sensors

MDPI

Article

Reactive Self-Collision Avoidance for a Differentially Driven Mobile Manipulator

Keunwoo Jang [1], Sanghyun Kim [2] and Jaeheung Park [1,3,*]

[1] Graduate School of Convergence Science and Technology, Seoul National University, Seoul 08826, Korea; jkw0701@snu.ac.kr
[2] Korea Institute of Machinery and Materials (KIMM), Daejeon 34103, Korea; kim87@kimm.re.kr
[3] Advanced Institute of Convergence Technology (AICT), Suwon 16229, Korea
* Correspondence: park73@snu.ac.kr

Abstract: This paper introduces a reactive self-collision avoidance algorithm for differentially driven mobile manipulators. The proposed method mainly focuses on self-collision between a manipulator and the mobile robot. We introduce the concept of a distance buffer border (DBB), which is a 3D curved surface enclosing a buffer region of the mobile robot. The region has the thickness equal to buffer distance. When the distance between the manipulator and mobile robot is less than the buffer distance, which means the manipulator lies inside the buffer region of the mobile robot, the proposed strategy is to move the mobile robot away from the manipulator in order for the manipulator to be placed outside the border of the region, the DBB. The strategy is achieved by exerting force on the mobile robot. Therefore, the manipulator can avoid self-collision with the mobile robot without modifying the predefined motion of the manipulator in a world Cartesian coordinate frame. In particular, the direction of the force is determined by considering the non-holonomic constraint of the differentially driven mobile robot. Additionally, the reachability of the manipulator is considered to arrive at a configuration in which the manipulator can be more maneuverable. In this respect, the proposed algorithm has a distinct advantage over existing avoidance methods that do not consider the non-holonomic constraint of the mobile robot and push links away from each other without considering the workspace. To realize the desired force and resulting torque, an avoidance task is constructed by converting them into the accelerations of the mobile robot. The avoidance task is smoothly inserted with a top priority into the controller based on hierarchical quadratic programming. The proposed algorithm was implemented on a differentially driven mobile robot with a 7-DOFs robotic arm and its performance was demonstrated in various experimental scenarios.

Keywords: self-collision avoidance; whole-body motion planning and control; mobile manipulation

check for updates

Citation: Jang, K.; Kim, S.; Park, J. Reactive Self-Collision Avoidance for a Differentially Driven Mobile Manipulator. *Sensors* **2021**, *21*, 890. https://doi.org/10.3390/s21030890

Academic Editor: Carlos J. Pérez Del Pulgar
Received: 6 January 2021
Accepted: 25 January 2021
Published: 28 January 2021

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

Copyright: © 2021 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

A mobile manipulator, which is a manipulator mounted on a mobile robot, has infinite workspace based on the mobility offered by the mobile robot. Furthermore, the degrees of freedom (DOFs) of the mobile robot typically provide the mobile manipulator with redundancy with respect to the tasks such as end-effector trajectory tracking. By utilizing these properties, a mobile manipulator can perform complex and diverse tasks such as painting [1], production [2], and manufacturing [3]. However, to perform these complex tasks in dynamic and unstructured environments, one of the most critical capabilities for a mobile manipulator is to detect self-collision by using proprioceptive or exteroceptive sensors and reactively generate sensor-based avoidance motion. In this paper, we present a new algorithm for reactive self-collision avoidance for a differentially driven mobile manipulator.

1.1. Related Works

Self-collision can be avoided using either offline or online motion generation. Planning collision-free motion is typically implemented offline, whereas online motion generation is mainly embedded with the controllers. First, in the field of motion planning, Kuffner et al. [4] proposed a motion planning algorithm to compute dynamically stable and collision-free trajectories for humanoid robots based on Rapidly-exploring Random Trees (RRT). Oriolo and Mongillo [5] proposed a randomized planner that resolves the redundancy of non-holonomic mobile manipulator. The planner allows a mobile robot to be located within a compatible region for a given end-effector position so that the inverse kinematics solutions for the manipulator can be derived. Regarding pose constraints on the end-effector, Berenson et al. [6] developed Constrained Bi-directional RRT (CBiRRT) that plans the trajectory by projecting sampled position onto Task Space Region (TSR). Burget et al. [7] proposed a planning framework called Bi-directional Informed RRT*(BI^{2}RRT*) that can efficiently obtain optimal paths for mobile manipulation under task space constraints. Furthermore, Welschehold et al. [8] proposed a motion planner for the mobile manipulator, which exploited the concept of dynamical system approach for obstacle avoidance and the concept of inverse reachability for approximating the inner and outer boundaries of workspace. Kang et al. [9] proposed a sampling-based method that efficiently explores whole-body configuration space by sampling more around a region close to obstacles. However, these methods are difficult to implement in unstructured and dynamic environments because trajectories may have to be regenerated in real time. Additionally, their computational cost is relatively high for robots with large number of DOFs, such as humanoids or mobile manipulators.

To overcome these limitations, many reactive methods have been proposed to detect and avoid self-collision in real time. Our proposed method belongs to this category. Seto et al. [10,11] designed the outer parts of links as elastic elements so that the reaction forces are generated between elastic elements when links move close to each other. In [12,13], motions for self-collision avoidance were generated based on the gradient of a cost function related to the distances between links. Dariush et al. [14] penalized joint motions using the inverse matrix of weighted Jacobian based on the gradient of a collision function. Fang et al. [15] proposed a method for generating relative motion between the links using an inequality task. Quiroz-Omaña et al. [16] designed a distance function and converted it to the form of inequality constraint to generate relative acceleration. However, these methods are not applicable to non-holonomic mobile manipulators because they were developed for holonomic systems. Specifically, the methods may not instantaneously generate motion of the mobile robot in a certain direction because the non-holonomic constraint is not considered [17,18]. On the other hand, Dietrich et al. [19,20] proposed a repulsive force-based approach with an efficient damping design and extended continuous null space projection method. Sugiura et al. [21] proposed a method using only 1-DOF repulsive force while dynamically swapping the priority of the tasks. Gonon et al. [22] proposed a modified artificial potential field that includes a viscous damping term to dissipate energy and enforces a limit on the repulsive forces, thus can prevent the repulsive force from oscillating with high frequency. Although repulsive force-based methods are conservative and effective solution for avoiding collisions, the methods repel two proximate links and thus the modified motion may be farther away from the reference motion more than necessary. Lei et al. [23] proposed a method for humanoid dual-arm robot that generates smooth repulsive velocity based on designing the links as discretized spheres. However, the method does not consider self-collision between the manipulator and mobile robot.

1.2. Overview of This Paper

In this paper, we propose a new self-collision avoidance algorithm for a differentially driven mobile manipulator. Our focus is on avoiding self-collision between a manipulator and mobile robot. Our goal is to generate a motion in which the manipulator can avoid self-collision without modifying its reference motion. To this end, we propose the concept

of the DBB, a border of the buffer region surrounding the mobile robot. This region has a thickness equal to the buffer distance (see Figure 1a). When the manipulator and the mobile robot are close to each other, meaning their distance is less than the buffer distance (see Figure 1b), our strategy is to position the manipulator outside the DBB by moving the mobile robot. This can be accomplished by generating a force exerted on the mobile robot because the DBB is attached to the mobile robot and moves with it (see Figure 1c). Therefore, the manipulator can avoid self-collision with the mobile robot without modifying reference motion of the manipulator (see Figure 1d).

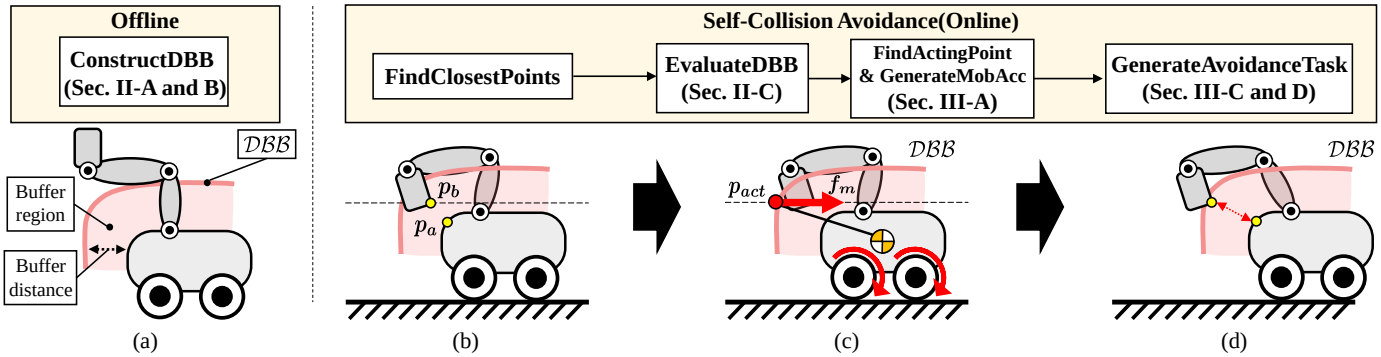


Figure 1. Overview of the proposed algorithm. (**a**) For each link colliding with the mobile robot, the DBB is defined as the boundary of the buffer region enclosing the mobile robot. (**b**) For example, when the distance between the manipulator and mobile robot is less than the buffer distance, the two closest points ($\boldsymbol{p_a}$, $\boldsymbol{p_b} \in \mathbb{R}^3$) are calculated. (**c**) To avoid self-collision, our strategy is to exert a force $\boldsymbol{f_m} \in \mathbb{R}^3$ on the mobile robot to ensure that the manipulator lies outside the DBB. To accomplish this goal, a point on the DBB is selected as the acting point $\boldsymbol{p_{act}} \in \mathbb{R}^3$ for the force after evaluating the 3D points on the DBB as scores. The target direction of the force is toward the closest point $\boldsymbol{p_b}$ on the manipulator. (**d**) To realize the desired force and resulting torque, we construct the avoidance task by converting them to the accelerations for the mobile robot. As the task is continuously inserted into the controller, self-collision between the manipulator and mobile robot can be avoided.

Especially, the direction of force is determined by considering the following two factors. First, the singularity of the differentially driven mobile robot due to non-holonomic constraint is considered in order not to lose the controllability of the mobile robot. Second, we consider the reachability of the manipulator, which is a representation of the robot's workspace with information regarding pose quality. Because the direction of force is determined to enhance the reachability, the resultant configuration of the robot can secure a large workspace of the manipulator.

To implement the proposed algorithm on a robot, an avoidance task is constructed by combining two types of motions depending on whether a link collides with the mobile robot or not. First, for a link pair including the mobile robot, the desired force and resulting torque are generated and converted into accelerations of the mobile robot. Second, for a link pair that does not includes the mobile robot, 1-DOF acceleration is generated in the direction in which that the distance between the closest points of the link pair increases. The avoidance task is then constructed by stacking two types of accelerations and their Jacobian matrices. The task is inserted continuously with the highest priority depending on the distances between the link pairs by using a controller based on Hierarchical Quadratic Programming (HQP) with the continuous task transition algorithm [24,25]. The HQP can handle multiple tasks with strict priorities, Stack of Tasks (SoT).

The remainder of this paper is organized as follows. First, Section 2 details the DBB and the computation of its score for deriving the desired force. Second, Section 3 explains our overall strategy for self-collision avoidance. Next, Section 4 describes the experimental validation of the proposed strategy. Finally, the paper is concluded in Section 5. To enhance readability of this paper, Table 1 lists the symbols used in this paper and their corresponding definitions. Bold Roman letters denote vectors and matrices while normal Roman letters denote real numbers.

Table 1. Notation and symbols.

Symbol	Description
$< l_a, l_b >$	Pair of the links a and b
$\mathcal{L} = \mathcal{L}_m \cup \mathcal{L}_m^c$	Set of potentially colliding link pairs
$\mathcal{L}_m$	Subset of $\mathcal{L}$ including the mobile robot
$\mathcal{L}_m^c$	Complement set of $\mathcal{L}_m$
$\mathcal{DBB}_i$	DBB of i-th link pair in $\mathcal{L}_m$
n	DOFs of the mobile manipulator
n_m	DOFs of the manipulator
$\mathcal{T}_j$	j-th equality or inequality task
$\mathcal{T}_j \prec \mathcal{T}_{j+1}$	$\mathcal{T}_j$ has higher priority than $\mathcal{T}_{j+1}$

2. Distance Buffer Border and Its Score Computation

This section introduces the concept of the DBB for generating a force that avoids self-collision between a manipulator and mobile robot. Additionally, we describe how to compute the score of the DBB to determine the direction and magnitude of the force. In Section 2.1, all link pairs that can potentially collide with each other are identified based on the collision model and kinematic information of the robot. Next, for the link pairs including the mobile robot, we define the DBB in Section 2.2. Finally, two factors are introduced to calculate the scores of the points on the DBB in Section 2.3.

