CoboDeck: A Large-Scale Haptic VR System Using a Collaborative Mobile Robot

Soroosh Mortezapoor* TU Wien, Vienna, Austria
Khrystyna Vasylevska† TU Wien, Vienna, Austria
Emanuel Vonach‡ TU Wien, Vienna, Austria
Hannes Kaufmann§ TU Wien, Vienna, Austria

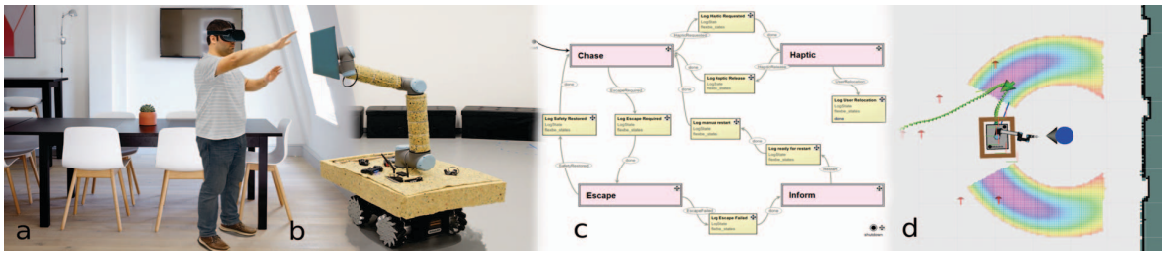

Figure 1: Concept of CoboDeck: (a) User interacts with a virtual wall. (b) Mobile cobot presents prop for haptic feedback. (c) State machine that models the cobots behavior. (d) Interaction situation as it is represented in the control system.

Abstract

We present CoboDeck - our proof-of-concept immersive virtual reality haptic system with free walking support. It provides prop-based encountered-type haptic feedback with a mobile robotic platform. Intended for use as a design tool for architects, it enables the user to directly and intuitively interact with virtual objects like walls, doors, or furniture. A collaborative robotic arm mounted on an omnidirectional mobile platform can present a physical prop that matches the position and orientation of a virtual counterpart anywhere in large virtual and real environments. We describe the concept, hardware, and software architecture of our system. Furthermore, we present the first behavioral algorithm tailored for the unique challenges of safe human-robot haptic interaction in VR, explicitly targeting availability and safety while the user is unaware of the robot and can change trajectory at any time. We explain our high-level state machine that controls the robot to follow a user closely and rapidly escape from him as required by the situation. We present our technical evaluation. The results suggest that our chasing approach saves time, decreases the travel distance and thus battery usage, compared to more traditional approaches for mobile platforms assuming a fixed parking position between interactions. We also show that the robot can escape from the user and prevent a possible collision within a mean time of 1.62 *s*. Finally, we confirm the validity of our approach in a practical validation and discuss the potential of the proposed system.

Index Terms: Computer systems organization—Embedded and cyberphysical systems—Robotics;Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality; Human-centered computing—Human computer interaction (HCI)—Interaction devices—Haptic devices;

1 Introduction

Design thinking is a traditional approach to problem-solving in design [12], for instance, in architecture. It employs strategic logic and analysis, leading to iterative prototyping to find the desirable solution. Its rationality might be seen as both a positive and negative trait: it supports a search for the most effective and rational solution, yet might lead to template-based results [12]. On the other side, the design feeling or emotional design [14] focuses on intuition, spontaneity, and emotions to elaborate the experience without turning to logic of any kind. This approach focuses on what the user feels while engaging with the designed object. Usually, it is not easy to unite these two approaches, but virtual reality (VR) offers such a possibility. Moreover, haptic VR can deepen the feeling aspect beyond just visuals. Haptic feedback can significantly improve task performance and presence in VR [48] and enhance the transfer of experience from the virtual to the physical world [5]. We propose a haptic VR system that facilitates the incorporation of thinking and feeling approaches in the creative design process. Our target use case is the early design phase for architecture and interior design. Arbitrary 3D sketches and architectural design pose several challenges in VR, requiring: 1) large-scale support, 2) realism of experience for locomotion and touch, 3) real-time support for dynamically created content, and 4) safe usage at all times. In this paper, we present CoboDeck, a system for large-area haptic VR experiences. We employ an omnidirectional mobile cobot with a human-friendly robotic arm to position physical props or proxies for haptic interaction and support free walking in VR (see Fig. 1). Cobot is a shorthand for a robot intended for safe, direct interaction and collaboration with a human. Our major contribution is the universal concept of a system for large-scale VR with haptic feedback using an autonomous mobile cobot. We propose an approach to close-distance human-robot interaction (HRI) where the user not necessarily sees the robot in VR. Our system is capable of enabling a wide variety of haptic interactions that include but are not limited to many existing approaches presented in Sect. 2. We suggest a novel algorithmic approach using a high-level state machine that defines the autonomous robot's behavior and describe every state in detail. Our concept simultaneously addresses multiple aspects crucial for VR: scalability, adaptability, availability, safety, and haptic interaction in general. Additionally, we discuss the specifics of robotic navigation for timely haptic feedback and collision avoidance when the user's trajectory is not known in advance. We also evaluate and discuss the properties of the proposed system: the space requirements and coverage, response times, safety, and possible optimizations.

2 Related Work

Providing a convincing haptic experience is still challenging, and all existing approaches have severe limitations. The use of passive physical props resembling a desired virtual object, also referred to as "passive haptics" [32], was shown to deliver very realistic and robust

*e-mail: soroosh.mortezapoor@tuwien.ac.at
†e-mail: khrystyna.vasylevska@tuwien.ac.at
‡e-mail: emanuel.vonach@tuwien.ac.at
§e-mail: hannes.kaufmann@tuwien.ac.at

2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR) | 979-8-3503-4815-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/VR55154.2023.00045

2642-5254/23/$31.00 ©2023 IEEE
DOI 10.1109/VR55154.2023.00045

haptic feedback [21]. It affords a substantial increase in realism in VR [25], which was also proven at large scales, e.g., a virtual pit [57] or even life-sized virtual environments [33]. The major shortcomings are the synchronization with the virtual environment (VE) and scalability, as passive props implicitly are not controlled by a computer and must be available in every form and every position required by the simulation. Other possible solutions, e.g., instructing people [10], attaching motorized strings [42] or using a linear motor [65] to actuate the physical props, are costly or restrictive. Another approach is to equip the floor with actuators to simulate different terrain [27] or coarse physical props [52, 55], which limits the scalability. Our system employs a robot arm on a mobile platform to dynamically position passive physical props as required by the VE.

McNeely [35] first speculated about using robots to provide force feedback in VR. Hirota & Hirose [22] and Yokokohji et al. [66] established this research field as "encountered-type haptic devices" (ETHD). The system places a haptic device at a desired location and waits for the user to encounter it. These pioneering works demonstrated the potential to experience different forces and real touch sensations of virtual objects with free-hand interaction but were limited by available technology to interaction with a single finger or sensor knob. More recent examples of ETHDs support free handling of presented props [37], different surface textures [4] or physical interaction with a revolving door [29]. However, they suffer from a general problem of ETHDs: the limited interactive volume of the used robotic arm. Thus, various researchers suggested grounding robotic haptic devices on the user's own body, e.g., the fingers [11], the arm [56], waist [3], back [13,23], or other body parts [34]. These body-grounded devices support large VEs and provide haptic feedback at different 3D positions but are bulky and can impede the user experience with their weight and inevitable sense of counter-force on the body. Vonach et al. [58] suggested a solution for unrestricted haptic interaction with a stationary robotic arm and a locomotion device [9, 26] thus is not for large-scale walkable VEs.

An alternative solution to increase the available interactive volume is to use mobile robots. For example, quadcopters have been employed to deliver haptic proxies for VR like a sheet of paper [64] or a clothes hanger [2]. Similarly, a quadcopter might follow the user's hand closely to render the appropriate force feedback [1]. Although these approaches allow a reasonable interactive volume, the possible object's weight and forces are very limited. In comparison, our robotic arm provides a payload of 10 kg and allows rendering stronger forces. Bouzbib et al. [7] employ a cartesian ceiling robot moving a physical column that allows high forces, but scalability and room adaptability are limited, and the available interactions are predefined. Other researchers combined haptic devices with mobile robots to create Mobile Haptic Interfaces (MHI) [41]. Han et al. [18] presented an MHI with omnidirectional movements, and Pavlik et al. [44] used a similar assembly in room-scale VR. However, the limitation of haptic interaction to a single point limits the applicability, while our system offers full-hand interaction. Fitzgerald & Ishii [15] added a palm-sized shape display on top of an omnidirectional mobile robot limiting interaction to the geometric surface at the robot's height. Meanwhile, mobile robots might configure physical props for haptic feedback [19] or serve as haptic proxies themselves [17, 53, 59]. These works show the potential for natural two-handed interaction but are limited to tabletop use. VR setups with cleaning robots carrying physical objects as ETHD showed results comparable to passive haptic feedback at room-scale [51, 61]. Their systems suffer from late haptics availability due to the slow movement of the robots or accidental collisions with the user. For a more extensive overview of ETHDs, we recommend the survey by Mercado et al. [38] and the work on haptics by Bouzbib et al. [8].

