barrientos, A. Bringing Adaptive and Immersive Interfaces to Real-World Multi-Robot Scenarios: Application to Surveillance and Intervention in Infrastructures. *IEEE Access* **2019**, *7*, 86319–86335. [[CrossRef](#)]
42. Hershberger, D.; Gossow, D.; Faust, J. RViz. Available online: <https://wiki.ros.org/rviz> (accessed on 6 December 2021).
43. Lu, D. Global_Planner. Available online: https://wiki.ros.org/global_planner (accessed on 10 May 2021).
44. Qualia, T. Nummic. Available online: <https://github.com/QualiaT/nummic> (accessed on 15 July 2022).
45. Konolige, K.; Marder-Eppstein, E. Navfn. Available online: <http://wiki.ros.org/navfn> (accessed on 20 June 2022).
46. Marder-Eppstein, E. Dwa_Local_Planner. Available online: http://wiki.ros.org/dwa_local_planner (accessed on 20 June 2022).
47. Marder-Eppstein, E.; Perko, E. Base_Local_Planner. Available online: http://wiki.ros.org/base_local_platnner (accessed on 20 June 2022).
48. Husky. Available online: <https://github.com/husky/husky> (accessed on 20 June 2022).
49. Jackal. Available online: <https://github.com/jackal/jackal> (accessed on 20 June 2022).
50. Gerkey, B. Amcl. Available online: <https://wiki.ros.org/amcl> (accessed on 20 June 2022).
51. Ioan, A. Sukan and Sachin Chitta, MoveIt Setup Assistant. Available online: http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/setup_assistant/setup_assistant_tutorial.html#moveit-setup-assistant (accessed on 20 June 2022).
52. Smits, R.; Aertbelien, E.; Orocos Developers. Kdl. Available online: <http://wiki.ros.org/kdl> (accessed on 20 June 2022).