2.1. Identification of Potentially Colliding Link Pairs

To decrease the computational cost for checking self-collision, simplified collision models are designed by using the convex shapes based on the kinematic structure of the manipulator. Figure 2a,b shows the kinematic structure of our robot and collision models of the robot, respectively. Utilizing these models and the joint position ranges of the manipulator, the link pairs which never collide with each other can be precomputed by randomly sampling the joint position of the manipulator. From this analysis, the link pairs potentially colliding with each other are identified. The set of the link pairs is defined as follows (note that l_2 never collides with l_m because the range of the 2nd joint position of the manipulator is physically limited from -101 deg to 101 deg).

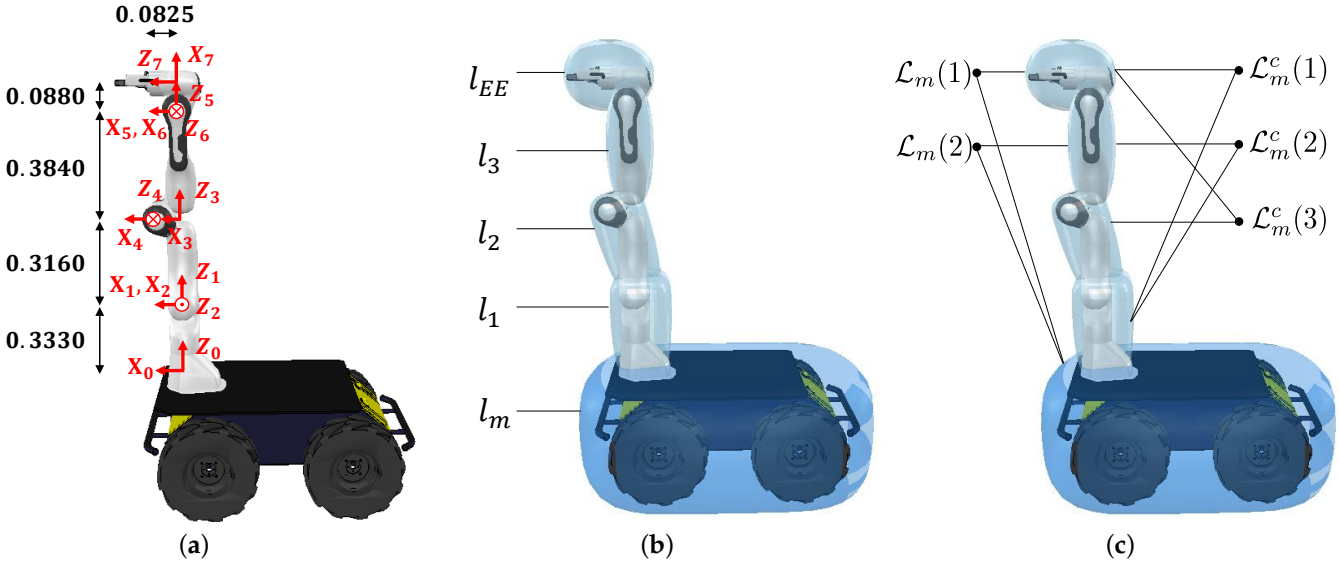


Figure 2. Our mobile manipulator system consists of a four-wheel differentially driven mobile robot called Husky (Clearpath Robotics. Co.) and 7-DOFs manipulator called Panda (Franka Emika. Co.). (**a**) Kinematic structure of the manipulator is shown with the scale of meter; (**b**) the simplified collision models of the robot consist of five links; (**c**) based on the collision models and joint range of the manipulator, all link pairs that potentially collide with each other are identified. $\mathcal{L}_m(i)$ denotes the link pair including the mobile robot, whereas $\mathcal{L}_m^c(j)$ denotes the link pair not including the mobile robot.

$$\mathcal{L} = \{< l_m, l_{EE} >, < l_m, l_3 >, < l_1, l_{EE} >, < l_1, l_3 >, < l_2, l_{EE} >\}, \quad (1)$$

where $l_\bullet$ denotes an individual link in Figure 2b. Because the avoidance motion is generated differently depending on whether or not the link pair includes the mobile robot l_m, the set $\mathcal{L}$ is divided into two subsets, namely $\mathcal{L}_m$ and its complement $\mathcal{L}_m^c$, and shown in Figure 2c as follows.

$$\begin{aligned} \mathcal{L}_m &= \{< l_m, l_{EE} >, < l_m, l_3 >\} \\ \mathcal{L}_m^c &= \{< l_1, l_{EE} >, < l_1, l_3 >, < l_2, l_{EE} >\}. \end{aligned} \quad (2)$$

In (2), we denote each element of subset as $\mathcal{L}_m(i)$ and $\mathcal{L}_m^c(j)$ respectively. Therefore, to avoid self-collision, we generate a force on the mobile robot for $\mathcal{L}_m$ as discussed in Section 3.1 and 1-DOF repulsive acceleration for $\mathcal{L}_m^c$ as discussed in Section 3.2.

2.2. Distance Buffer Border

Our avoidance strategy is to move the mobile robot in order to place the manipulator outside a region surrounding the mobile robot with a thickness equal to the buffer distance. To this end, we define a border of the region as the DBB of $\mathcal{L}_m$. Geometrically, the DBB represents a group of 3D points located away from the mobile robot l_m by the buffer distance.

Algorithm 1 describes the construction of the DBB in detail and is implemented offline. The input for the algorithm is the set of stored points on the manipulator's link for $\mathcal{L}_m$. For each link pair in $\mathcal{L}_m$, the two closest points are calculated after randomly sampling the joint positions of the manipulator. The point on the link of the manipulator is then stored to the set. This process repeats until the set contains a sufficient number of points. For the i-th link pair in $\mathcal{L}_m$, each set is denoted by $\mathcal{P}_i$ as shown in Figure 3. The algorithm operates as follows.

Algorithm 1 ConstructDBB

Input: $\mathcal{P}_i$: a set of points on the manipulator's link for $\mathcal{L}_m(i)$
Output: $\mathcal{DBB}_i$: distance buffer border of $\mathcal{L}_m(i)$

```
for each p_i in P_i do
    d_m ← DistanceToMobile(p_i, l_m)
    if ||d_m − d_b|| ≤ ε then
        Store p_i in DBB_i
    end if
end for
```

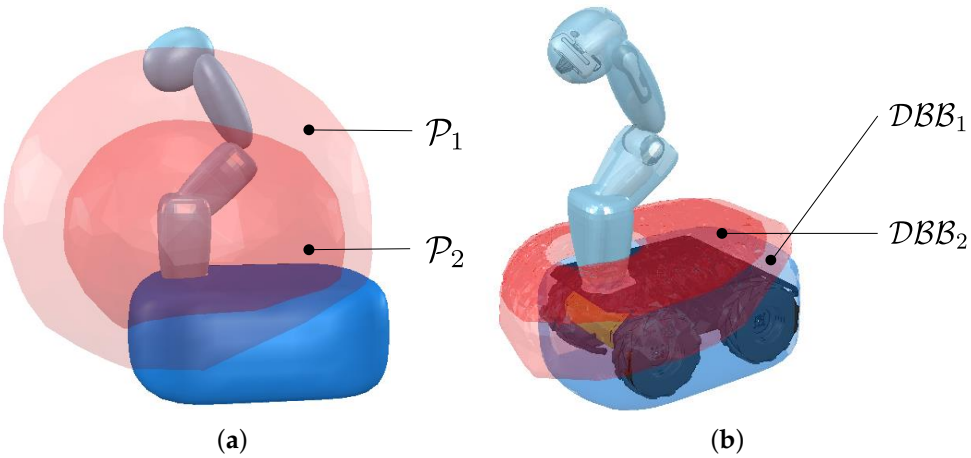


Figure 3. Visualization of $\mathcal{P}_i$ and $\mathcal{DBB}_i$ for $\mathcal{L}_m(i)$. (**a**) Red volume represents $\mathcal{P}_i$ which is a point set around the mobile robot; (**b**) red hyperplanes represent the distance buffer borders of $\mathcal{L}_m$. The buffer distance d_b is set to 0.15 m and the tolerance ϵ is set to 0.01 m.

First, for each point $\boldsymbol{p_i}$ of $\mathcal{P}_i$, the DistanceToMobile function calculates the distance between $\boldsymbol{p_i}$ and the mobile robot (see Line 2). Second, if the distance is within a bounded range, then the point $\boldsymbol{p_i}$ is stored in the DBB (see Line 3–4). After these procedures are repeated, the DBB is constructed as shown in Figure 3b and defined as follows:

$$\begin{aligned} \mathcal{DBB}_i &\ni {}^{\forall}\boldsymbol{p_i} \\ \text{s. t.} \quad &\|d_m - d_b\| \leq \epsilon, \boldsymbol{p_i} \in \mathcal{P}_i \end{aligned} \tag{3}$$

where $\mathcal{DBB}_i$ denotes the DBB for $\mathcal{L}_m(i)$, $\boldsymbol{p_i} \in \mathbb{R}^3$ is the position on the link of the manipulator, d_m is the minimum distance between $\boldsymbol{p_i}$ and l_m, d_b is buffer distance, and ϵ is tolerance value. Even though Figure 3 shows the DBB for our robot, the DBB for other differentially driven mobile manipulator can be obtained if kinematic structure and collision models of the robot are given. To accomplish our strategy, we generate a force on the mobile robot based on the DBB. The direction of the force is defined to begin at a point on the DBB and head toward a point on the manipulator. The point on the DBB becomes the acting point of the force as shown in Figure 1c. In the following subsection, we propose a score for evaluating every point on the DBB to select the acting point.

2.3. Evaluation of Distance Buffer Border

To select the point on the DBB that satisfies the desired capabilities of the force, the DBB is evaluated based on a score consisting of two factors: the singularity of the differentially driven mobile robot and the reachability of the manipulator.

2.3.1. Singularity of the Differentially Driven Mobile Robot

First, the singularity of the differentially driven mobile robot is considered to prevent the force from generating the motion of the non-holonomic mobile robot along the singular direction. Figure 4 shows the kinematic modeling of two-wheel differentially driven mobile robot which simplifies that of four-wheel differentially driven mobile robot [26]. The differentially driven mobile robot is subject to a constraint in terms of the velocity as follows.

$$-\dot{x}_o sin(\phi) + \dot{y}_o cos(\phi) = 0 \tag{4}$$

where $\dot{x}_o$ and $\dot{y}_o$ are planar velocity of the center of the mobile robot and ϕ is the heading angle of the robot from the X-axis in the world frame as shown in Figure 4. Physically, (4) means that there is no velocity component parallel to the wheel-axis at the center of the differentially driven mobile robot. The constraint is non-integrable, thus termed as non-holonomic constraint [27,28].

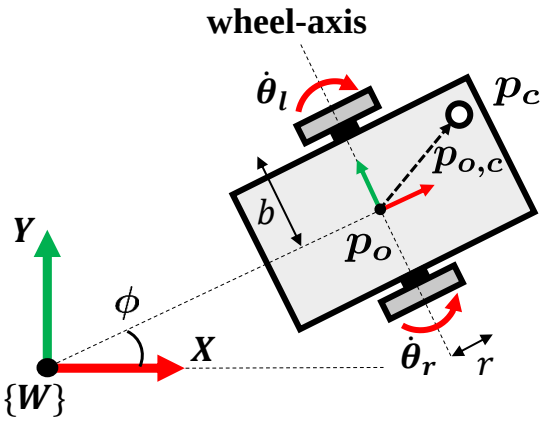


Figure 4. Schematic drawing of the differentially driven mobile robot. p_c is the control point of the mobile robot in the world frame $\{W\}$ and $p_{o,c}$ is the planar vector from the center of the mobile robot, p_o, to p_c in $\{W\}$. b and r are the distance between the wheel and center of the mobile robot and the radius of the wheel, respectively. $\dot{\theta}_r$ and $\dot{\theta}_l$ are the spinning velocities of the right and left wheel, respectively.