Addressing the problems with flexibility, safety, timing, and availability of the haptic props, in our algorithm, the robot stays close to the user while preventing unintended contact. Mobile robots following a person constitute an ongoing research challenge. To solve the target following task, researchers used cameras [40], laser range finders [30, 67], or skeletal information of the user [39]. Popov et al. [45] described an omnidirectional mobile platform detecting and following a person with a suitcase using the Kinect sensor. These kinds of mobile robots allow lateral movements and therefore excel, especially in limited spaces [60]. Wu et al. [62] developed a predictive following algorithm for an industrial omnidirectional mobile robot to avoid obstacles and permit robust pursuit. The aforementioned systems suffer from problems with visual occlusions and handling multiple persons, thus can only prevent simple collisions. None of them is intended for use in VR or for haptic feedback. The situation in a VR environment is unique and requires special considerations. An HMD practically blindfolds the user, and conventional strategies for avoiding collisions, like stopping completely, could provoke dangerous situations, e.g., a trip hazard. There have been attempts to address safety with regard to VR. Wu et al. [63] proposed a mixed reality setup to control a physical omnidirectional mobile robot in VR. The VR objects are considered for optimization of path planning and safety, but there is no haptic HRI. Similarly, Kato et al. [28] controlled a mobile robot over a VR telepresence system, but there is also no HRI. Bouzbib et al. [7] employed various emergency stops and a simple user intention model to position physical props timely and safely. However, only predefined objects are supported, and the use of emergency stops might be problematic in VR. Our algorithm supports impromptu haptic feedback and more sophisticated avoidance of dangerous situations. Mercado et al. [36] explored various visual feedback techniques to make the user aware of an ETHD to prevent collisions. In our system, we aim to avoid breaking immersion; thus, we expose the robot as a last resort. Bouzbib & Bailly [6] structured the approach to identify potential risks and reviewed solutions for ETHDs in VR but offered no universal concept for safety or availability.

Employing a mobile platform with a robot arm to provide physical props for free-hand haptic interaction in a walkable VR setup remains mostly unexplored. Unlike previous works, we employ an industrial-class mobile robot as opposed to small mobile robots such as cleaning robots. There is no general behavior approach for ETHDs in large-scale VR that supports spontaneous user actions and undefined scenarios. Existing control algorithms for mobile robots hardly consider the challenges of VR: safety, availability and robustness, and, moreover, do it simultaneously. In this work, we seek to bridge this gap.

3 Technical Setup

The VE in our proposed system is rendered using the Unity 3D game engine on a Microsoft Windows 10 desktop PC with an Intel Core i7 9900K, an Nvidia RTX 2080Ti graphics card, and 32GB of memory. The user is equipped with the HTC VIVE Pro HMD in a wireless configuration. We used the HTC wireless adapter with the battery placed in a belt bag to reduce the extra weight on the head. The user is equipped with two VIVE Pro Controllers to allow interaction with the VE and a VIVE v2 tracker on the tailbone. In order to track the VIVE equipment, four HTC Lighthouse v2 tracking stations are employed to create a workspace of 6 m *by* 10 m. The mobile robot used in the setup is a Robotnik RB-KAIROS platform equipped with a Universal Robots UR10 robotic arm, a WiFi 6 router, and some

Robotic Platform	Robotnik RB-KAIROS
Drive	Differential, omnidirectional
Max Velocity	3 m/s
Robot's Weight	130 kg
Robotic Arm	Universal Robots UR10 CB3
Arm Payload	10 kg max
Controller CPU & RAM	Intel Core i7 4790S, 8GB
OS & ROS	Ubuntu 18.04 LTS, ROS Melodic

Table 1: The specifications of the robot and its onboard computer.

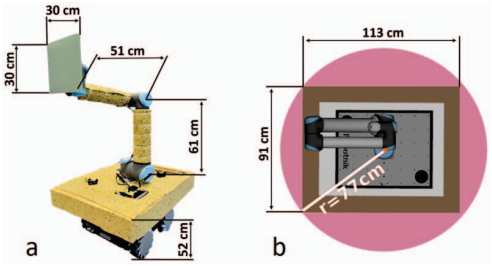

Figure 2: The augmented RB-KAIROS robot. (a) The physical robot with the added foam bumper and (b) its minimal enclosing circle.

additional equipment. The robot has four Mecanum wheels [24] that allow driving in differential and omnidirectional modes. Table 1 provides more detailed robot specifications. Two VIVE v2 trackers are added to the robot diagonally to augment the robot's localization system. Moreover, we added a bumper to the rover with impact sensors on a wooden frame covered by 10 *cm* of composite foam. A Teensy© 4.1 microcontroller board controls the impact sensors. It is connected directly to the robot's hardware safety system and internal computer with Robot Operating System (ROS). On impact, the robot's emergency stop is triggered with the event communicated ROS-wide. Fig. 2a depicts the robot. Altogether, the robot's footprint is a 113 *cm by* 91 *cm* rectangle as shown in Fig. 2b. The minimal enclosing circle with a 77 *cm* radius surrounding the robot defines the minimum space for an in-place rotation (see Fig. 2b).

4 System Description

CoboDeck is a collaborative system comprising two major parts: the Unity and the ROS subsystems. Their components and data exchange are shown in Fig. 3 and are explained in detail below. The two subsystems are deployed on two separate computers, which communicate using a high-bandwidth, low-latency WiFi 6 connection (with ROS Connector). The virtual wall between the user and the robot represents the simplest haptic use case. The user wearing an HMD sees only the wall and not the robot. The mobile robot's manipulator places the haptic prop at the proper interaction position to enable haptic interaction with the virtual wall.

The Unity subsystem is running on the user's PC, handling everything related to VR, such as rendering the VE (Synchronization & Visualization), tracking the user's pose (SteamVR), and handling the interaction with the virtual objects (Interaction Manager). Additionally, it provides the robot's localization system with the VIVE-based rover position using the two VIVE trackers mounted on the robot.

On the other hand, the ROS subsystem is responsible for the robot's operation and behavior in the real workspace. It operates on the robot's onboard computer. ROS is a meta-operating system comprising multiple worker processes called nodes. They address the different needs of an autonomous mobile robot, i.e., the high-level application-dependent behavior (FlexBE), control of motors, path planning, and collision avoidance (Navigational Subsystems), as well as the integration of different means of localization (Multi-modal Localization). Low-level control of the platform's hardware is provided by native ROS-compatible drivers and controllers from Robotnik and Universal Robots (RB-KAIROS platform).

4.1 Unity Subsystem

On the user's side, all VR aspects of the system are handled by the Unity. The user's position is defined by a tracker on the tailbone matching the center of mass. As opposed to the head-based position estimation, the user has more freedom to move. For instance, he might crouch, bend forward, or to the sides without misinforming the system about the actual position of the lower part of the body. The hand motions are tracked with hand-held controllers. The collected tracking information is published to the ROS subsystem via ROS *TCP Connector* Unity package for data exchange [54].

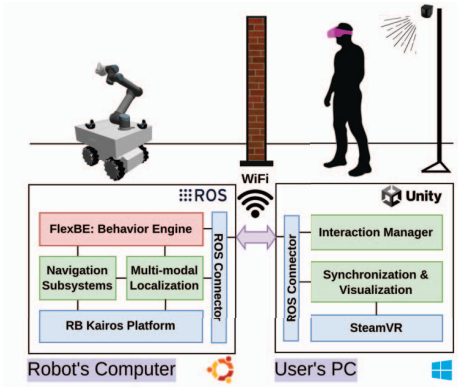

Figure 3: This diagram depicts the building blocks of the collaborative ROS−Unity system.

When the user wants to interact with the VE, we detect the exact position in the space where the interaction should happen and notify the ROS subsystem. This way, the user's pose is communicated to move the robot to a nearby position for direct interaction and then move the arm into the communicated pose for haptic feedback. We call this pose the Desired End-Effector Pose (*DEEP*). The other way around, the Unity environment is kept in sync with the ROS subsystem by relying on the external clock server and following the updates of */joint_states* and other topics to be able to recreate the position of the robotic arm in the Unity scene. Topics are the asynchronous way of communication in ROS.

The robot is described using the Unified Robot Description Format (URDF), an XML format that might contain references to other files, supporting the modular approach. We used the ROS URDF Importer from Unity Robotic Hub to import the robot model from the ROS standard description file. Although we assume that the user should stay unaware of the robots' presence in the environment, there might be a case when robot visualization is necessary for debugging purposes or to notify the user that the robot cannot prevent a potential collision. In the latter case, the ROS subsystem will enter the "Inform" behavior described below, making the user aware of the danger by showing a model of the robot and a warning message.

4.2 Robotic Platform

The onboard computer of our robotic platform (see specification in Table 1) runs an instance of ROS Melodic on the Ubuntu 18.04 LTS operating system. The robot's sensors and actuators are connected to the onboard computer using USB, CAN bus, or LAN. Their data is transmitted from/to ROS using their corresponding drivers. ROS can be described as a highly-modular network of connected peer-to-peer processes (nodes) that follows master-slave architecture. The master is a lightweight process that provides the API registration needed to establish one- or bi-directional communication between the nodes. It provides asynchronous (topics) and synchronous RPC-style (services) communication and supports multi-platform distributed operation. The ROS nodes are always organized by relevance in various ROS packages. From all ready-to-use ROS packages, the noteworthy *transform* (TF) package offers functionality similar to game engines' scene graphs. It is used to transform the poses of different parts of the system and environment relative to each other in a data structure called a TF tree. Unlike the scene graphs, the TF tree stores information over time, and a buffered TF client can access the transformation history. Additionally, the static and dynamic transforms can be distinguished easily in a TF tree.