The velocity relationship between the control point and the configuration of the differentially driven mobile robot is given by

$$\dot{\boldsymbol{p}}_{\boldsymbol{c}} = \boldsymbol{J}_{\boldsymbol{c}}(\boldsymbol{q}_{\boldsymbol{c}})\dot{\boldsymbol{q}}_{\boldsymbol{b}}, \tag{5}$$

where $\dot{\boldsymbol{p}}_{\boldsymbol{c}} = \begin{bmatrix}\dot{x}_c & \dot{y}_c\end{bmatrix}^T \in \mathbb{R}^2$ is the planar velocity of the control point of the mobile robot. $\dot{\boldsymbol{q}}_{\boldsymbol{b}} = \begin{bmatrix}\dot{\theta}_r & \dot{\theta}_l\end{bmatrix}^T \in \mathbb{R}^2$ is spinning velocity of the wheels and the subscripts r and l of $\dot{\theta}$ denote the right and left wheel, respectively. $\boldsymbol{J}_{\boldsymbol{c}}(\boldsymbol{q}_{\boldsymbol{c}}) \in \mathbb{R}^{2\times 2}$ is Jacobian matrix given by

$$\boldsymbol{J}_{\boldsymbol{c}}(\boldsymbol{q}_{\boldsymbol{c}}) = \begin{bmatrix} c(b\cos\phi - y_{o,c}) & c(b\cos\phi + y_{o,c}) \\ c(b\sin\phi + x_{o,c}) & c(b\sin\phi - x_{o,c}) \end{bmatrix}, \tag{6}$$

where $c = r/2b$, r is the radius of the wheel, b is the distance between the wheel and the center of the mobile robot, and $\boldsymbol{q}_{\boldsymbol{c}} = \begin{bmatrix}\boldsymbol{p}_{\boldsymbol{o,c}} & \phi\end{bmatrix}^T$. $\boldsymbol{p}_{\boldsymbol{o,c}} = \begin{bmatrix}x_{o,c} & y_{o,c}\end{bmatrix}$ are the coordinates of the control point from the center of the mobile robot in the global frame and ϕ is the orientation of the mobile robot.

To identify the singularity, we derive the determinant of the product of the Jacobian matrix $\boldsymbol{J}_{\boldsymbol{c}}(\boldsymbol{q}_{\boldsymbol{c}})$ as

$$det(\boldsymbol{J}_{\boldsymbol{c}}\boldsymbol{J}_{\boldsymbol{c}}^T) = 4b^2c^4(x_{o,c}\cos\phi + y_{o,c}\sin\phi)^2. \tag{7}$$

From (7), the Jacobian matrix $\boldsymbol{J}_{\boldsymbol{c}}(\boldsymbol{q}_{\boldsymbol{c}})$ loses rank when

$$x_{o,c}\cos\phi + y_{o,c}\sin\phi = 0. \tag{8}$$

Geometrically, the left side of (8) represents the distance between the control point and the line of the wheel-axis. As the value of (8) tends to zero, meaning the control point is located on the wheel-axis, the control point of the differential-driven mobile robot cannot instantaneously move along the wheel-axis [29].

Thus, assuming that each point $\boldsymbol{p}_{\boldsymbol{i}}$ of $\mathcal{DBB}_i$ is set to the control point of the mobile robot, we can measure how close it is to the singularity by setting $\boldsymbol{q}_{\boldsymbol{c}} = \begin{bmatrix}x_i & y_i & 0\end{bmatrix}^T$ where x_i and y_i are the coordinates of $\boldsymbol{p}_{\boldsymbol{i}}$ along X-axis and Y-axis, respectively. For our robot, r is set to 0.165 m and b is set to 0.51 m. In Figure 5a, the value of (7) for each point on $\mathcal{DBB}_1$ is computed. As shown in Figure 5a, the determinant value is symmetric about X-axis.

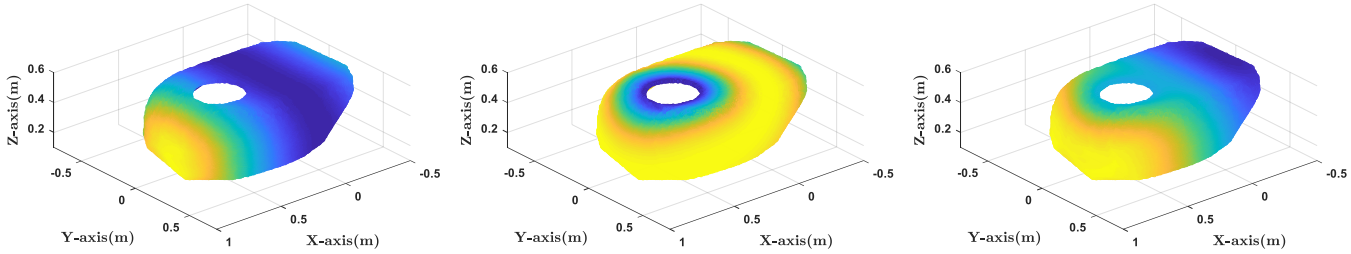


Figure 5. All the points in $\mathcal{DBB}_1$ are colored depending on (**a**) the determinant value in (7), (**b**) the reachability in (9), and (**c**) the score in (10) (yellow: high, blue: low).

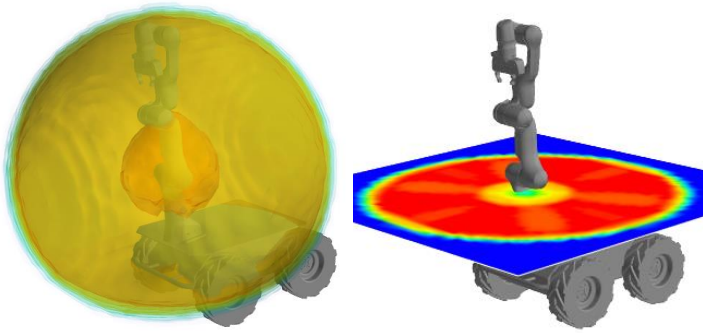
2.3.2. Reachability of the Manipulator

Second, the reachability of the manipulator is considered in order for the force to place the manipulator in the suitable workspace. The reachability is defined as the density of Inverse Kinematics (IK) solutions for the pose of the end-effector [30]. Reachability is computed by uniformly sampling the pose of the end-effector over the entire workspace and recording the number of IK solutions for each pose. The reachability of our robot is illustrated in Figure 6. One can see that the value of reachability increases and then

decreases as the pose of the end-effector moves outwards from the base of the manipulator. Based on this observation, reachability can be expressed as a scalar concave function of the distance from the base of the manipulator. Among the various types of concave functions, the following second-order polynomial function was selected in this paper.

$$\mathcal{R}(\boldsymbol{p_i}) = -A(\|\boldsymbol{p_i} - \boldsymbol{p_{base}}\|_2 - B)^2 + C, \tag{9}$$

where $\mathcal{R}(\boldsymbol{p_i}) : \mathbb{R}^3 \to \mathbb{R}^+$ maps points on the DBB to reachability values, $\boldsymbol{p_{base}} \in \mathbb{R}^3$ is the position of the base of the manipulator, and A, B, C are positive coefficients of the polynomial. Based on the reachability data in Figure 6, we set A to 525.9 /m^2, B to 0.575 m, and C to 100. Figure 5b presents the reachability value for each point on $\mathcal{DBB}_1$. Although reachability is originally defined for the pose of the end-effector, the reachability of other link of the manipulator can also be obtained using kinematics information.



(**a**) (**b**)

Figure 6. Visualization of reachability shown in OPENRAVE [31]. (**a**) The contour of reachability of the end-effector; (**b**) the reachability cut by a horizontal plane at the base of the manipulator is colored (right, red: high, blue: low).

2.3.3. Score of the DBB

We compute a score for every point on the DBB denoted as $\mathcal{S}(\boldsymbol{p_i}) \in \mathbb{R}$. A score is expressed as

$$\mathcal{S}(\boldsymbol{p_i}) = sign(x_i) det(\boldsymbol{J_c} \boldsymbol{J_c}^T) \mathcal{R}(\boldsymbol{p_i}) \tag{10}$$

where

$$sign(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0. \end{cases} \tag{11}$$

Note that the function *sign* ensures that the DBB has a point of global maximum score as shown in Figure 5c.

3. Self-Collision Avoidance Algorithm

In this section, we explain how to avoid self-collision for the differentially driven mobile manipulator. Algorithm 2 details the procedure. First, the FindClosestPoints function calculates the closest pair of points for each link pair. The link pair in $\mathcal{L}_m$ then generates the acceleration of the mobile robot, whereas the acceleration of the manipulator is generated for $\mathcal{L}_m^c$. For the subset $\mathcal{L}_m$, the FindActingPoint function determines the acting point of the force based on the computation for the score of the DBB. Next, the GenerateMobAcc function generates the force and the resulting torque exerted on the mobile robot and converts them to the linear and angular acceleration of the mobile robot as $\underline{\ddot{\boldsymbol{x}}}_{\boldsymbol{m}} \in \mathbb{R}^2$ (see Line 5–6 and Section 3.1). On the other hand, for the subset $\mathcal{L}_m^c$, the GenerateRepAcc function generates a 1-DOF repulsive acceleration which pushes the two proximal links of the manipulator away from each other and stacks the accelerations for k link pairs of $\mathcal{L}_m^c$ as $\underline{\ddot{\boldsymbol{x}}}_{\boldsymbol{r}} \in \mathbb{R}^k$ (see Line 10 and Section 3.2). Then, the GenerateAvoidanceTask function combines these accelerations and constructs the task for avoiding self-collision, $\mathcal{T}_{sca}$, as an

equality task (see Line 13 and Section 3.3). Finally, the UpdateSoT function inserts the task $\mathcal{T}_{sca}$ as a top priority in the original SoT by using the continuous task transition scheme, as summarized in Line 14–15 and Section 3.4. In the following subsections, each function in the Algorithm 2 is described in detail.

Algorithm 2 Self-Collision Avoidance

Input: A set of link pairs $\mathcal{L} = \mathcal{L}_m \cup \mathcal{L}_m^c$; DBBs of the links $\mathcal{DBB}$

1: **while** IsControl() **do**
2: UpdateKinematics($\boldsymbol{q}$)
// Avoidance between mobile robot and manipulator
3: **for** each $\mathcal{L}_m(i)$ **do**
4: $(\boldsymbol{p}_{a,i}, \boldsymbol{p}_{b,i}) \leftarrow$ FindClosestPoints($\mathcal{L}_m(i)$)
// $\boldsymbol{p}_{a,i}$ on mobile robot, $\boldsymbol{p}_{b,i}$ on manipulator
5: $\boldsymbol{p}_{act,i} \leftarrow$ FindActingPoint($\boldsymbol{q}, \boldsymbol{p}_{b,i}, \mathcal{DBB}_i$)
6: $\underline{\ddot{\boldsymbol{x}}}_m \leftarrow$ GenerateMobAcc($\boldsymbol{p}_{act,i}, \boldsymbol{p}_{b,i}$)
7: **end for**
// Avoidance between links of manipulator
8: **for** each $\mathcal{L}_m^c(j)$ **do**
9: $(\boldsymbol{p}_{a,j}, \boldsymbol{p}_{b,j}) \leftarrow$ FindClosestPoints($\mathcal{L}_m^c(j)$)
10: $\underline{\ddot{\boldsymbol{x}}}_r \leftarrow$ GenerateRepAcc($\boldsymbol{p}_{a,j}, \boldsymbol{p}_{b,j}$)
11: **end for**
// Insert the task continuously to the controller
12: **if** $\|\underline{\ddot{\boldsymbol{x}}}_m\|_2 > 0$ **or** $\|\underline{\ddot{\boldsymbol{x}}}_r\|_2 > 0$ **then**
13: $\mathcal{T}_{sca} \leftarrow$ GenerateAvoidanceTask($\underline{\ddot{\boldsymbol{x}}}_m, \underline{\ddot{\boldsymbol{x}}}_r$)
14: $SoT \leftarrow$ UpdateSoT($\mathcal{T}_{sca}$)
15: $\boldsymbol{u} \leftarrow$ HQPSolver(SoT) // See (29)
16: **else**
17: $\boldsymbol{u} \leftarrow$ HQPSolver(SoT) // See (26)
18: **end if**
19: **end while**

3.1. Generation of the Acceleration for the Mobile Robot

This subsection describes the generation of the force. The direction of the force is designed to start from the selected point on the DBB and head to the closest point on the manipulator. Thus, we focus on how to select the acting point of the force that satisfies the following requirements.

First, the acting point should be located with the same height of the closest point on the manipulator because the force should act on a horizontal plane to move the mobile robot. Second, the acting point should be selected so that the force has two orthogonal components that play different roles. As shown in Figure 7a, the direction of the force can be decomposed into two orthogonal directions. One is the direction of the line connecting the closest point on the manipulator and the DBB. The other is its normal direction. The former increases the distance between the mobile robot and the manipulator as the DBB moves closer to the manipulator as shown in Figure 7b. On the other hand, as shown in Figure 7c, the latter places the manipulator closer to the DBB with high score in order to avoid selecting the acting point, i.e., the control point of the mobile robot near the singularity and enhance the reachability of the manipulator. Combining two orthogonal components, self-collision between the manipulator and the mobile robot can be avoided in Figure 7d.

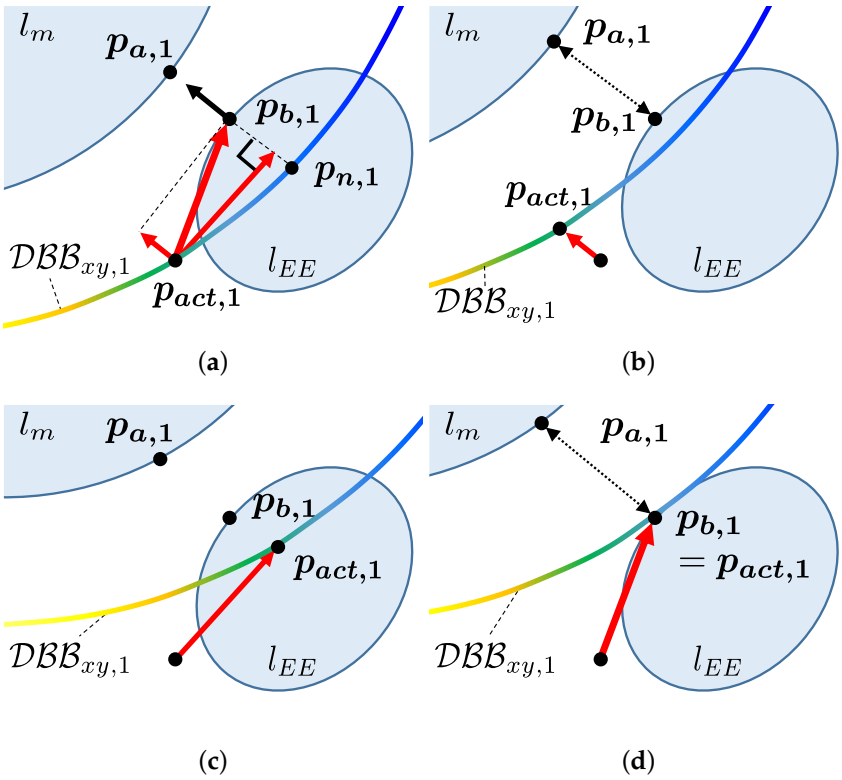


Figure 7. Illustration of the direction of force considering collision between the end-effector l_{EE} and mobile robot l_m. $\mathcal{DBB}_{xy,1}$ denotes the points whose heights are same as that of the closest point $\boldsymbol{p_{b,1}}$ on the end-effector. The points are colored by the score (yellow: high, blue: low). (**a**) The direction of the force starts from the acting point $\boldsymbol{p_{act,1}}$ and points toward $\boldsymbol{p_{b,1}}$. The force can be decomposed into two orthogonal components; (**b**) one of them moves the mobile robot away from the end-effector; (**c**) the other direction places the high-score part of DBB closer to the end-effector; (**d**) by combining these components, self-collision between the end-effector and mobile robot can be avoided.

The FindActingPoint function (see Algorithm 3 and Figure 8) finds the acting point that satisfies these requirements. Algorithm 3 operates as follows. First, the TransformToWorld function transforms the points of $\mathcal{DBB}_i$ to be expressed in the world frame. Next, the FindSameHeight function finds the points in $\mathcal{DBB}_i$ whose height are same as that of the closest point $\boldsymbol{p_{b,i}}$ on the manipulator, which satisfies the first requirement (see Line 1–3). The obtained points are denoted as $\mathcal{DBB}_{xy,i}$. Then, for the second requirement, we calculate the acting point on $\mathcal{DBB}_{xy,i}$ that the generated force can have two orthogonal components. Among the points in $\mathcal{DBB}_{xy,i}$, the point $\boldsymbol{p_{n,i}}$ closest to the point $\boldsymbol{p_{b,i}}$ is identified (see Line 4). Starting at $\boldsymbol{p_{n,i}}$, the position of point $\boldsymbol{p_{t,i}}$ translated along the tangential direction $\boldsymbol{t_i}$ with a step size α_i as follows (see Line 5 and Figure 8a):

$$\begin{aligned} \boldsymbol{p_{t,i}} = \boldsymbol{p_{n,i}} + \alpha_i \left(\frac{\nabla \mathcal{S}(\boldsymbol{p_{n,i}}) \cdot \boldsymbol{t_i}}{\|\nabla \mathcal{S}(\boldsymbol{p_{n,i}})\|}\right) \boldsymbol{t_i}, \\ \boldsymbol{t_i} \perp (\boldsymbol{p_{n,i}} - \boldsymbol{p_{b,i}}), \|\boldsymbol{t_i}\|_2 = 1, \end{aligned} \tag{12}$$

where the inner product of $\nabla \mathcal{S}(\boldsymbol{p_{n,i}})$ and $\boldsymbol{t_i}$ determines the direction of $\boldsymbol{t_i}$ to the higher score of the DBB. The step size α_i is calculated as

$$\alpha_i \propto \frac{d_i}{|\mathcal{S}(\boldsymbol{p_{n,i}})|}. \tag{13}$$

where d_i is the distance of the i-th link pair of $\mathcal{L}_m$. Therefore, the acting point $\boldsymbol{p_{act,i}}$ is calculated as that with the shortest distance from $\boldsymbol{p_{t,i}}$ to $\mathcal{DBB}_{xy,i}$ (see Line 6). In (13), as the distance between the manipulator and mobile robot decreases, the step size decreases to

generate both orthogonal directions of the force. However, a larger step size can be used to proceed more rapidly toward a higher score of the DBB when the distance increases. Additionally, the step size increases as the absolute value of the score of point $\boldsymbol{p_{n,i}}$ decreases to zero, indicating that the point $\boldsymbol{p_{n,i}}$ is near the singularity. To prevent obtaining a step size that is too small or too large as shown in Figure 8b, the step size is bounded by lower and upper limits.

Algorithm 3 FindActingPoint($\boldsymbol{q}$, $\boldsymbol{p_{b,i}}$, $\mathcal{DBB}_i$)

1: $z_{b,i} \leftarrow$ height of $\boldsymbol{p_{b,i}}$
2: $\mathcal{DBB}_i \leftarrow$ TransformToWorld($\boldsymbol{q}$, $\mathcal{DBB}_i$)
3: $\mathcal{DBB}_{xy,i} \leftarrow$ FindSameHeight($z_{b,i}$, $\mathcal{DBB}_i$)
4: $\boldsymbol{p_{n,i}} \leftarrow$ FindMinDistancePoint($\mathcal{DBB}_{xy,i}$, $\boldsymbol{p_{b,i}}$)
5: $\boldsymbol{p_{t,i}} \leftarrow$ Equation (12) and (13)
6: $\boldsymbol{p_{act,i}} \leftarrow$ FindMinDistancePoint($\mathcal{DBB}_{xy,i}$, $\boldsymbol{p_{t,i}}$)
7: **return** $\boldsymbol{p_{act,i}}$

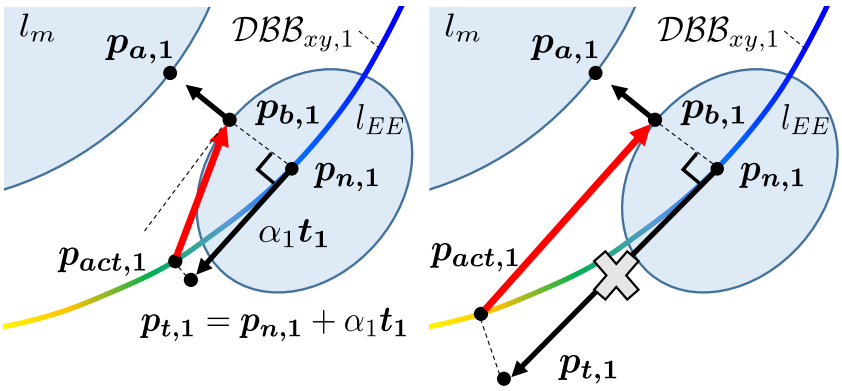


Figure 8. Illustration of finding the acting point. (**a**) The acting point on the DBB is selected as the closest to the point $\boldsymbol{p_{n,1}}$ which is translated along the tangential direction $\boldsymbol{t_1}$ with a step size α_1; (**b**) with a large step size, the generated force may not have a component along the direction connecting the closest point on the manipulator and DBB.

After finding the acting point, the GenerateMobAcc function first computes the force as follows:

$$\boldsymbol{f_{m,i}} = f_{max}(1 - \frac{d_i}{d_b}) \frac{\boldsymbol{p_{b,i}} - \boldsymbol{p_{act,i}}}{\|\boldsymbol{p_{b,i}} - \boldsymbol{p_{act,i}}\|_2}, \tag{14}$$

where $\boldsymbol{f_{m,i}} \in \mathbb{R}^3$ is the force for the i-th link pair and f_{max} is the maximum force. Figure 9 presents the variables in (14) when the link pair $\mathcal{L}_m(1)$ is considered.

The resultant force for all link pairs in $\mathcal{L}_m$ is calculated by adding each force as follows:

$$\boldsymbol{f_m} = \sum_{i=1}^{N(\mathcal{L}_m)} \boldsymbol{f_{m,i}}, \tag{15}$$

where $\boldsymbol{f_m} \in \mathbb{R}^3$ and $N(\mathcal{L}_m)$ are the resultant force and number of link pairs in $\mathcal{L}_m$, respectively.

To realize the resultant force, the corresponding accelerations and corresponding Jacobian matrices are derived as follows:

$$\underline{\ddot{\boldsymbol{x}}}_{\boldsymbol{m}} = \begin{bmatrix} \dot{v}_d & \dot{w}_d \end{bmatrix}^T, \tag{16}$$

where

$$m\dot{v}_d = \boldsymbol{f_m} \cdot \boldsymbol{e_x}, \tag{17}$$

$$I\dot{w}_d = \Big(\sum_{i=1}^{N(\mathcal{L}_m)} (\boldsymbol{p_{act,i}} - \boldsymbol{p_o}) \times \boldsymbol{f_{m,i}} \Big) \cdot \boldsymbol{e_z}. \tag{18}$$

In (17) and (18), $m \in \mathbb{R}$, $\dot{v}_d \in \mathbb{R}$, and $\boldsymbol{e_x} \in \mathbb{R}^3$ are the mass of the mobile robot, desired linear acceleration, and a unit vector perpendicular to the rolling axis of the wheel and pointing forward, respectively. In addition, $I \in \mathbb{R}$, $\dot{w}_d \in \mathbb{R}$, and $\boldsymbol{e_z} \in \mathbb{R}^3$ are the moment of inertia, desired angular acceleration, and a unit vector perpendicular to the ground and pointing upward, respectively. By (17) and (18), the resultant force can be converted into the desired linear and angular accelerations. The Jacobian matrix of the differentially driven mobile robot can be expressed as

$$\begin{aligned} \underline{J}_m &= \left[\begin{array}{c|c} J_m & \boldsymbol{O_{2\times n_m}} \end{array} \right], \\ J_m &= \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2b} & -\frac{r}{2b} \end{bmatrix}, \end{aligned} \tag{19}$$

where $\boldsymbol{J_m} \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{O_{2\times n_m}} \in \mathbb{R}^{2\times n_m}$ are the Jacobian matrix of the mobile robot and the zero matrix, respectively.