4.2.1 Simulation

For data visualization and troubleshooting, we use *Rviz* - a free, open-source 3D visualization tool for ROS [20]. *Rviz* contains visualization and debugging tools specifically for ROS and provides real-time visualization of the joints, sensors, navigation, and other

data. We use *Gazebo 9* [46] for accurate physics and sensor simulation. These are decoupled from the visualization and interface, integrating maximum information about the robot from the URDF file and the environment's precise 3D model. We simulate the user in ROS by defining the area of direct contact with the body as a cylinder with a 1 *m* diameter. The user's position data comes from the Unity subsystem as predefined, pre-recorded, or live.

4.2.2 Localization

The robotic system requires access to precise, reliable, and up-to-date localization data to have a valid estimate of the current situation within the physical environment. Since the environment is mostly known, we use a multi-sensor map-based localization approach to localize the robot. The sensors utilized for localization include LiDAR, IMU, odometry wheel encoders, and VIVE trackers. Additionally, the joint angles of the robot arm are transmitted from the arm's driver. Then, they are used to localize the links of the arm relative to its base and, respectively, to the base of the rover (*base_link*) - specified in the robot's description file. The main localization task in the environment for the rover is to find a correct transformation from *map* to *base_link*, which is done by the Adaptive Monte Carlo Localization (AMCL). This probabilistic localization system for robot motion in 2D [16] is based on the Kullback-Leibler distance sampling for the Monte Carlo localization approach [43], which uses a particle filter to track the robot's pose against a known map. With the help of a laser-based map (LiDAR data) and TF messages, the AMCL node can estimate the robot's pose in the known map.

AMCL requires odometry information. Usually, it includes the data from the wheel encoders and IMU. Since the IMU data and the wheel encoders are prone to error, especially on slippery surfaces or after fast omnidirectional movements, especially with rotations, the AMCL output might encounter unpredictable drift. We use the VIVE trackers added on top of the rover to correct the drift timely. By placing them on opposite corners, we minimized the possibility of occlusion. The trackers' data are used to recalibrate the AMCL node once per second or up to 10 Hz when the robot is rotating.

VIVE tracking is limited and covers only a part of our available space. Transitioning between the tracked and non-tracked areas, the trackers stop sending data, and on the edges of calibrated space, they might produce inaccurate data. To get the best possible localization, we employed extended Kalman filters to filter out the potentially noisy data and combine the available filtered data to estimate the correct rover position. In such a situation, when the VIVE trackers' data are unavailable, localization relies on the fused data of the AMCL, odometry, and IMU using the extended Kalman filter.

4.2.3 Navigation

The navigation planning and execution are handled by a ROS navigation package called *move_base*. It supports different configurations of global and local path planners and collision avoidance algorithms. This ROS package contains the necessary tools to build configurable costmaps based on the environment's map and the sensor data, mainly from the LiDARs. Configuration parameters of the costmaps allow altering the cost of an area around an obstacle (obstacle inflation) to ensure no path is planned in this region. The generated costmaps are utilized by the global and local path planners. They are responsible for planning a path based on the environment's map and costmap, as well as executing it in smaller chunks considering static and dynamic obstacles.

It is common to have only one running instance of *move_base* nodes per robot. Nevertheless, due to particular requirements of the robot's behavior in different situations, we decided to use three parallel *move_base* instances (see also 4.4), run in their own namespaces. Each of the three instances can have its own set of path-planning algorithms with different parameter values and individual costmap layers and settings. There are two costmaps per instance: global and local. The global costmap is created based on the environment's map and is static. In contrast, the local costmap is created in a smaller rolling window (we used standard 5 *m by* 5 *m*) around the robot that is computed on-the-fly based on the LiDAR data. Continuous reevaluation makes it responsive to dynamic obstacles. The path planners consider both costmaps when planning.

We used ROS Global Planner for the global path and Timed Elastic Band local planner [47] for local path adaptation in all three instances. However, their parameter values differ according to the behavior's requirements. For instance, Escape computes a simpler but faster path, and the other behaviors use more complex and efficient paths at the cost of higher planning time.

Having three parallel *move_base* instances results in three navigation subsystems capable of commanding the robot at the same time. We added the control signal multiplexer *twist_mux* ROS package, which includes a message multiplexer node to connect only one of the outputs of these three subsystems to the wheels' controller at a time based on their predefined priorities. Such a structure ensures that navigation subsystems take control of the robot based on their priority and release it after completing the navigation.

4.2.4 Arm Inverse Kinematics & Trajectory Planning

Move_it is an advanced open-source library for motion planning and manipulation tasks. It offers instruments for forward and Inverse Kinematics (IK) calculations that rely on external libraries and, thus, are highly flexible. *Move_it* uses the published states of all joints in the */joint_states* topic by its corresponding controller, and its action server ensures that the robot follows the computed trajectory in a controlled manner. Furthermore, it allows the utilization of different inverse kinematics solver algorithms with diverse properties and comes with a list of implemented algorithms to select from. We used RRTConnect IK solver [31] for its relatively quick and light IK and self-collision-free trajectory calculations. We set the range of accepted solutions to 2 *cm* and 0.1 *rad* around the DEEP.

During the rover's motion, the robotic arm stays in the parked position, a safe position where all parts of the arm are within the space above the rover's bumper frame. This way, it is close to the rover and occupies a minimal volume, which ensures better stability and higher safety while moving. When the rover reaches the destination for haptic interaction, the arm moves to the DEEP for the needed duration. When the haptic feedback is no longer necessary, the Haptic Release signal is sent, and the arm returns to the parked position again.

4.3 Safety

Our system is at the prototype stage; thus, it is not entirely fail-safe. Building a reliable safety system requires extensive testing and is currently a work in progress. However, we have evaluated the most common risks and implemented some crucial safety precautions.

One of the basic safety measures is the emergency stop button. During the robot's operation, there is always an observer with a remote emergency control in hand. Should there be a danger to people or equipment, the observer can press the button and trigger the emergency brakes of the robot to render it motionless. In addition, we added a strongly padded bumper frame with an emergency button functionality that mechanically dampens the impact if a collision should happen nonetheless. It is useful should the braking path be too short, or a dynamic obstacle is on a collision course. Furthermore, the bumper frame makes it practically impossible for the robot's wheels to reach the feet of the user. To decrease the effect of a possible impact, we have limited the robot's maximum allowed velocity and acceleration, which is explained later. The robot's localization system, specifically LiDAR sensors, provides real-time information about obstacles. The navigation system is configured so that the robot avoids any possible obstacles and holds a safe distance of 1 *m* from static or moving objects (e.g., people).

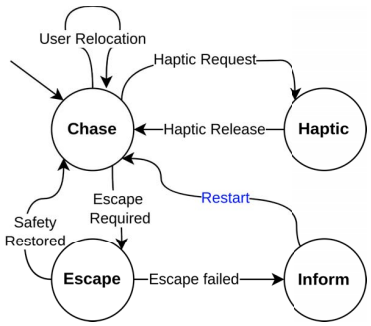

Figure 4: The high-level state machine that defines the robot's behavior. Each state abstracts an extended state machine called a behavior.

Finally, we set up a testing procedure. Any change is first tested in the simulation, then in reality with a dummy. Only trained personnel is allowed to interact with the robot directly at the moment, with the presence of an observer equipped with an emergency-stop device.

4.4 Robot's Behavior Set

FlexBE is a versatile high-level behavior engine that offers the possibility to define and easily modify complex behaviors in robotics via visual programming [49, 50]. The engine features concurrent states that make it more suitable and flexible to define and implement autonomous robots' behaviors. Such robots often require at least two parallel processes to operate effectively: one for performing an action and the other for monitoring the environment variables.

Fig. 4 shows the high-level deterministic finite state machine of our algorithm with four primary states called behaviors hereafter. These four behaviors Haptic, Chase, Escape, and Inform, are abstracted extended finite deterministic state machines with many internal states. They have access to the data generated by the ROS nodes and made available on the ROS topics, can process these data further, and issue high-level commands to the ROS navigation and actuation control nodes. When a behavior is entered, it remains active as long as environmental signals do not trigger a transition. As soon as a transition is activated, the state machine transitions to another behavior to alter the robot's response to the environmental situation. After startup, our system enters the Chase behavior, where the robot tries to stay close to the user to be available for a Haptic Request. Once the Unity application issues such a request, the Haptic state is entered, where a haptic prop is presented. Afterward, the system returns to the Chase state. The Escape state is triggered if the user moves in a way that might lead to a collision. In this situation, the robot immediately moves to a safe location and switches back to Chase when safety is restored. Only if Escape fails, the system enters the Inform state to warn the user about the situation.

Apart from Inform, each behavior has its own navigation stack to use, with a predefined order of priority to take control of the robot. The highest priority in activation belongs to the Escape, followed by Haptic, and then Chase. The behaviors with higher priority do not need to wait for the other behaviors with less priority to finish or cancel their navigation tasks before taking full control. To react quickly and adequately to environmental changes, each behavior incorporates two concurrent states: one for executing the sequence of the actions that define the behavior and the other for monitoring the environmental parameters to detect the necessity of a transition. The output of these two concurrent states is channeled in a disjunctive way, meaning that any of the states reaching their output will be sufficient to perform the behavior transition to the subsequent behavior.