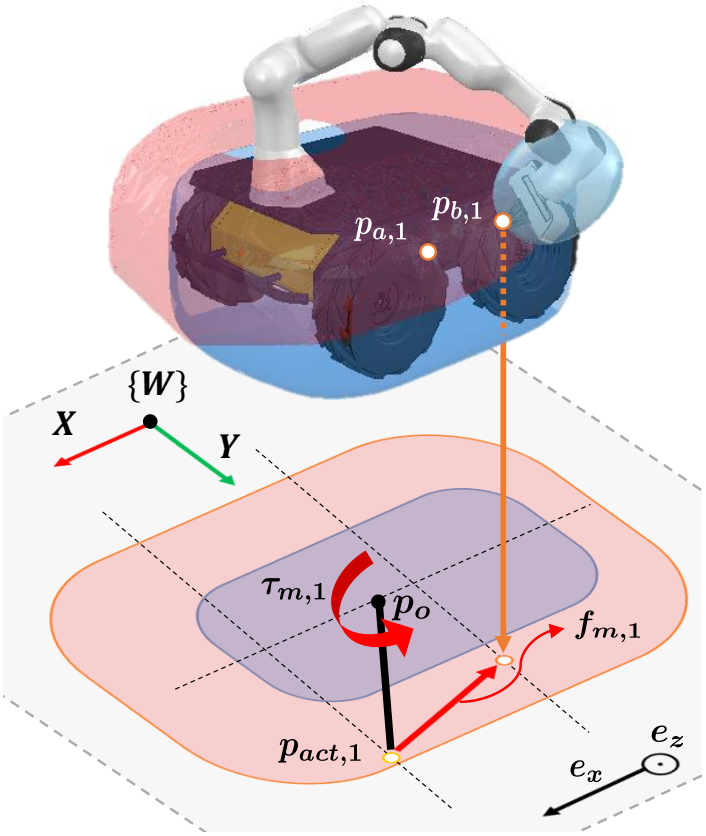


Figure 9. Illustration of generating the desired force and resulting torque for the link pair $\mathcal{L}_m(1)$. When the end-effector and mobile robot are close to each other, the force $\boldsymbol{f_{m,1}}$ and resulting torque $\boldsymbol{\tau_{m,1}}$ with respect to the center of the mobile robot, $\boldsymbol{p_o}$, are generated. The generated force and resulting torque are projected onto the acceleration directions of the mobile robot under the non-holonomic constraint. The directions are expressed as the unit vectors, $\boldsymbol{e_x}$ and $\boldsymbol{e_z}$.

3.2. Generation of the Repulsive Acceleration for the Other Link Pair

To avoid the self-collision of $\mathcal{L}_m^c$, we design a 1-DOF repulsive acceleration to push the link pair away from each other.

Let us consider that the distance of the j-th link pair in $\mathcal{L}_m^c$ is less than the buffer distance. The task for avoiding self-collision with the repulsive acceleration $\ddot{\boldsymbol{x}}_{\boldsymbol{r,j}} \in \mathbb{R}^1$ and Jacobian $\boldsymbol{J}_{\boldsymbol{r,j}} \in \mathbb{R}^{1\times n}$ is designed as follows.

$$\ddot{\boldsymbol{x}}_{\boldsymbol{r,j}} = \boldsymbol{u}_{\boldsymbol{j}}^T(k_p \frac{\boldsymbol{p}_{\boldsymbol{b,j}} - \boldsymbol{p}_{\boldsymbol{a,j}}}{\|\boldsymbol{p}_{\boldsymbol{b,j}} - \boldsymbol{p}_{\boldsymbol{a,j}}\|_2} - k_v(\dot{\boldsymbol{p}}_{\boldsymbol{b,j}} - \dot{\boldsymbol{p}}_{\boldsymbol{a,j}})), \tag{20}$$

$$\begin{aligned} \boldsymbol{J}_{\boldsymbol{r,j}} &= \boldsymbol{u}_{\boldsymbol{j}}^T(\boldsymbol{J}_{\boldsymbol{b,j}} - \boldsymbol{J}_{\boldsymbol{a,j}}), \\ \boldsymbol{u}_{\boldsymbol{j}} &= \frac{\boldsymbol{p}_{\boldsymbol{b,j}} - \boldsymbol{p}_{\boldsymbol{a,j}}}{\|\boldsymbol{p}_{\boldsymbol{b,j}} - \boldsymbol{p}_{\boldsymbol{a,j}}\|_2} \end{aligned} \tag{21}$$

where k_p and k_v are gains, $\boldsymbol{u}_{\boldsymbol{j}} \in \mathbb{R}^3$ is the unit vector from $\boldsymbol{p}_{\boldsymbol{a,j}}$ to $\boldsymbol{p}_{\boldsymbol{b,j}}$, and $\boldsymbol{J}_{\boldsymbol{a,j}}$ and $\boldsymbol{J}_{\boldsymbol{b,j}} \in \mathbb{R}^{3\times n}$ are translation Jacobian matrices for points $\boldsymbol{p}_{\boldsymbol{a,j}}$ and $\boldsymbol{p}_{\boldsymbol{b,j}}$, respectively. For convenience, we define the link to which $\boldsymbol{p}_{\boldsymbol{b,j}}$ belongs as being farther from the base of the manipulator than the link to which $\boldsymbol{p}_{\boldsymbol{a,j}}$ belongs.

When k link pairs in $\mathcal{L}_m^c$ are considered, the repulsive acceleration and Jacobian matrix are stacked as

$$\underline{\ddot{\boldsymbol{x}}}_{\boldsymbol{r}} = \left[\ddot{\boldsymbol{x}}_{\boldsymbol{r,1}}, \cdots, \ddot{\boldsymbol{x}}_{\boldsymbol{r,j}}, \cdots, \ddot{\boldsymbol{x}}_{\boldsymbol{r,k}}\right]^T \tag{22}$$

$$\underline{\boldsymbol{J}}_{\boldsymbol{r}} = \left[\boldsymbol{J}_{\boldsymbol{r,1}}^T, \cdots, \boldsymbol{J}_{\boldsymbol{r,j}}^T, \cdots, \boldsymbol{J}_{\boldsymbol{r,k}}^T\right]^T, \tag{23}$$

where $\underline{\ddot{\boldsymbol{x}}}_{\boldsymbol{r}} \in \mathbb{R}^k$ and $\underline{\boldsymbol{J}}_{\boldsymbol{r}} \in \mathbb{R}^{k\times n}$ are the stacked accelerations and Jacobians, respectively.

3.3. Construction of an Acceleration-Based Task for Self-Collision Avoidance

Based on the obtained accelerations and Jacobians in Sections 3.1 and 3.2, we construct a task, $\mathcal{T}_{sca}$, for avoiding self-collision of all link pairs by stacking them as follows.

$$\ddot{\boldsymbol{x}}_{\boldsymbol{sca}} = \begin{bmatrix} \underline{\ddot{\boldsymbol{x}}}_{\boldsymbol{m}} \\ \underline{\ddot{\boldsymbol{x}}}_{\boldsymbol{r}} \end{bmatrix}, \tag{24}$$

$$\boldsymbol{J}_{\boldsymbol{sca}} = \begin{bmatrix} \underline{\boldsymbol{J}}_{\boldsymbol{m}} \\ \underline{\boldsymbol{J}}_{\boldsymbol{r}} \end{bmatrix}, \tag{25}$$

where $\ddot{\boldsymbol{x}}_{\boldsymbol{sca}} \in \mathbb{R}^{(2+k)}$ is the desired acceleration for the avoidance task and $\boldsymbol{J}_{\boldsymbol{sca}} \in \mathbb{R}^{(2+k)\times n}$ is the corresponding Jacobian matrix.

3.4. Insertion of the Task in HQP-Based Controller

To insert the designed task, $\mathcal{T}_{sca}$, a controller is designed based on the HQP with the continuous task transition approach developed in our previous work [24,25]. HQP is a cascade of QP formulation for dealing with prioritized SoT [32,33]. The main characteristic of the controller with the continuous task transition method is that the continuity of control inputs is always guaranteed even if arbitrary tasks are inserted and removed from the existing SoT. In particular, by using an activation parameter that interpolates feasible solution between existing SoT and new SoT, the method can generate continuous control inputs without modifying the control structure.

We consider the HQP formulation of a single equality task, $\mathcal{T}_2$, with $\ddot{\boldsymbol{x}}_{\boldsymbol{d_2}} \in \mathbb{R}^{m_2}$ and $\boldsymbol{J_2} \in \mathbb{R}^{m_2\times n}$, as follow:

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}},\boldsymbol{u},\boldsymbol{w_2}} \quad & \|\boldsymbol{w_2}\|_2, \\ \text{s. t.} \quad & \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{g} = \boldsymbol{u} \\ & \boldsymbol{J_2}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{\boldsymbol{2}}\dot{\boldsymbol{q}} + \boldsymbol{w_2} = \ddot{\boldsymbol{x}}_{\boldsymbol{d_2}} \end{aligned} \tag{26}$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, $\boldsymbol{C} \in \mathbb{R}^{n\times n}$, $\boldsymbol{g} \in \mathbb{R}^n$, and $\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{b}}^T & \dot{\boldsymbol{q}}_{\boldsymbol{m}}^T \end{bmatrix}^T \in \mathbb{R}^n$ are the inertia matrix, Coriolis and centrifugal matrix, gravity vector, and joint velocity vector of the non-holonomic

mobile manipulator, respectively [34]. In addition, $\boldsymbol{w_2} \in \mathbb{R}^{m_2}$ is a slack variable for $\mathcal{T}_2$ and $\boldsymbol{u} \in \mathbb{R}^n$ is the control torque vector for the robot.

The activation parameter, β, is determined based on the distance between each link pair. Figure 10 presents the activation trajectory when using a cubic spline to insert $\mathcal{T}_{sca}$ smoothly. When the distance is less than the buffer distance of 0.15 m, β gradually increases, and the avoidance task begins to be inserted. In addition, if the distance is less than 0.05 m, β is set to 1 so that the task for avoiding self-collision is fully considered. Because the avoidance tasks for $\mathcal{L}_m$ and $\mathcal{L}_m^c$ are stacked according to (24), we construct a diagonal matrix $\boldsymbol{B}$ from the activation parameters as follows.

$$\boldsymbol{B} = \begin{bmatrix} \beta_m & 0 & 0 & \cdots & 0 \\ 0 & \beta_m & 0 & \cdots & 0 \\ 0 & 0 & \beta_{r,1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \beta_{r,k} \end{bmatrix}, \tag{27}$$

where $\boldsymbol{B} \in \mathbb{R}^{(2+k)\times(2+k)}$ is the diagonal matrix of the activation parameters, β_m is the activation parameter for the link pairs of $\mathcal{L}_m$, and $\beta_{r,j}$ is the activation parameter for the j-th link pair in $\mathcal{L}_m^c$. When considering multiple link pairs of $\mathcal{L}_m$, we choose the maximum value among the activation parameters as

$$\beta_m = \max\left(\beta_1, \cdots, \beta_{N(\mathcal{L}_m)}\right). \tag{28}$$

Based on the activation parameter matrix $\boldsymbol{B}$, the HQP formulation for inserting the self-collision avoidance task as the higher-priority task than $\mathcal{T}_2$ ($\mathcal{T}_{sca} \prec \mathcal{T}_2$), can be represented as

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}}, \boldsymbol{u}, \boldsymbol{w_2}} \quad & \|\boldsymbol{w_2}\|_2, \\ \text{s. t.} \quad & \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{g} = \boldsymbol{u} \\ & \boldsymbol{J_2}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_2\dot{\boldsymbol{q}} + \boldsymbol{w_2} = \ddot{\boldsymbol{x}}_{\boldsymbol{d_2}} \\ & \boldsymbol{J_{sca}}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{\boldsymbol{sca}}\dot{\boldsymbol{q}} + (\boldsymbol{I_{2+k}} - \boldsymbol{B})\boldsymbol{J_{sca}}\ddot{\boldsymbol{q}}_{\boldsymbol{2}}^{*} + \boldsymbol{w}_{\boldsymbol{sca}}^{*} = \boldsymbol{B}\ddot{\boldsymbol{x}}_{\boldsymbol{sca}} \end{aligned} \tag{29}$$

where $\boldsymbol{w}_{\boldsymbol{sca}}^{*} \in \mathbb{R}^{2+k}$ is the optimal slack variable vector for the self-collision avoidance task $\mathcal{T}_{sca}$, $\boldsymbol{I_{2\times k}} \in \mathbb{R}^{(2+k)\times(2+k)}$ is an identity matrix, and $\ddot{\boldsymbol{q}}_{\boldsymbol{2}}^{*}$ is the optimal solution of (26). Thus, if $\boldsymbol{B}$ is a zero matrix, then the feasible solution of (29) is the same as that of (26). When $\boldsymbol{B}$ is the identity matrix, the solution of (29) is strictly satisfied with the priority order, $\mathcal{T}_{sca} \prec \mathcal{T}_2$. In addition, when β gradually increases 0 to 1, the feasible solution of (29) can be derived by interpolating the solution of the HQP of $\mathcal{T}_2$ and the HQP with $\mathcal{T}_{sca} \prec \mathcal{T}_2$. Consequently, the HQP-based controller with the continuous task transition method can insert a self-collision avoidance task without a discontinuous control input.