4.4.1 Haptic Behavior

The robot's first and primary function is to provide haptic feedback. When the user intends to interact with a haptic object, the Unity

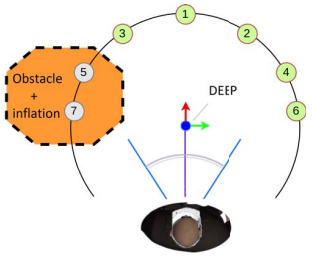

Figure 5: Haptic Rover Poses: The DEEP (blue dot) is in front of the user. The circle around it with $r = 1\ m$ contains 7 predefined poses, from which two are in an obstacle and discarded.

subsystem issues a Haptic Request and sends the DEEP. The robot needs to plan and drive along a 2D collision-free path and respond to static and possibly dynamic obstacles before reaching the DEEP using its arm. The rover's goal is called the Haptic Rover Pose - a position in the room from which the arm can bring the prop (end-effector) in the DEEP. The pose for the rover is calculated based on the arm's measurements, the prop's size and shape, and the physical constraints that influence the reachability of the arm with the end-effector. Furthermore, the navigation algorithm's positioning tolerance should be large enough to prevent the robot from staggering while trying to reach its goal. This tolerance should be accounted for the DEEP reachability as well. Not considering it in the calculations can cause the rover's stop where the DEEP is unreachable for the arm. Hence the IK solver will fail to plan any path for the arm. In this case, or if the user's intention changes or the haptic feedback is no longer required, a Haptic Release signal is sent. Note that the positioning tolerance of the rover does not influence the DEEP precision, as the robot arm compensates for it. In this behavior, the concurrent *FlexBe* states perform the Haptic behavior and monitor the Release signal topic from Unity simultaneously while providing haptic feedback. The criteria for sending the Haptic Request are application dependent and thus are configured in Unity. For this, we rely on the distance to the object's interactable surfaces ($1.5\ m$) and use the user's orientation to determine the DEEP in 3D.

We define the space of Haptic Rover Poses as a set of points on the half circle with a $1\ m$ radius in front of the DEEP XY projection (Fig. 5). Considering the XY goal tolerance of the haptic navigation stack ($20\ cm$), the distance of the Haptic Rover Poses to the DEEP remains reachable. The positioning of the robot along the vector from the user to the DEEP is preferred. This reduces the stress on the arm's joints' motors when the user interacts with the prop, especially if the user suddenly decides to push strongly or lean against the end-effector, for example, in the case of haptic interaction with a virtual wall. Thus the highest priority goes to the Haptic Rover Pose #1. The order of the other haptic poses might be changed according to the application scenario, the environmental variables, and the prop in use. The Haptic Rover Poses are checked against the global and local costmaps, and those with non-zero costs are excluded.

Typically after moving the robot to the Haptic Rover Pose utilizing its navigation stack, the DEEP, which is expressed in the global map coordinates, needs to be transformed into a position within the coordinate system of the arm's base (*arm_base_link*). For this, we utilize the transformation from the robot's *base_link* to *arm_base_link*, relying on the robot description (URDF) and the current angles of the relevant joints. Nevertheless, *move_it* provides the possibility of specifying the DEEP in any coordinate system as long as there is a path in the transformation tree from the center of this coordinate system to the *arm_base_link*. Then the transformation calculation can be automatically performed by the *move_it* nodes.

4.4.2 Chase Behavior

When haptic interaction is expected in a dynamically changing scene like a design space, there is little to no knowledge about the spatial

arrangement and positions of the virtual objects that can be introduced to the scene or destroyed at run time by the user. Nevertheless, the haptic feedback should be provided in the shortest possible time after a Haptic Request. To address this challenge, the robot must stay in close range to the user for any possible haptic request, yet not too close to hinder the user's natural behavior or cause any interruptions in the physical or virtual environments. Therefore a behavior called *Chase* is defined to keep the robot around the user when the robot is not in use for haptic interaction by the Haptic behavior.

The first task in the Chase behavior is to ensure that the arm is in the parked pose for moving around safely. The pose definition depends on the shape of the prop that is attached to the robot's arm and needs to be specified using configuration variables before the start. After moving the arm to its parked position, the robot starts navigating to stay close to the user. As long as the state machine has not received any transition signals to switch to other states, it remains in the Chase behavior, and the robot keeps repeating the chasing procedures in a loop. The algorithm is designed to choose the best position around the user and decide when to move the robot there. The chasing procedures are triggered in the behavior when at least one of the following criteria is met: 1) The user has been *standing still* for > 0.5 *sec* after a translational movement of > 30 *cm* or a rotational movement of $> \frac{\pi}{6}$ *rad*; 2) the distance between the robot and the user is over 4 *m*. In the latter case, the robot navigates toward the user, stops at a 3 *m* distance, and waits for him to satisfy the first criterion. As long as the user moves, the robot remains in the range of $3-4$ *m* from the user. When the user stops moving, the robot should navigate closer and be ready to switch to the Haptic state quickly. However, this new position should be safe enough for the user to continue the movement.

Apart from the user's intentions, parameters like the geometry of the physical workspace where both user and robot move, the positions of the static and dynamic obstacles, and the user's orientation play important roles in finding a proper position for the robot in Chase. For that, we define a Chase space - two equivalent arcs around the user that form a donut-like shape (Fig. 6). It moves and rotates together with the user in its center. The Chase space is a grid of cells representing Chase Poses. Every cell is assigned a weight in the range $[0,1]$ that specifies the desirability of that particular Chase Pose (visualized by rainbow coloring with the highest weight in the purple area, see Fig. 6a). The higher the weight, the higher the chance of the Pose becoming the robot's destination in the cycle of the Chase behavior. In Fig. 6a, the robot is at the Chase Pose with the highest weight. The resolution of the Chase space is adjustable. Based on the application and our requirements, we decided to set the resolution of the Chase space to 5 *cm*. This value could be reduced for more refined results or increased to lighten the computational load. The inner radius of the Chase space arcs is 1.5 *m*, and the outer radius is 3 *m*, constituting the closest and furthest possible distances for a Chase Pose. Note that these points are the positions of the *base_link* - the center of the rover. Consequently, if the robot stands on any Chase Pose of the Chase space, the distance between the user's legs and the bumpers of the rover is less than the distance of the Chase Pose to the user's position.

To reduce the clash probability between the robot and the user, if the latter suddenly decides to start walking from a still-standing state, we excluded a $\frac{\pi}{3}$*rad* from the Chase space in front of the user. Similarly, a $\frac{\pi}{4}$*rad* behind the user was excluded, as deploying and stationing the robot manipulator behind the user is not advantageous from many perspectives. This includes a higher distance to the probable Rover Haptic Pose and higher complexity of planning and collision avoidance due to the user standing in the way.

The location alone is insufficient for successful positioning of the robot near the user. The target orientation should also be decided and passed to the relevant navigation node. Considering the robot's shape, we decided to position it with the longer side facing the user. Thus we used the tangent vectors of the arches that belong to the Chase space (depicted as small arrows in Fig. 6a).

Finally, after calculating the current Chase space, it is checked against the related global and local costmaps from its corresponding navigation subsystem. In this state, we chose an obstacle inflation extent of 1 *m*. The occupied areas of the costmap are excluded from the Chase space to ensure a valid destination for path planning. Fig. 6b shows a situation with the user in a position where most of the arcs around him are occupied by the walls' inflation. Therefore only the rest of the Chase space within the free space is used for goal setting and is visualized. A Haptic Request signal received by the concurrent monitoring state of this behavior can make the state machine transition to the Haptic behavior. Respectively, an Escape Required signal can cause a transition to the Escape behavior.

4.4.3 Escape Behavior

One of the most critical aspects of this system is the user's safety. Following the user around and keeping the robot close to him must not put him at risk of any physical harm. Nonetheless, since the user's movements might be unpredictable, the robot might get too close to the user or be in his way. Such a situation can occur when the user suddenly changes the direction toward the robot and walks. Avoiding a collision requires agility and efficiency in destination finding, path planning, and executing the plan by moving the robot to the computed destination.

Escape behavior preempts a potential user−robot collision by escaping to a safe position outside the user's path or far enough away from him. When the 2D distance of the user to the robot's center is below 2.5 *m*, and in $\pm\frac{\pi}{5}$ *rad* of his view, or below 1.5 *m* anywhere outside this view angle, the Escape Required signal is raised and as receiving it, the state machine transitions to the Escape behavior. Note that during the Haptic behavior, the user has to violate these conditions. Therefore the Escape is disabled while the Haptic Request is active.

For destination finding, we developed a node that computes an ordered set of destination poses all the time at 30 *Hz*, regardless of the current behavior of the robot. The predefined configurable set of these poses forms the Escape space, as shown in Fig. 6a. The Escape poses are always on the opposite side of the robot from the user's position to increase the distance. The Escape space is subsequently verified against behavior-related costmaps to filter out the occupied elements of the space, as demonstrated in Fig. 6b.

The Escape behavior utilizes a dedicated *move_base* instance optimized for simple time-sensitive path planning actions with the highest priority among the navigation subsystems in the multiplexer node. As safety is a priority, the obstacle inflation extent of the Escape navigation subsystem is reduced from 1 *m* to a minimum of 0.77 *m* to maximize the available space for the Escape's planning algorithm. This allows the robot to get closer to obstacles in the scene when it is crucial and make the environment safe again.

As shown in Fig. 6, the Escape poses' orientations are the same as the robot's current orientation. By avoiding the rotational movement when navigating from the current position to the Escape pose, we decrease the probability of collision with the user. The Escape navigation subsystem is configured to use its omnidirectional drive mode only. In differential drive mode, the time to perform the rotational movement while the user is on the go is counterproductive. The rectangular shape of the robot and a possibly extended robotic arm might also be hazardous to the user during the emergency rotation.

With all considered measures and depending on the geometrical shape of the physical room, there can be moments when the robot cannot perform the Escape behavior in time or the navigation node is unable to create a valid plan or execute it. In such cases, the failure can be dangerous for the user, as the robot cannot move away from the user's walking path. When such a situation happens, the state machine transitions from Escape to Inform behavior.

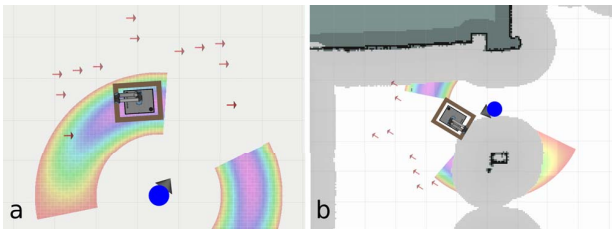

Figure 6: Visualization of the Chase and Escape spaces, with the user (blue circle) and his viewing direction (grey arrow). (a) The Chase space positions as rainbow-colored arcs (from purple for the highest to red for lowest desirability). The Escape space has 12 Escape Poses (red arrows). (b) A detected Escape situation with Escape Poses after verification against the costmap (same as the Chase space).

4.4.4 Inform Behavior

The purpose of the Inform behavior is to inform the user about a probable collision with the robot in the current walking path. The Inform signal is sent from ROS to Unity to trigger the visualization of the potential danger for the user. The robot's model is rendered as soon as Unity receives the signal, depending on the application. The position of the robot's 3D model is synchronized with its actual position in the physical workspace. The robot and the stop sign should be clearly visible in the user's HMD to trigger the user's stop. Then a restart signal is sent back to ROS, which causes a transition back to the Chase behavior. If the Chase behavior fails for any reason (e.g., when the robot is stuck in a corner of a room surrounded by walls and the user), the state machine transitions to the Escape behavior, and if the Escape behavior fails, it transitions to the Inform behavior again. This loop continues until the robot can find a destination to follow in the Chase behavior or a destination and route to escape in the Escape behavior.

5 Evaluation and Discussion

The evaluation of the CoboDeck is done in three steps. First is the analysis of its spatial requirements and limitations. Then, the technical evaluation compares our suggested behaviors to existing approaches to haptics employing mobile platforms. In this phase, we simulate an ideal user walking and interacting with the VE. We analyze the behaviors in simulation, focusing on the main objectives of each behavior. Finally, we address the real world by deploying the behaviors to the real robot. This phase aims to validate the technical evaluation. We discuss practical specifics, such as adjustments and limitations, as well as possible optimization methods.

5.1 Spatial Considerations and Requirements

For safety reasons, the robot cannot navigate the whole available room. The unusable free space is mainly made of the obstacle inflation. The minimum inflation width depends on the robot's shape and size (see Fig. 7) and the workspace characteristics: length, width, and columns or walls in it. Our robot requires at least a 77 cm margin for safe navigation along the robot's planned path with omnidirectional motion or rotation in place (see Fig. 2b). Consequently, a 77 cm inflation is applied along each wall and column. For instance, a column excludes the area sized $((154+a)\ by\ (154+b))\ cm^2$ where a and b are the sides of the column. In our case, that's 184 by 204 cm or 3.75 m^2, affirming that open or column-free spaces are preferable to spaces with columns.

The total amount of space lost due to obstacle inflation depends on the workspace's boundary's shape, the number of columns present, and their sizes. Most common rooms are rectangular, some have columns every 3 to 4 m (or 6 to 8 m for larger rooms with stronger columns). When the inflation size of the walls is 77 cm, the minimum distance between two columns or a column and a wall needs to be > 1.54 m to be usable by the robot for navigation purposes. This requirement applies to environments with separate spaces connected

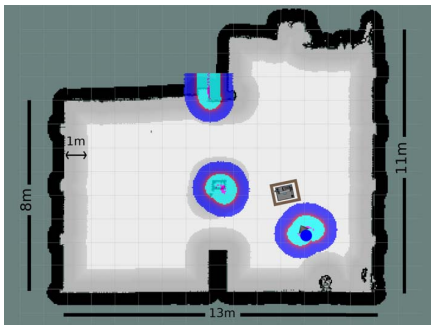

Figure 7: The workspace 2D map. The walls are black, the inflation of map-based obstacles is gray, and the available space for navigation is white. The dark-blue circle is the user that is only included in the local costmap (LiDAR data) with its local inflation (shades of blue).

using doors or narrow passages, meaning that if the width of the portal is below this threshold, the robot will not be able to travel through and reach the other part of the environment.

Moreover, we increased the obstacle inflation size from 77 cm to 100 cm for the Haptic and Chase states. This was done to minimize risk to equipment in the workspace that could occasionally be missed by the LiDARs, such as VIVE's tripods (see Fig. 7). This also helps with positioning errors and sliding on the floor during braking or deceleration.

In our case, the room's total free space is approx. 115 m^2. It consists of two connected areas with small walls and a column in between. After the inflation area is subtracted, about 62 m^2 is left in the global costmap (see Fig. 7), leaving the robot with 54% of the total free space to operate in. A circular room with the same area, 115 m^2, having one column in the middle with 1 m inflation size, leads to a usable area of around 74 m^2 (64%). The same settings in a square room provide around 71 m^2 (62%) of usable area.

The space for each behavior may be easily reconfigured and adapted to the working environment. However, with the current settings, we can presume a circular space of at least 4 m radius around the user to allow the robot to perform any behaviors defined. At the same time, it is not compulsive for the whole circle to be available, and only a $\frac{\pi}{2}$ of the circle is sufficient for all described behaviors. Adding the inflation layer of the walls in an example square room leads to a minimum of 6 $m\ by\ 6$ m free space in a room to set up the system. This minimum assumes that the user is either stationary or allowed to do only minor translational movements around this area. The larger this rectangular space becomes, the more walking area the user will have.

5.2 Technical Evaluation

To test the different aspects of our behaviors, we conducted an evaluation in a simulated 3D replica of the actual workspace under three conditions. In the first condition, the robot uses the Chase behavior when idle and stays close to the user, waiting for an interaction request. The second condition is performed with the robot having a parking position in the corner of the workspace as shown in Fig. 8b. Each time the Haptic Request is received, the robot drives to present the prop, and after the Haptic Release, it navigates back to the parking position and waits to receive the subsequent Haptic Request. The third condition keeps all parameters of its preceding, except the robot's parking position is chosen close to the center of the workspace (see Fig. 8c). Each condition is tested with a batch of 100 interaction requests with Escape enabled to see the differences. The maximum allowed velocity of the robot in Haptic and Chase is set to 1.5 m/s and 2.5 m/s in Escape. The robot model utilized in the simulation is provided by the manufacturer. Therefore, the model's physics properties and behavior comply with the real robot.

The user's movement is predefined with a script and has a speed

Condition	Response time, s		Distance, m	Escapes
	Mean	Std. Dev.		
Chase	10.25	1.27	1130.94	26
Corner	12.79	1.07	1554.40	10
Center	13.00	2.92	1180.45	15

Table 2: The test results in three conditions. Mean response time (and SD) from haptic request to drive and present the prop for interaction (100 requests/condition). Total distance traveled by the rover during the test. Number of Escapes triggered per test condition.

of 1.5 $\frac{m}{s}$. We specified seven positions for the user, distributed in different parts of the virtual workspace. In every condition, the simulated user's behavior is identical: the user repeatedly visits 7 predefined positions in the same order until there are 100 served interactions. Visiting each position, the user is orientated at these predefined positions so that there is enough room for the robot to perform the Haptic behavior successfully. He stands still for 10 s upon arrival before making a Haptic Request. The DEEP is requested at 85 cm from the user in the front and at a height of 1.5 m. When the Haptic Request is sent, it stays active for 30 s. After that, the Haptic Release is issued. The simulated user then waits for 5 s and heads to the next interaction point. In all three conditions, the Escape behavior is enabled since this behavior is related to the user's safety, and an escape situation can happen in any of these variations. The robot knows neither the next defined interaction point in the test nor the user's behavior. While performing each test, the trajectory of the robot was recorded. The resulting trajectories are shown in Fig. 8.

5.2.1 Chase Behavior

The response time is measured from the Haptic Request signal to the prop reaching the DEEP, including driving the robot to the Haptic Rover Pose. The boxplot of the response time is shown in Fig. 9, and the extended details are provided in Table 2.