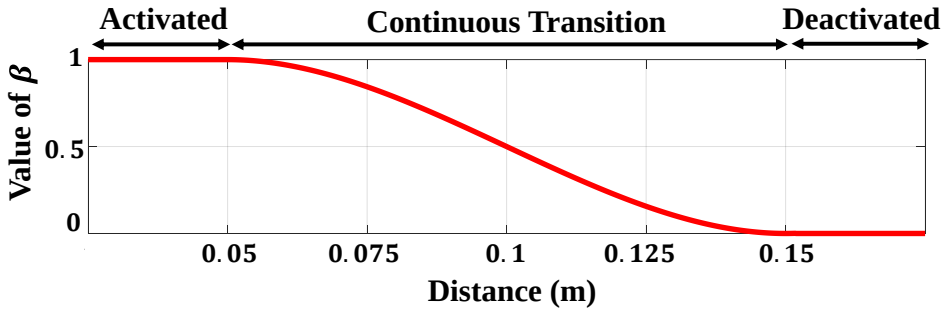


Figure 10. Value of the activation parameter depending on the distance of the link pair.

4. Experimental Results

The self-collision avoidance algorithm was verified through various experiments using a differentially driven mobile robot with a 7-DOFs robotic manipulator. The subsections below describe the details of our system configuration and the experimental results for the robot. It is worthwhile to note that the video clips of the experiments described in this paper are available (https://youtu.be/a9dc4Ij71_M), which are applied for not only our robot, but also other differentially driven mobile manipulators to show the generality of the proposed algorithm.

4.1. System Overview

Our mobile manipulator consists of the velocity-controlled four-wheel differentially driven mobile robot called Husky (Clearpath Robotics. Co.) and a 7-DOFs robot arm manipulator called Panda (Franka Emika. Co.). The specification of the computer for the controller is an Intel *i*7 4.2 GHz CPU with 16 GB of RAM and the control frequencies of the manipulator and mobile robot are 1 kHz and 10 Hz, respectively. The desired velocity command for the mobile robot is computed from the desired torque calculated in (29) by applying the admittance control law [35].

4.2. Experimental Results

4.2.1. Self-Collision Avoidance While Tracking the Predefined Trajectory

To validate the effectiveness of the proposed method, we conducted a comparative experiment using the repulsive force-based method [19,20]. This experiment was designed for the end-effector to track a predefined trajectory that approaches the mobile robot. The task for trajectory tracking of the end-effector is denoted as $\mathcal{T}_{ee} \in \mathbb{R}^6$ and the task for the repulsive force-based method is denoted as $\mathcal{T}_{rep} \in \mathbb{R}^3$. The target position is -0.2 m along the Y-axis from the end-effector. The left snapshots in Figure 11a,b shows the initial positions. The red dots and arrows depict the target position and desired trajectory, respectively. The trajectory was generated for a time period of 30 s using a cubic spline function.

The experimental results are presented in Figures 11 and 12. In Figure 11a, as the end-effector moves close to the mobile robot, force is exerted to move the mobile robot back. As a result of the force, the end-effector reaches the target position while avoiding self-collision. In contrast, in Figure 11b, because repulsive force is generated to push the end-effector away from the mobile robot, self-collision is avoided, but the end-effector can not reach the target position. The distances between the link pairs are shown in Figure 12a. Because the distances are less than 0.15 m, the self-collision avoidance tasks ($\mathcal{T}_{sca}$ and $\mathcal{T}_{rep}$) are inserted continuously with top priority. In Figure 12b, the repulsive force-based method has a position error, while the proposed method does not.

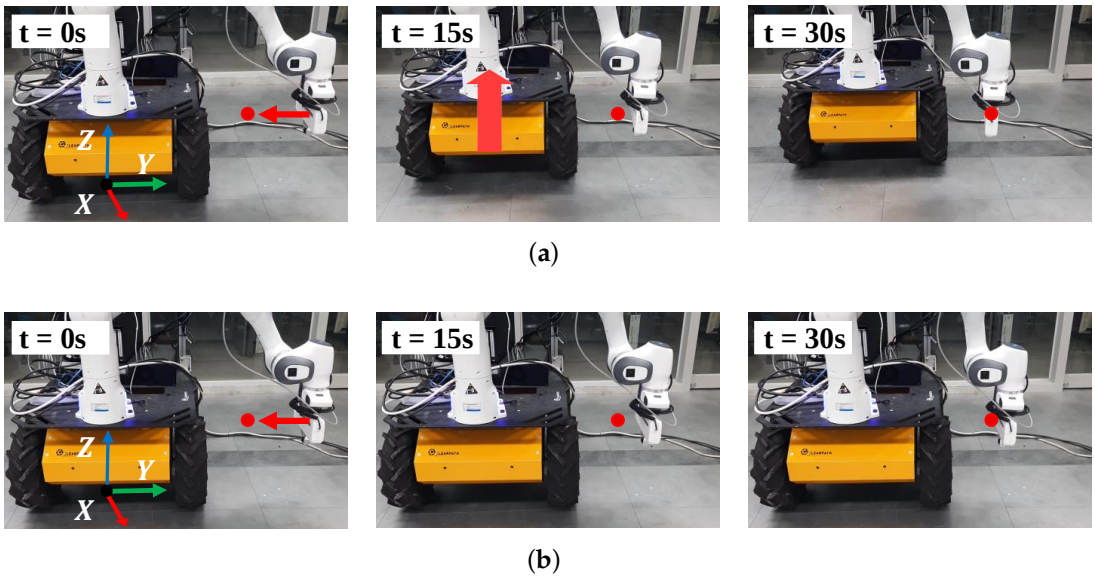


Figure 11. Snapshots during experiments in which the end-effector tracks a predefined trajectory: (**a**) The proposed method generates force to move the mobile robot back, enabling the manipulator to not only avoid self-collision but also reach the target position; (**b**) the repulsive-force based method pushes the manipulator from the mobile robot so that the manipulator cannot reach the target position.

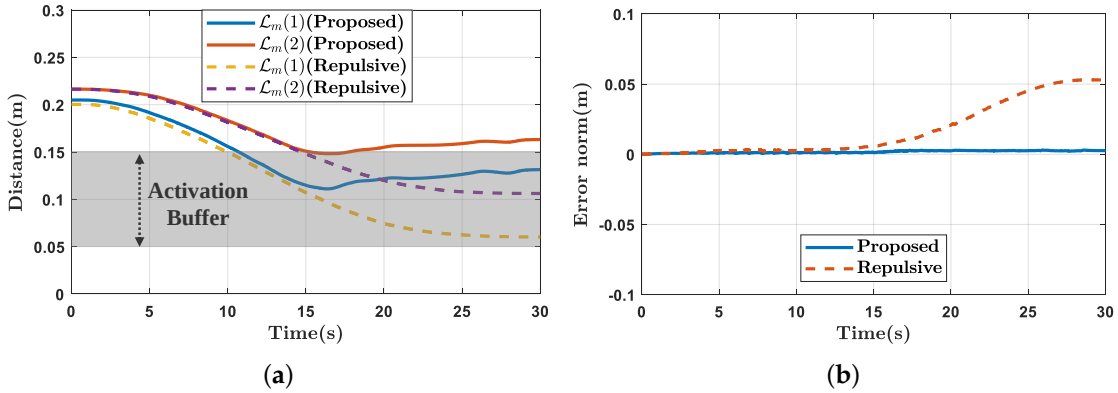


Figure 12. Experimental results of self-collision avoidance while tracking the predefined trajectory. (**a**) Distances of the link pairs ($\mathcal{L}_m(1), \mathcal{L}_m(2)$); (**b**) the norm of the position error.

4.2.2. Self-Collision Avoidance While Manually Guiding the End-Effector

In this experiment, the end-effector was manually guided by an operator to approach the mobile robot to validate reactive self-collision avoidance during human–robot interaction. In the initial state, no tasks are executed other than the gravity compensation of the manipulator. Two directions are considered: the lateral direction and the front direction. The left snapshots in Figure 13 show the initial positions of the mobile manipulator and the guiding directions are depicted by red arrows.

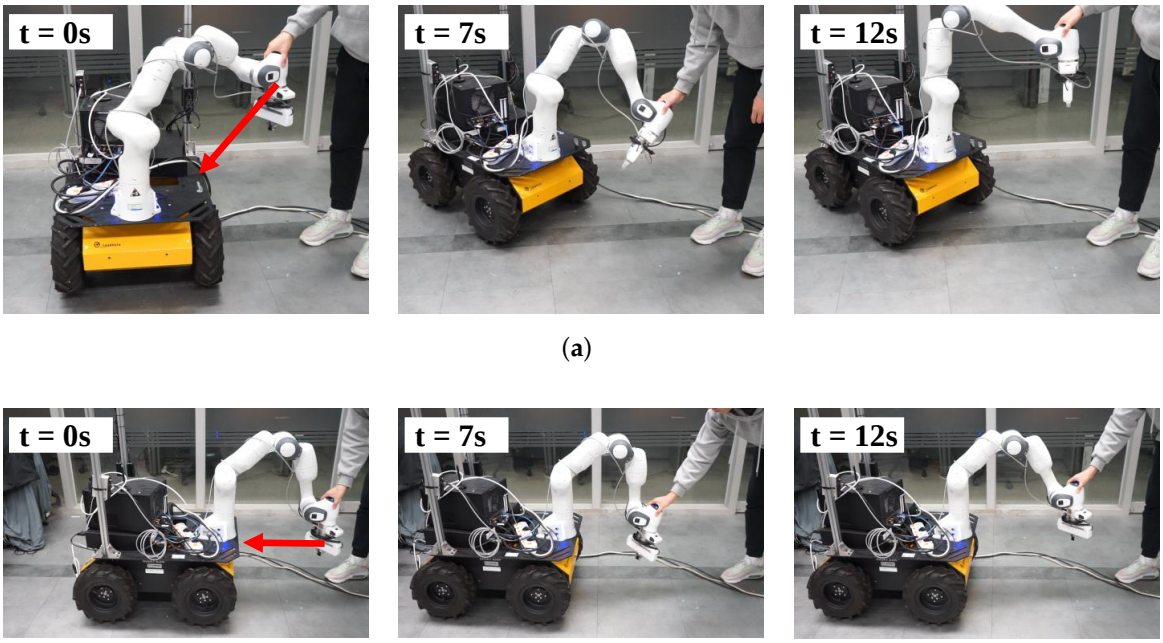


Figure 13. (**a**) The snapshots during the experiment that the manipulator approaches the mobile from the lateral direction; (**b**) the snapshots during the experiment that the manipulator approaches the mobile robot from the front direction. Red arrows show the guiding directions.

As shown in Figure 13a, self-collision between the manipulator and mobile robot are avoided by generating a force exerted on the mobile robot. As shown in Figure 13b, as the manipulator approaches the mobile robot, the mobile robot moves back to avoid self-collision. Figure 14 presents the distances between the links of the manipulator and mobile robot and the values of the activation parameter. As the distance decreases below the buffer distance of 0.15 m, the value of the activation parameter increases accordingly and the self-collision avoidance task, $\mathcal{T}_{sca}$, is inserted continuously as shown in Figure 14b. In Figure 15, the command values of the linear and angular accelerations of the HQP-based controller are plotted. Therefore, self-collision can be avoided regardless of the approach direction of the manipulator, which is an advantage over existing methods [11,20] that do not consider the non-holonomic constraint of the differentially driven mobile robot.