The mean response time with Chase was shorter by more than 2 s compared to conditions with the parking position. The response time variance in Chase is also much smaller than in other conditions. For a fair comparison of the response times between conditions, we summed up the response times for each sequence of seven targets shown in Fig. 8 and compared these sums using the Kruskal-Wallis and Jonckhee-Terpstra (J.-T.) tests. The response times were significantly affected by the condition type $H(2) = 27.62, p < 0.001$. Pairwise comparisons with J.-T. and adjusted p-values showed that Chase significantly differed from the Corner ($p < 0.001, r = 0.85$) and Center conditions ($p < 0.001, r = 0.85$), but the Corner and Center did not differ from each other ($p = 0.22, r = 0.14$).

The Center parking condition produced a slightly longer mean response time than the corner parking due to the occupation density of the central area requiring more complex paths. At the same time, it is unlikely the central parking condition's average response time would be shorter than the Chase in an obstacle-free map. In Chase, the robot maintains a distance to the user that is always closer than the distance to the central parking position. It is worth noting that, although noticeably more Escape situations were detected and reacted to in the Chase condition, the total distance traveled during the evaluation is lower than in conditions with the fixed parking positions. This suggests that the energy consumption of the Chase behavior is comparable to other conditions while the average response time is shorter. Moreover, in workspaces larger than ours, Chase might result in better energy consumption efficiency than other conditions.

5.2.2 Haptic Behavior

The response time of the arm is defined as the time for the arm trajectory planning and execution. With the RRTConnect IK solver, it is measured to be within a range of 3.20 to 5.30 s with a mean of 3.70 s and standard deviation of 0.67 over all three test variations (i.e., 300 samples). This does not include the time for the rover navigation and maneuvering into the Haptic Rover Pose.

The success rate of the prop presentation was measured at 100% in all tests, which shows the correctness of the Haptic Rover Pose calculation. It should be noted that this success also depends on the free area in front of the user, as interaction in a space smaller than the minimum requirements prevents the robot from finding any Haptic Rover Pose. For example, if the user faces a wall when a Haptic Request is made, the distance to the wall in the interaction direction should be larger than ≈ 2.5 m (including wall inflation). For example, refer to Haptic Rover Pose #6 on Fig. 5.

5.2.3 Escape behavior

The frequency of the possible collisions was higher with the Chase behavior (26 incidents in 100 trials) than with parking in the corner (10 in 100 trials) or in the center (15 in 100 trials). Note that, for the parking conditions, the robot approaches the user when he waits for the interaction in position. A collision situation might occur when the user moves to the next position, and the robot is returning to the parking spot. In Chase, collisions are more probable, since the time that the robot stays close to the user or moves together with him is longer. Over the 300 trials, we did not observe any failures of Escape. We defined the *time-to-safety* as the time from detecting an escape situation to the time when the Escape behavior is completed and the environment's safety is restored. It is measured to be in the range 1.12 to 1.87 s with the mean 1.62 s, $SD = 0.52$, see Fig. 10. This suggests that our proof-of-concept system is capable of avoiding and reacting properly to potential collisions within a short time. Of course, a real user's behavior can be more complex compared to the simulated one. We plan to address the fine-tuning of the safety measures in future work by adding extra components to the robot and defining safety guidelines and procedures.

5.3 Practical Validation and Lessons Learned

Out of safety considerations, we first recorded a real user moving in our workspace without the robot for practical validation. We then replayed the recorded movement for the robotic system and observed the behavior in the physical environment to confirm our setup's validity. Only then we allowed a trained staff member to interact with the robot directly in the presence of an observer and a rendered robot model. Similarly, we used hand-held VIVE controllers to track and visualize the user's hands. In the future, the hands will be tracked to allow direct interaction.

Overall, the system performance matches the expectations and fulfills the fundamental requirements for the behaviors. Nevertheless, we observed certain differences between the real-world system reactions to some situations compared to the simulation. For instance, the simulation assumes a perfect grip on the floor. In our real lab, the robot's wheels were slipping along the linoleum on the floor during abrupt braking. It led to overshooting and staggering behavior in attempts to accurately position the rover. That, in turn, led to the increased positioning tolerance values up to 30 cm and contributed to the increase in the obstacle inflation distance by 23 cm, from 77 cm to 100 cm, as stated before. In addition, we limited the maximum speed of the rover to half of the maximum possible speed for the Chase and Haptic behaviors, set to 1.5 $\frac{m}{s}$. We slightly decreased it for Escape to 2.5 $\frac{m}{s}$ to reflect the need for fast removal of the robot from the user's way. This speed limitation also gives the observer enough time to react by pressing the remote emergency stop and inform the user if there should be a need.

It was expected, and observed in the validation, that the current definition of the Haptic Rover Poses limits the feasibility of reaching a DEEP based on its height. Should the haptic interaction be required above 1.6 m or below 0.5 m from the floor, the IK solver might find it impossible to find a solution. This could mean that the position is theoretically out of the arm's reach, or the solver algorithm could not find a solution in the given time, which can happen if the DEEP is close to the arm's reachability limits. It is possible to reduce the

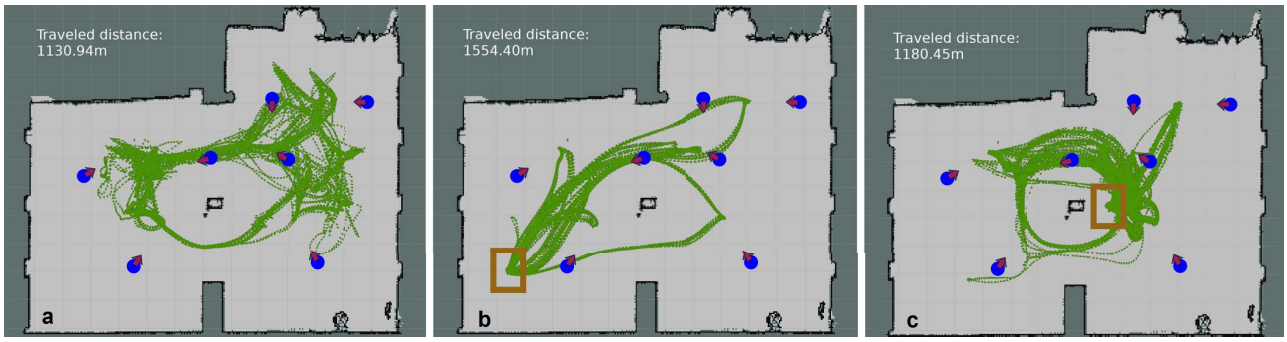

Figure 8: The robot's trajectory in each test variation (a) with Chase, (b) parked in the corner, and (c) parked in the center. The blue markers show the user position and the red arrows point to where haptic requests are made. The robot's parking positions are marked by brown rectangles.

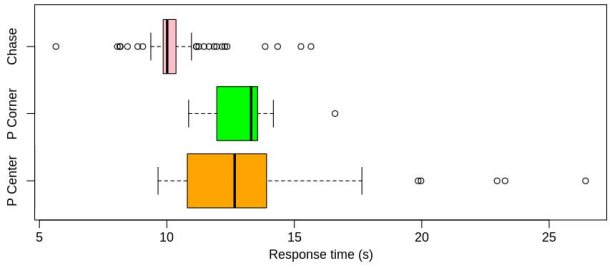

Figure 9: Boxplot of the response time in 100 haptic requests with Chase enabled, parked in the room's center and in the corner.

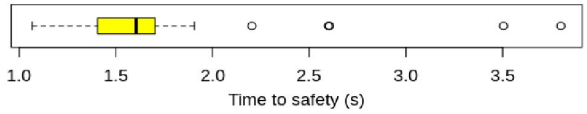

Figure 10: Boxplot of the distribution of the time-to-safety, from detecting the Escape situation to restoring the environment's safety.

distance of the Haptic Rover Poses to the DEEP further to extend the reachable area. However, for safety reasons, we decided to define them far enough from DEEP to have more reaction time in case of an Escape situation immediately after a Haptic Release signal.

Once the Haptic behavior ends, the state machine returns to Chase. Depending on which Haptic Rover Pose was picked, the robot might immediately find itself in an Escape situation, causing another state transition. This is easy to observe with a real user, which is beneficial safety-wise. At the same time, the Escape might be felt by the user due to the vibrations caused by the robot, which might cause a break in presence. We plan to fine-tune the robot's reaction in the future. We also tested the Escape by actively chasing the robot and intentionally triggering it for 30 minutes. We simulated a "curious" user that performs many head rotations during locomotion and interaction. Our observations suggest that unexpected user behavior will be handled correctly and timely. One limitation is the currently lowered Escape speed and the floor grip, as a fast-moving user might trigger a sequence of Escapes.

One of the important lessons learned is about the deployment model of the ROS packages and nodes. Since ROS allows a distributed architecture, it is possible to run the robot's nodes on several PCs, including the robot's onboard computer, which is very helpful for initial debugging. However, distributing nodes on different machines can lead to new challenges that impact the robot's general behavior, especially localization, and navigation. The delay introduced by the distributed deployment, even using a low-latency WiFi 6 connection, can influence the accuracy of the localization. Hence the nodes for the navigation and obstacle avoidance on a remote PC might receive the old position data of the robot or the environment's objects, even in a magnitude of 100 *ms*. This can cause wrong reactions and path readjustments by the navigation subsystems. Finally, we concluded that all operation-critical packages must run on the robot's computer exclusively, while the visualization packages might still run externally on remote machines. In addition, the robot's computer has lower computational power and memory than a VR-ready PC. Thus computationally expensive algorithms might operate with a noticeable delay even on a dedicated lightweight Unix system. CPU plays an important role in path planning and collision avoidance nodes. The amount of RAM is particularly crucial for the localization system and IK solvers, and we plan to increase it.

As the target application of our system is the support of the early stage design phase with haptics for arbitrary 3D sketches created during run-time, we designed the Cobodeck with such interaction in mind to facilitate a unification of various hand interactions discussed in Sect. 2. Our robot can potentially provide various types of interaction thanks to a force-torque sensor, a two-fingered gripper, and a substantial arm payload that allows using interchangeable end-effectors. The possible interaction types range from continuous haptics for large objects such as walls as well as fine feedback for design peculiarities, to material simulations with the help of physics-based dynamic force feedback. Integrating our haptic system into a VR application is uncomplicated and requires only two plugins.

6 Conclusion

In this paper, we presented a novel algorithmic approach for close human-robot interaction addressing the unique challenges of VR. We implemented it in our proof-of-concept large-scale haptic VR system with an autonomous mobile cobot. Our system supports free locomotion by the user and can provide encounter-based haptics over an arbitrary large area in 10 *s* on average. Aiming for large area VR, our system can also function in small workspaces, but no less than 6 *m by* 6 *m*. We designed a state machine, rapidly transitioning between different states as required by situation. We show that the Chase approach is more efficient compared to the robot's assigned parking positions regarding the haptic prop's availability and at least as efficient considering the traveled distance. Our system also supports a use case when the user is unaware of the robot. For the user's safety, we proposed the collision prevention Escape behavior that operates with a mean time-to-safety of 1.6 *s*.

In the future, we plan to optimize the overall system performance and further improve user safety by developing a more complex VR-oriented concept based on established industry standards. Finally, we aim to implement advanced haptics that will also reflect the elastic response of materials for large simulated objects.

Acknowledgments

This work has been funded by the grant F77 (SFB "Advanced Computational Design", SP 5) of the Austrian Science Fund FWF. Special thanks to M. Wimmer, J. Füssl, I. Kovacic and their teams.

REFERENCES

[1] M. Abdullah, M. Kim, W. Hassan, Y. Kuroda, and S. Jeon. Hapticdrone: An Encountered-Type Kinesthetic Haptic Interface with Controllable Force Feedback: Example of Stiffness and Weight Rendering. In *2018 IEEE Haptics Symposium (HAPTICS)*, pages 334–339, Mar. 2018. ISSN: 2324-7355.

[2] P. Abtahi, B. Landry, J. J. Yang, M. Pavone, S. Follmer, and J. A. Landay. Beyond The Force: Using Quadcopters to Appropriate Objects and the Environment for Haptics in Virtual Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, Glasgow Scotland Uk, May 2019. ACM.

[3] M. Al-Sada, K. Jiang, S. Ranade, M. Kalkattawi, and T. Nakajima. Hapticsnakes: Multi-Haptic Feedback Wearable Robots for Immersive Virtual Reality. *Virtual Reality*, 24(2):191–209, June 2020.

[4] B. Araujo, R. Jota, V. Perumal, J. X. Yao, K. Singh, and D. Wigdor. Snake Charmer: Physically Enabling Virtual Objects. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '16, pages 218–226, New York, NY, USA, Feb. 2016. Association for Computing Machinery.

[5] S. J. Biggs and M. A. Srinivasan. Haptic Interfaces. In *Handbook of Virtual Environments*. CRC Press, 2002. Num Pages: 24.

[6] E. Bouzbib and G. Bailly. "Let's Meet and Work it Out": Understanding and Mitigating Encountered-Type of Haptic Devices Failure Modes in VR. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 360–369, Mar. 2022.

[7] E. Bouzbib, G. Bailly, S. Haliyo, and P. Frey. CoVR: A Large-Scale Force-Feedback Robotic Interface for Non-Deterministic Scenarios in VR. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, pages 209–222, New York, NY, USA, Oct. 2020. Association for Computing Machinery.

[8] E. Bouzbib, G. Bailly, S. Haliyo, and P. Frey. "Can I Touch This?": Survey of Virtual Reality Interactions via Haptic Solutions: Revue de Littérature des Interactions en Réalité Virtuelle par le biais de Solutions Haptiques. In *Proceedings of the 32nd Conference on l'Interaction Homme-Machine*, IHM '21, pages 1–16, New York, NY, USA, Jan. 2022. Association for Computing Machinery.

[9] T. Cakmak and H. Hager. Cyberith Virtualizer: A Locomotion Device for Virtual Reality. In *ACM SIGGRAPH 2014 Emerging Technologies*, SIGGRAPH '14, page 1, New York, NY, USA, July 2014. Association for Computing Machinery.

[10] L.-P. Cheng, T. Roumen, H. Rantzsch, S. Köhler, P. Schmidt, R. Kovacs, J. Jasper, J. Kemper, and P. Baudisch. TurkDeck: Physical Virtual Reality Based on People. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*, pages 417–426, 2015.

[11] I. Choi, E. W. Hawkes, D. L. Christensen, C. J. Ploch, and S. Follmer. Wolverine: A Wearable Haptic Interface for Grasping in Virtual Reality. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 986–993, Oct. 2016. ISSN: 2153-0866.

[12] R. Dam and T. Siang. What is design thinking and why is it so popular. *Interaction Design Foundation*, pages 1–6, 2018.

[13] X. de Tinguy, T. Howard, C. Pacchierotti, M. Marchal, and A. Lécuyer. WeATaViX: WEarable Actuated TAngibles for VIrtual Reality eXperiences. In *Haptics: Science, Technology, Applications: 12th International Conference, EuroHaptics 2020, Leiden, The Netherlands*, pages 262–270, Berlin, Heidelberg, Sept. 2020. Springer-Verlag.

[14] P. M. Desmet and P. Hekkert. Special issue editorial: Design & emotion. *International Journal of Design*, 3(2), 2009.

[15] D. Fitzgerald and H. Ishii. Mediate: A Spatial Tangible Interface for Mixed Reality. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, pages 1–6, New York, NY, USA, Apr. 2018. Association for Computing Machinery.

[16] B. P. Gerkey. Amcl: Package summary. http://wiki.ros.org/amcl. [Last accessed 4-September-2022].

[17] E. J. Gonzalez, P. Abtahi, and S. Follmer. REACH+: Extending the Reachability of Encountered-type Haptics Devices through Dynamic Redirection in VR. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, pages 236–248, New York, NY, USA, Oct. 2020. Association for Computing Machinery.

[18] K. L. Han, O. K. Choi, I. Lee, I. Hwang, J. S. Lee, and S. Choi. Design and Control of Omni-Directional Mobile Robot for Mobile Haptic Interface. In *2008 International Conference on Control, Automation and Systems, ICCAS 2008*, 2008.

[19] Z. He, F. Zhu, A. Gaudette, and K. Perlin. Robotic Haptic Proxies for Collaborative Virtual Reality, Jan. 2017. arXiv:1701.08879 [cs].

[20] D. Hershberger, D. Gossow, J. Faust, and W. Woodall. Rviz: Package summary. http://wiki.ros.org/rviz. [Last accessed 4-September-2022].

[21] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell. Passive Real-World Interface Props for Neurosurgical Visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94*, volume 30, pages 452–458, New York, New York, USA, 1994. ACM Press. ISSN: 02775212.

[22] K. Hirota and M. Hirose. Development of Surface Display. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 256–262. IEEE, 1993.

[23] A. Horie, M. Y. Saraiji, Z. Kashino, and M. Inami. EncounteredLimbs: A Room-scale Encountered-type Haptic Presentation using Wearable Robotic Arms. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 260–269, Mar. 2021.

[24] B. E. Ilon. Wheels for a Course Stable Selfpropelling Vehicle Movable in Any Desired Direction on the Ground or Some Other Base, Apr. 1975.

[25] B. E. Insko. *Passive Haptics Significantly Enhances Virtual Environments*. PhD thesis, University of North Carolina at Chapel Hill, 2001. ISBN: 0-493-17286-6.

[26] H. Iwata. Walking About Virtual Environments on an Infinite Floor. In *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*, pages 286–293, Mar. 1999.

[27] S. Je, H. Lim, K. Moon, S.-Y. Teng, J. Brooks, P. Lopes, and A. Bianchi. Elevate: A Walkable Pin-Array for Large Shape-Changing Terrains. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–11, New York, NY, USA, May 2021. Association for Computing Machinery.

[28] Y. Kato. A Remote Navigation System for a Simple Tele-Presence Robot with Virtual Reality. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4524–4529, Sept. 2015.

[29] Y. Kim, H. J. Kim, and Y. J. Kim. Encountered-Type Haptic Display for Large Vr Environment Using Per-Plane Reachability Maps. *Computer Animation and Virtual Worlds*, 29(3-4):e1814, 2018.

[30] T. Kohtsuka, T. Onozato, H. Tamura, S. Katayama, and Y. Kambayashi. Design of a Control System for Robot Shopping Carts. In *Proceedings of the 15th international conference on Knowledge-based and intelligent information and engineering systems - Volume Part I*, KES'11, pages 280–288, Berlin, Heidelberg, Sept. 2011. Springer-Verlag.