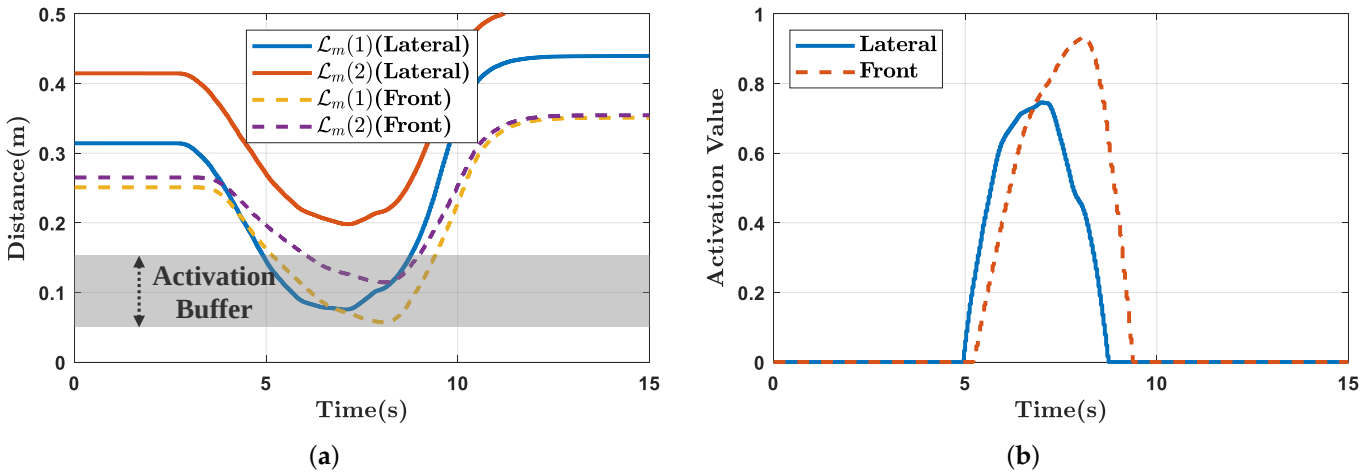


Figure 14. Experimental results of self-collision avoidance while manually guiding the end-effector. (**a**) The distances between the link pairs ($\mathcal{L}_m(1)$, $\mathcal{L}_m(2)$); (**b**) the value of the activation parameter.

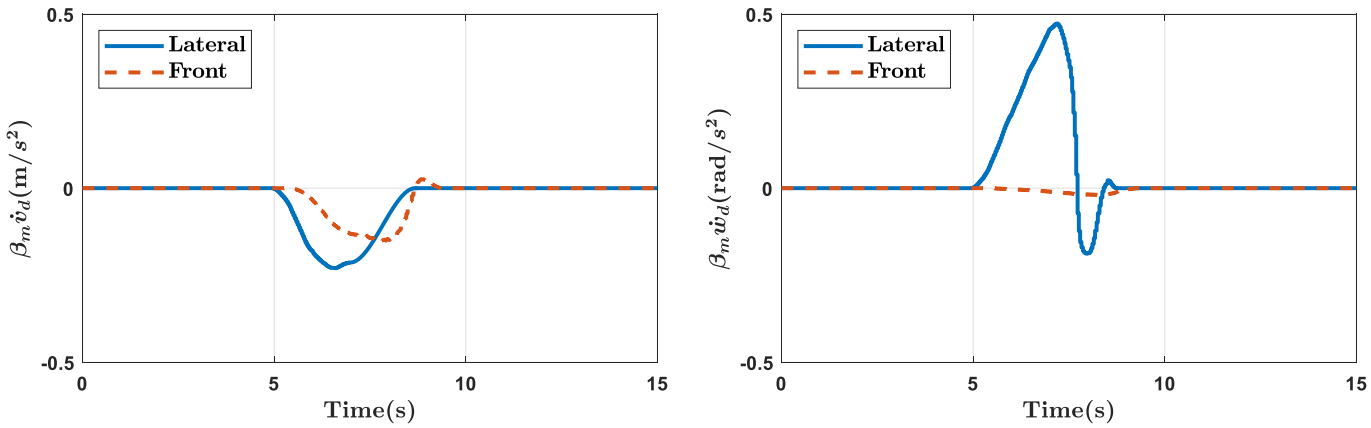


Figure 15. Experimental results of self-collision avoidance while manually guiding the end-effector. (**a**) The desired linear accelerations multiplied by activation parameter; (**b**) the desired angular accelerations multiplied by activation parameter.

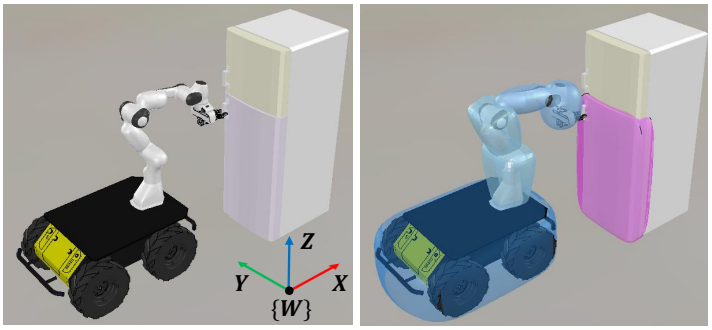
4.2.3. Extension to Obstacle Avoidance When Opening the Refrigerator

In this subsection, we extend our method to obstacle avoidance. The proposed method was tested in a reactive scenario representing a typical example of mobile manipulation. We consider the scenario of opening the refrigerator as shown in Figure 16a. We assume that the end-effector achieves a fixed grasp on the door of the refrigerator, meaning there is no relative motion between them. In this respect, collision between the door and mobile robot is considered. We used a hyper-ellipsoid to design a collision model for the door as shown in Figure 16b.

To open the refrigerator, a control strategy based on adaptive control [36,37] was utilized. The strategy estimates the radial direction of the door based on the force measured at the end-effector so that the end-effector can open the door even with the incomplete knowledge regarding door models. The strategy uses only the manipulator to open the door, meaning the robot may collide with the door depending on the initial pose of the mobile robot. We validated our extension to obstacle avoidance by comparing the results of experiment with and without obstacle avoidance. The scenario of opening the refrigerator was validated in both simulation and experiment.

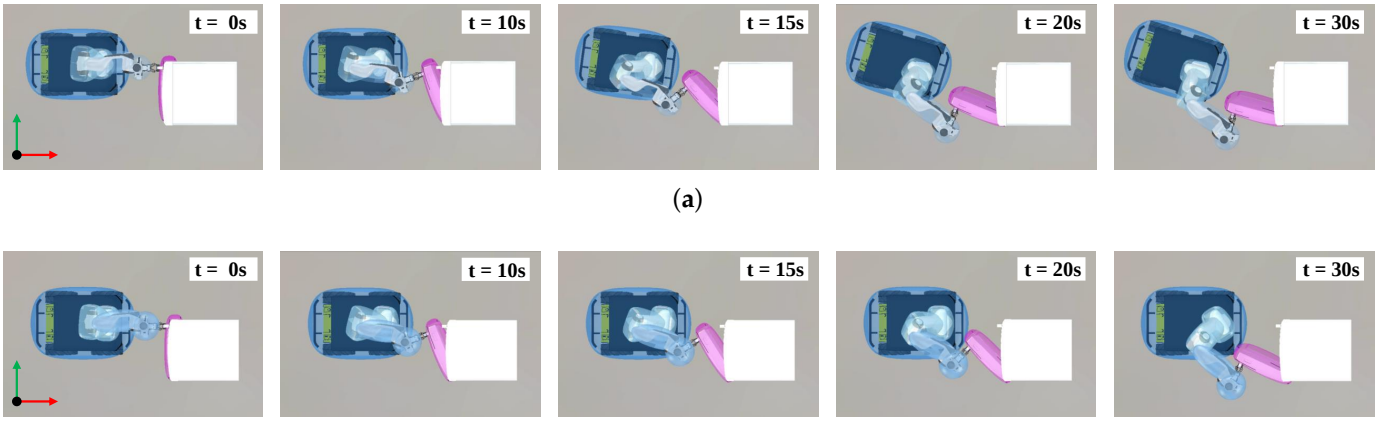
The simulation results are presented in Figures 17 and 18. In Figure 17a, the mobile robot moves back and turns clockwise as the door moves closer to the mobile robot. In contrast, the door collides with the mobile robot at 30 s in Figure 17b. As the distance between the door and mobile robot is less than the buffer distance in Figure 18a, the obstacle avoidance task is inserted continuously as shown in Figure 18b.

The experimental results are presented in Figure 19. As shown in Figure 19a, as the distance between the door and robot decreases, the mobile robot begins to move back at 20 s and the manipulator opens the door completely at 40 s, while avoiding collision. In contrast, in Figure 19b, the manipulator stops opening the door at 30 s because the robot is about to collide with the door.



(**a**) (**b**)

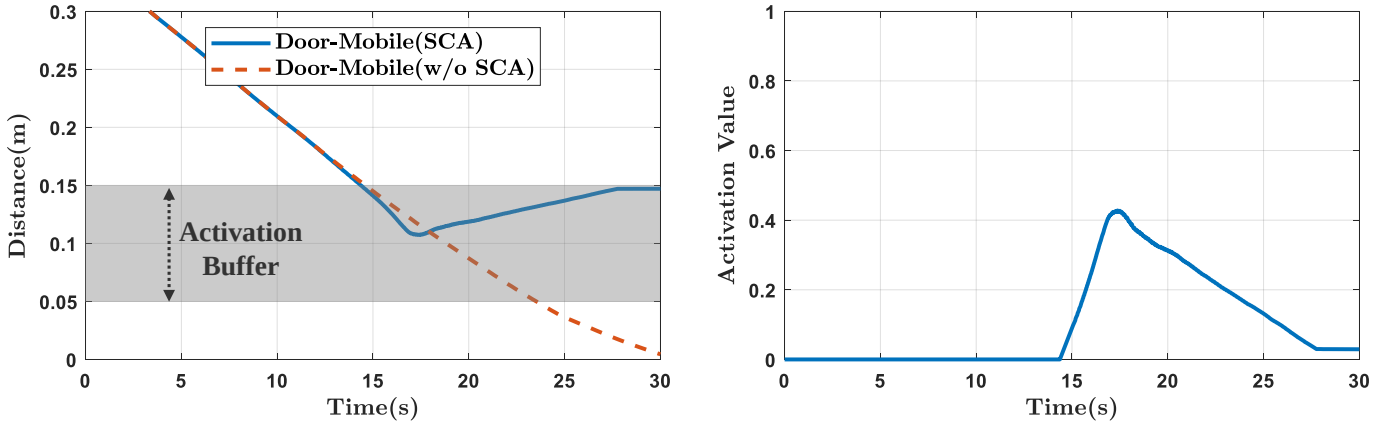
Figure 16. (**a**) Illustration of the scenario of the mobile manipulator opening a refrigerator; (**b**) collision models including the door of the refrigerator are shown. The collision model for the door is colored with magenta.



(**a**)

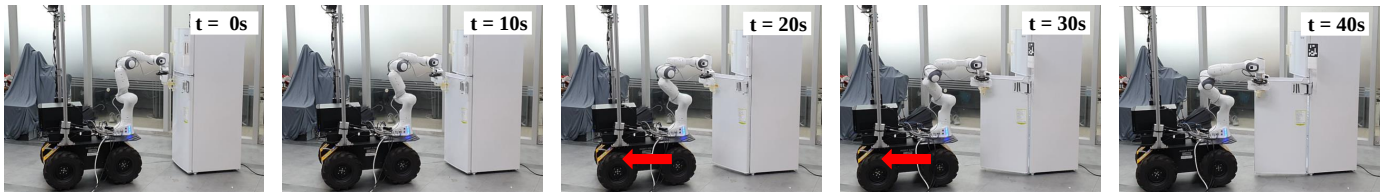
(**b**)

Figure 17. Simulation results of opening a refrigerator. (**a**) Snapshots of opening a refrigerator with obstacle avoidance; (**b**) snapshots of opening a refrigerator without obstacle avoidance.



(**a**) (**b**)

Figure 18. Simulation results of opening the refrigerator. (**a**) The distance between the door and the mobile robot; (**b**) the value of the activation parameter.



(**a**)



(**b**)

Figure 19. Experimental results of opening a refrigerator. (**a**) Snapshots of opening a refrigerator with obstacle avoidance; (**b**) snapshots of opening a refrigerator without obstacle avoidance.

4.3. Discussion

The experimental results in Section 4.2 demonstrate that the proposed method can place the manipulator outside the DBB. Specifically, the proposed method has the following advantages. First, the force can always generate motion for the differentially driven mobile robot with non-holonomic constraint as shown in Section 4.2.2. This is because the acting point is selected such that it is away from the singularity of the mobile robot. Second, the proposed method can be applied to holonomic mobile manipulators if the score of the DBB is designed to include only the reachability of the manipulator. Finally, command values are free from chattering and vibration problems caused by the mobile robot unlike repulsive force-based method [38]. This is because the continuous task transition of (29) can calculate continuous control input. Therefore, the desired accelerations of the mobile robot are smooth, as shown in Figure 15.

From a practical perspective, a trade-off relationship exists between the density of the DBB and the discontinuous position of the acting point. The denser the DBB, the more computational cost increases. However, with a denser DBB, the position of the acting point can be obtained more continuously. According to our practical experience, the proper number of points in the DBB is approximately 50,000 for running the algorithm at a control frequency of 1 kHz.

5. Conclusions

We presented a reactive self-collision avoidance algorithm for differentially driven mobile manipulator. The proposed algorithm generates a force exerted on a mobile robot in a direction that the manipulator is placed outside the DBB. The force is designed based on the concept of the DBB and its score measurement. The score consists of two components, the determinant value of the Jacobian matrix of differentially driven mobile robot and the reachability of the manipulator. After calculating the score for each point on the DBB, a point with high score is selected as the acting point of force, which means that the point is away from the singularity of differentially driven mobile robot and has high reachability of the manipulator. Based on the force and resulting torque, an avoidance task is formulated and inserted into the HQP-based controller with a continuous task transition algorithm. The results of several experiments validated the proposed self-collision avoidance algorithm. Our future work will involve extending the proposed method to other mobile platforms like car-like robots and developing collision avoidance method by using external sensors like cameras. Additionally, we will apply the proposed

algorithm to a wider range of mobile manipulation tasks that needs to detect collision in dynamic and unstructured environment.

Author Contributions: Conceptualization, K.J.; methodology, K.J., S.K.; software, K.J., S.K.; validation, K.J.; formal analysis, K.J., J.P.; investigation, K.J., S.K., J.P.; writing—original draft preparation, K.J.; writing—review and editing, K.J., S.K., J.P.; visualization, K.J.; supervision, J.P.; project administration, J.P.; funding acquisition, S.K., J.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by Industrial Strategic Technology Development Program (No. 10077538) funded by the Ministry of Trade, Industry & Energy (MI, Korea). Furthermore, this study is a part of the research project "Development of core machinery technologies for autonomous operation and manufacturing", which has been supported by a grant from National Research Council of Science & Technology under the R&D Program of Ministry of Science, ICT and Future Planning.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

DBB	Distance Buffer Border
DOFs	Degrees Of Freedom
HQP	Hierarchical Quadratic Programming
SoT	Stack of Tasks

References

1. Asadi, E.; Li, B.; Chen, I.M. Pictobot: A cooperative painting robot for interior finishing of industrial developments. *IEEE Robot. Autom. Mag.* **2018**, *25*, 82–94. [CrossRef]
2. Engemann, H.; Du, S.; Kallweit, S.; Cönen, P.; Dawar, H. OMNIVIL—An Autonomous Mobile Manipulator for Flexible Production. *Sensors* **2020**, *20*, 7249. [CrossRef] [PubMed]
3. Outón, J.L.; Villaverde, I.; Herrero, H.; Esnaola, U.; Sierra, B. Innovative Mobile Manipulator Solution for Modern Flexible Manufacturing Processes. *Sensors* **2019**, *19*, 5414. [CrossRef] [PubMed]
4. Kuffner, J.J.; Kagami, S.; Nishiwaki, K.; Inaba, M.; Inoue, H. Dynamically-stable motion planning for humanoid robots. *Auton. Robot.* **2002**, *12*, 105–118. [CrossRef]
5. Oriolo, G.; Mongillo, C. Motion Planning for Mobile Manipulators along Given End-effector Paths. In Proceedings of the Robotics and Automation (ICRA), 2005 IEEE International Conference, Barcelona, Spain, 18–22 April 2005; pp. 2154–2160.
6. Berenson, D.; Chestnutt, J.; Srinivasa, S.S.; Kuffner, J.J.; Kagami, S. Pose-Constrained Whole-Body Planning using Task Space Region Chains. In Proceedings of the Humanoid Robots (Humanoids), 2009 9th IEEE-RAS International Conference, Paris, France, 7–10 December 2009; pp. 181–187.
7. Burget, F.; Bennewitz, M.; Burgard, W. BI2RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation. In Proceedings of the Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference, Daejeon, Korea, 9–14 October 2016; pp. 3714–3721.
8. Welschehold, T.; Dornhege, C.; Paus, F.; Asfour, T.; Burgard, W. Coupling Mobile Base and End-Effector Motion in Task Space. In Proceedings of the Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference, Madrid, Spain, 1–5 October 2018; pp. 1–9.
9. Kang, M.; Kim, D.; Yoon, S. Harmonious Sampling for Mobile Manipulation Planning. In Proceedings of the Intelligent Robots and Systems (IROS), 2019 IEEE/RSJ International Conference, Macau, China, 3–8 November 2019; pp. 3185–3192.
10. Seto, F.; Kosuge, K.; Hirata, Y. Real-time self-collision avoidance system for robots using robe. *Int. J. Humanoid Robot.* **2004**, *1*, 533–550. [CrossRef]
11. Seto, F.; Kosuge, K.; Hirata, Y. Self-collision avoidance motion control for human robot cooperation system using RoBE. In Proceedings of the Intelligent Robots and Systems (IROS), 2005 IEEE/RSJ International Conference, Edmonton, AB, Canada, 2–6 August 2005; pp. 50–55.
12. Stasse, O.; Escande, A.; Mansard, N.; Miossec, S.; Evrard, P.; Kheddar, A. Real-time (self)-collision avoidance task on a HRP-2 humanoid robot. In Proceedings of the Robotics and Automation (ICRA), 2008 IEEE International Conference, Pasadena, CA, USA, 19–23 May 2008; pp. 3200–3205.

13. Schwienbacher, M.; Buschmann, T.; Lohmeier, S.; Favot, V.; Ulbrich, H. Self-Collision Avoidance and Angular Momentum compensation for a biped humanoid robot. In Proceedings of the Robotics and Automation (ICRA), 2011 IEEE International Conference, Shanghai, China, 9–13 May 2011; pp. 581–586.
14. Dariush, B.; Bin Hammam, G.; Orin, D. Constrained Resolved acceleration control for humanoids. In Proceedings of the Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference, Taipei, Taiwan, 18–22 October 2010; pp. 710–717.
15. Fang, C.; Rocchi, A.; Hoffman, E.M.; Tsagarakis, N.G.; Caldwell, D.G. Efficient self-collision avoidance based on focus of interest for humanoid robots. In Proceedings of the Humanoid Robots (Humanoids), 2015 15th IEEE-RAS International Conference, Seoul, Korea, 3–5 November 2015; pp. 1060–1066.
16. Quiroz-Omaña, J.J.; Adorno, B.V. Whole-Body Control With (Self) Collision Avoidance Using Vector Field Inequalities. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4048–4053. [CrossRef]
17. De Luca, A.; Oriolo, G.; Robuffo Giordano, P. Kinematic modeling and redundancy resolution for nonholonomic mobile manipulators. In Proceedings of the Robotics and Automation (ICRA), 2002 IEEE International Conference, Orlando, FL, USA, 15–19 May 2006; pp. 1867–1873.
18. De Luca, A.; Oriolo, G.; Robuffo Giordano, P. Kinematic control of nonholonomic mobile manipulators in the presence of steering wheels. In Proceedings of the Robotics and Automation (ICRA), 2002 IEEE International Conference, Anchorage, AK, USA, 3–7 May 2010; pp. 1792–1798.
19. Dietrich, A.; Wimböck, T.; Täubig, H.; Albu-Schäffer, A.; Hirzinger, G. Extensions to reactive self-collision avoidance for torque and position controlled humanoids. In Proceedings of the Robotics and Automation (ICRA), 2011 IEEE International Conference, Shanghai, China, 9–13 May 2011; pp. 3455–3462.
20. Dietrich, A.; Wimböck, T.; Albu-Schäffer, A.; Hirzinger, G. Integration of reactive, torque-based self-collision avoidance into a task hierarchy. *IEEE Trans. Robot.* **2012**, *28*, 1278–1293. [CrossRef]
21. Sugiura, H.; Gienger, M.; Janssen, H.; Goerick, C. Reactive Self Collision Avoidance with dynamic task prioritization for humanoid robots. *Int. J. Humanoid Robot.* **2010**, *7*, 31–54. [CrossRef]
22. Gonon, D.; Jud, D.; Fankhauser, P.; Hutter, M. Safe Self-collision Avoidance for Versatile Robots Based on Bounded Potentials. In *Field and Service Robotics*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 19–33.
23. Lei, M.; Wang, T.; Yao, C.; Liu, H.; Wang, Z.; Deng, Y. Real-Time Kinematics-Based Self-Collision Avoidance Algorithm for Dual-Arm Robots. *Appl. Sci.* **2020**, *10*, 5893. [CrossRef]
24. Kim, S.; Jang, K.; Park, S.; Lee, Y.; Lee, S.Y.; Park, J. Continuous Task Transition Approach for Robot Controller based on Hierarchical Quadratic Programming. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1603–1610. [CrossRef]
25. Kim, S.; Jang, K.; Park, S.; Lee, Y.; Lee, S.Y.; Park, J. Whole-body Controller for non-holonomic mobile manipulator based on HQP with continuous task transition. In Proceedings of the Advanced Robotics and Mechatronics (ARM), 2019 IEEE International Conference, La Mirada, CA, USA, 3–5 July 2019; pp. 414–419.
26. Campion, G.; Bastin, G.; Dandrea-Novel, B. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Trans. Robot. Autom.* **1996**, *12*, 47–62. [CrossRef]
27. Bloch, A.M.; Reyhanoglu, M.; McClamroch, N.H. Control and stabilization of nonholonomic dynamic systems. *IEEE Trans. Autom. Control* **1992**, *37*, 1746–1757. [CrossRef]
28. Campion, G.; d'Andrea Novel, B.; Bastin, G. Modelling and state feedback control of nonholonomic mechanical systems. In Proceedings of the IEEE Conference on Decision and Control, Brighton, UK, 11–13 December 1991; pp. 1184–1189.
29. Yamamoto, Y.; Xiaoping, Y. Coordinating locomotion and manipulation of a mobile manipulator. *IEEE Trans. Autom. Control* **1994**, *39*, 1326–1332. [CrossRef]
30. Diankov, R. Automated Construction of Robotic Manipulation Programs. Ph.D. Thesis, Carnegie Mellon University, Pittburgh, PA, USA, 2010.
31. Diankov, R.; Kuffner, J. *OpenRAVE: A Planning Architecture for Autonomous Robotic*; Technical Report CMU-RI-TR-10-29; Robotics Institute, Carnegie Mellon University: Pittsburgh, PA, USA, 2008.
32. Escande, A.; Mansard, N.; Wieber, P.B. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *Int. J. Robot. Res.* **2014**, *33*, 1006–1028. [CrossRef]
33. Hong, S.; Jang, K.; Kim, S.; Park, J. Regularized Hierarchical Quadratic Program for Real-Time Whole-Body Motion Generation. *IEEE/ASME Trans. Mechatron.* **2020**. [CrossRef]
34. White, G.D.; Bhatt, R.M.; Tang, C.P.; Krovi, V.N. Experimental evaluation of dynamic redundancy resolution in a nonholonomic wheeled mobile manipulator. *IEEE/ASME Trans. Mechatron.* **2009**, *14*, 349–357. [CrossRef]
35. Dietrich, A.; Bussmann, K.; Petit, F.; Kotyczka, P.; Ott, C.; Lohmann, B.; Albu-Schäffer, A. Whole-body impedance control of wheeled mobile manipulators. *Auton. Robot.* **2016**, *40*, 505–517. [CrossRef]
36. Karayiannidis, Y.; Smith, C.; Viña, F.E.; Ogren, P.; Kragic, D. "Open sesame!" adaptive force/velocity control for opening unknown doors. In Proceedings of the Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference, Vilamoura, Portugal, 7–12 October 2012; pp. 4040–4047.
37. Karayiannidis, Y.; Smith, C.; Barrientos, F.E.V.; Ögren, P.; Kragic, D. An Adaptive Control Approach for Opening Doors and Drawers Under Uncertainties. *IEEE Trans. Robot.* **2016**, *32*, 161–175. [CrossRef]
38. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*; Springer: Berlin/Heidelberg, Germany, 1986; pp. 396–404.