[31] J. Kuffner and S. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2, 2000.

[32] R. Lindeman, J. Sibert, and J. Hahn. Hand-Held Windows: Towards Effective 2d Interaction in Immersive Virtual Environments. In *Proceedings IEEE Virtual Reality*, pages 205–212. IEEE Comput. Soc, 1999. ISSN: 1087-8270.

[33] K.-L. Low, G. Welch, A. Lastra, and H. Fuchs. Life-Sized Projector-Based Dioramas. In *Proceedings of the ACM symposium on Virtual reality software and technology - VRST '01*, page 93, New York, New York, USA, 2001. ACM Press.

[34] A. Maekawa, S. Takahashi, M. Y. Saraiji, S. Wakisaka, H. Iwata, and M. Inami. Naviarm: Augmenting the Learning of Motor Skills using a Backpack-type Robotic Arm System. In *Proceedings of the 10th Augmented Human International Conference 2019*, pages 1–8, Reims France, Mar. 2019. ACM.

[35] W. McNeely. Robotic Graphics: A New Approach to Force Feedback for Virtual Reality. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 336–341. IEEE, 1993.

[36] V. R. Mercado, F. Argelaguet, G. Casiez, and A. Lécuyer. Watch out for the Robot! Designing Visual Feedback Safety Techniques

When Interacting With Encountered-Type Haptic Displays. *Frontiers in Virtual Reality*, 3, 2022.

[37] V. R. Mercado, T. Howard, H. Si-Mohammed, F. Argelaguet, and A. Lécuyer. Alfred: the Haptic Butler On-Demand Tangibles for Object Manipulation in Virtual Reality using an ETHD. In *2021 IEEE World Haptics Conference (WHC)*, pages 373–378, July 2021.

[38] V. R. Mercado, M. Marchal, and A. Lécuyer. "Haptics On-Demand": A Survey on Encountered-Type Haptic Displays. *IEEE Transactions on Haptics*, 14(3):449–464, July 2021.

[39] M. Munaro, S. Ghidoni, D. T. Dizmen, and E. Menegatti. A Feature-Based Approach to People Re-Identification Using Skeleton Keypoints. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5644–5651, May 2014. ISSN: 1050-4729.

[40] S. Nishimura, H. Takemura, and H. Mizoguchi. Development of Attachable Modules for Robotizing Daily Items -Person Following Shopping Cart Robot-. In *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1506–1511, Dec. 2007.

[41] N. Nitzsche, U. D. Hanebeck, and G. Schmidt. Design Issues of Mobile Haptic Interfaces. *Journal of Robotic Systems*, 20(9):549–556, 2003.

[42] M. Ortega and S. Coquillart. Prop-Based Haptic Interaction with Co-Location and Immersion: An Automotive Application. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications, HAVE '05*, pages 23–28. IEEE, 2005.

[43] S. Parsons. Probabilistic robotics by sebastian thrun, wolfram burgard and dieter fox, mit press, 647 pp., isbn 0-262-20162-3. *The Knowledge Engineering Review*, 21(3):287–289, 2006.

[44] R. A. Pavlik, J. M. Vance, and G. R. Luecke. Interacting With a Large Virtual Environment by Combining a Ground-Based Haptic Device and a Mobile Robot Base. In *Volume 2B: 33rd Computers and Information in Engineering Conference*, page V02BT02A029, Feb. 2014.

[45] V. L. Popov, S. A. Ahmed, N. G. Shakev, and A. V. Topalov. Detection and Following of Moving Targets by an Indoor Mobile Robot using Microsoft Kinect and 2D Lidar Data. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 280–285, Nov. 2018.

[46] O. Robotics. Gazebo: Homepage. https://gazebosim.org/. [Last accessed 4-September-2022].

[47] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram. Efficient trajectory optimization using a sparse model. In *2013 European Conference on Mobile Robots*, pages 138–143, Sept. 2013.

[48] E.-L. Sallnäs, K. Rassmus-Gröhn, and C. Sjöström. Supporting Presence in Collaborative Environments by Haptic Force Feedback. *ACM Transactions on Computer-Human Interaction*, 7(4):461–476, Dec. 2000. ISBN: 1073-0516.

[49] P. Schillinger. Flexbe: The flexible behavior engine. http://philserver.bplaced.net/fbe/. [Last accessed 4-September-2022].

[50] P. Schillinger, S. Kohlbrecher, and O. von Stryk. Human-Robot Collaborative High-Level Control with an Application to Rescue Robotics. In *IEEE International Conference on Robotics and Automation*, 2016.

[51] R. Suzuki, H. Hedayati, C. Zheng, J. L. Bohn, D. Szafir, E. Y.-L. Do, M. D. Gross, and D. Leithinger. RoomShift: Room-scale Dynamic Haptics for VR with Furniture-moving Swarm Robots. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–11, New York, NY, USA, Apr. 2020. Association for Computing Machinery.

[52] R. Suzuki, R. Nakayama, D. Liu, Y. Kakehi, M. D. Gross, and D. Leithinger. LiftTiles: Constructive Building Blocks for Prototyping Room-scale Shape-changing Interfaces. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '20, pages 143–151, New York, NY, USA, Feb. 2020. Association for Computing Machinery.

[53] R. Suzuki, E. Ofek, M. Sinclair, D. Leithinger, and M. Gonzalez-Franco. HapticBots: Distributed Encountered-type Haptics for VR with Multiple Shape-changing Mobile Robots. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 1269–1281, Oct. 2021.

[54] U. Technologies. Unity robotics hub. https://github.com/Unity-Technologies/Unity-Robotics-Hub. [Last accessed 4-September-2022].

[55] S.-Y. Teng, C.-L. Lin, C.-h. Chiang, T.-S. Kuo, L. Chan, D.-Y. Huang, and B.-Y. Chen. TilePoP: Tile-type Pop-up Prop for Virtual Reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, pages 639–649, New York, NY, USA, Oct. 2019. Association for Computing Machinery.

[56] H.-R. Tsai, J. Rekimoto, and B.-Y. Chen. ElasticVR: Providing Multi-level Continuously-Changing Resistive Force and Instant Impact Using Elasticity for VR. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–10, New York, NY, USA, 2019. Association for Computing Machinery. event-place: Glasgow, Scotland Uk.

[57] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks. Walking > Walking-in-Place > Flying, in Virtual Environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*, pages 359–364, New York, New York, USA, 1999. ACM Press. ISSN: 00978930.

[58] E. Vonach, C. Gatterer, and H. Kaufmann. VRRobot: Robot Actuated Props in an Infinite Virtual Environment. In *Proceedings of IEEE Virtual Reality 2017*, pages 74–83, Los Angeles, CA, USA, 2017. IEEE.

[59] E. Vonach, C. Schindler, and H. Kaufmann. StARboard & TrACTOr: Actuated Tangibles in an Educational TAR Application. *Multimodal Technologies and Interaction*, 5(2):6, Feb. 2021. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[60] C. Wang, X. Liu, X. Yang, F. Hu, A. Jiang, and C. Yang. Trajectory Tracking of an Omni-Directional Wheeled Mobile Robot Using a Model Predictive Control Strategy. *Applied Sciences*, 8(2):231, Feb. 2018. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[61] Y. Wang, Z. T. Chen, H. Li, Z. Cao, H. Luo, T. Zhang, K. Ou, J. Raiti, C. Yu, S. Patel, and Y. Shi. MoveVR: Enabling Multiform Force Feedback in Virtual Reality Using Household Cleaning Robot. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–12, New York, NY, USA, 2020. Association for Computing Machinery. event-place: Honolulu, HI, USA.

[62] H. Wu, W. Xu, B. Yao, Y. Hu, and H. Feng. Interacting Multiple Model-Based Adaptive Trajectory Prediction for Anticipative Human Following of Mobile Industrial Robot. *Procedia Computer Science*, 176:3692–3701, Jan. 2020.

[63] M. Wu, S.-L. Dai, and C. Yang. Mixed Reality Enhanced User Interactive Path Planning for Omnidirectional Mobile Robot. *Applied Sciences*, 10(3):1135, Jan. 2020. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[64] K. Yamaguchi, G. Kato, Y. Kuroda, K. Kiyokawa, and H. Takemura. A Non-grounded and Encountered-type Haptic Display Using a Drone. In *Proceedings of the 2016 Symposium on Spatial User Interaction - SUI '16*, pages 43–46, New York, New York, USA, 2016. ACM Press. Series Title: SUI '16.

[65] S. Yamaguchi, H. Shionoiri, T. Nakamura, and H. Kajimoto. An Encounter Type VR System Aimed at Exhibiting Wall Material Samples for Show House. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*, ISS '18, pages 321–326, New York, NY, USA, Nov. 2018. Association for Computing Machinery.

[66] Y. Yokokohji, R. L. Hollis, and T. Kanade. What You Can See Is What You Can Feel - Development of a Visual/Haptic Interface to Virtual Environment. In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pages 46–53, 1996.

[67] Y. Yoo and W. Chung. Detection and Following of Human Legs Using the SVDD (Support Vector Data Description) Scheme for a Mobile Robot with a Single Laser Range Finder. In *International Conference on Electrical, Control and Computer Engineering 2011 (InECCE)*, pages 97–102, June 2011.