2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Virtual Borders in 3D: Defining a Drone's Movement Space Using Augmented Reality

Malte Riechmann[1], André Kirsch[1], Matthias Koenig[2] and Jan Rexilius[1]

***Abstract*— Robots are increasingly finding their way into home environments, where they can assist with household tasks like vacuuming or surveilling. While the robots can navigate on their own, users might not want them to go everywhere or not in a specific way. For example, users might not want a drone to fly over a table where important letters and the newspaper are stored, even though it is the shortest path to the goal. Therefore, an application is required, that is easy to learn and to apply even for inexperienced users.**

In this paper, we present a framework that uses a tablet as augmented reality (AR) device to modify a robot's movement space in 3D. A user can define virtual borders in the real world with the tablet and add them to a map, changing the navigational behavior of the robot. The framework is evaluated by a user study with inexperienced participants that verifies our approach. Further analyses show, that even complex scenarios can be covered with our framework.

I. Introduction

In modern society, the influence of robots on our daily life is growing. Good examples of that are vacuuming or lawn mowing robots that are increasingly deployed. Robots working in proximity to humans must meet certain requirements to increase the safety and the comfort of their human counterparts. One tool to achieve that are virtual borders [1]. Virtual borders define so-called keep-in and keep-out areas for robots, i.e., areas a robot is not allowed to enter or leave. People without any robotic background should be able to teach such borders to a robot without much effort. There are several solutions to this problem for robots acting in 2D. In these solutions, arbitrary polygons are determined by defining a chain of vertices on the ground plane using various interaction devices.

However, little work was spent on virtual borders for robots working in 3D. The commonly used virtual borders aim at robots working on the ground, thus 2D virtual borders are sufficient here. But, they are limited in their ability to restrict the movement space of robots working in 3D. For example, when an area on the ground is excluded from the robot, flying drones could be allowed to cross the area when flying high enough. To this end, we propose a 3D approach for virtual borders employing an appropriate interaction method. As pointed out, the methods used in previous works to define virtual borders in 2D are not applicable as the complexity of the borders in 3D space grows.

[1]Campus Minden, Bielefeld University of Applied Sciences and Arts, 32427 Minden, Germany, [2]University of Southern Denmark, Sønderborg, Denmark Contact:malte.riechmann@hsbi.de

Acknowledgements: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 465783762.

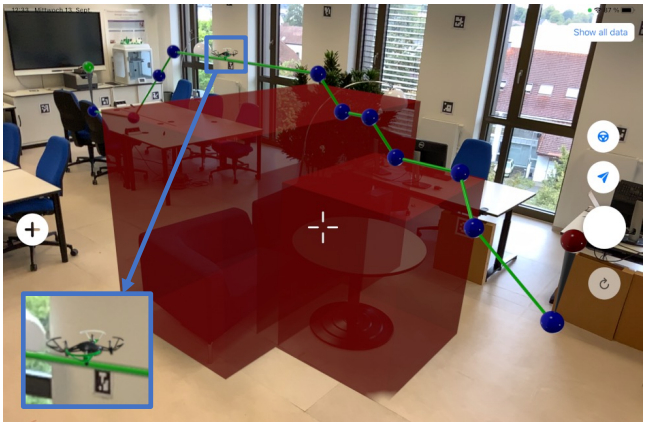

Fig. 1. Screenshot of the proposed AR application. The user restricts the drone's workspace by inserting multiple cubes defining virtual borders. The result is visualized in the AR application and the drone avoids the marked area as shown by the generated path.

Further, placing single vertices in an open 3D space is often challenging due to missing reference points to determine an accurate position. In addition to positioning vertices, the shape also becomes more complex. The number of faces of one border grows from one to many, with more vertices and edges. Constructing complex shapes by individually positioning all vertices can be exhaustive. Further, one vertex in 3D can be connected to more than two other vertices so that the edges cannot be simply determined by the order of the vertices. A user would have to define the edges manually.

To address these challenges, we present a new method to define virtual borders in 3D employing augmented reality (AR). Our method is based on combining simple 2D and 3D shapes into a 3D shape of any complexity. Fig. 1 shows an example of a complex virtual border in 3D that can be defined with our solution.

The contribution of this work is twofold. (1) First, we present an interaction method to accurately define arbitrary virtual borders in 3D. We use an AR application as an interface because it can visualize virtual objects embedded in the real world which can help a user to determine their position and dimensions. Our interaction approach is both simple and easy to learn, and still allows defining complex virtual borders. Users with little to no knowledge of AR and robotics can instruct a robot to avoid certain areas without much effort. (2) The second part of our contribution is a framework for virtual borders integration. The framework includes a robot moving in 3D, a planning unit for the robot with the map that should contain the borders, and the device for the AR interaction.

We use the Apple iPad Pro 2022 as AR device as it meets our requirements: First, many people know how to use a tablet. Second, the iPad has an additional LiDAR sensor, which increases the quality of the AR experience. The RGB-D images are used to improve the visual odometry to accurately track the iPads position in 6-DoF. The depth information is also used to occlude virtual objects behind real-world object. That can help to determine the position of a virtual object in the AR application.

II. Related Work

AR is a popular tool to improve human-robot interactions, as surveys have shown the vast amount of possible applications [2]–[4]. For drones, this is particularly noticeable in the field of navigation. Examples for directly controlling drones using AR are DroneARchery [5], an application for a head-mounted device to generate flight paths of a swarm of drones by simulating archery, and a mobile phone touch interface [6]. Other applications are showing a drone's motion intent by showing its path in AR [7] and making walls see-through by projecting the virtual environment, a drone is operating in, onto a wall [8].

a) Change in navigational behavior: Besides the AR approach, there exist many different interfaces and interactions that can change the navigational behavior of mobile robots in implicit and explicit ways. Implicit navigational behavior deals with the presence of humans and automatically alters the navigation paths of robots accordingly [9]–[12]. However, humans might want to instruct the robot to avoid certain areas, such as expensive carpets when vacuuming. To this end, the user needs to define the working space explicitly. A typical approach for lawn mowing robots is to use boundary wires [13]. An alternative is to use colored tape [14]. These boundaries are easy to install even for non-expert users, but require physical objects in an environment.

An alternative is to use virtual borders that are visible to the robot but not to the human, if not using an appropriate device. So, special user interfaces are required to work with virtual borders. There are some interfaces that use AR to define 2D virtual borders on the floor, for example by Sprute et al. [15] and Papcun et al. [16]. AR has also been successfully used for inputting navigation goals [17] or tagging locations [18]. There are further enhancements for defining virtual borders, like a system that generates suggestions for borders [1] or additional behavioral constrains in some virtual areas, e.g. slower/faster motion speed [19].

All aforementioned user interfaces have in common that they allow defining locations and borders in 2D. They are not designed for teaching virtual borders in 3D. Out of the described approaches, especially AR promises to be adaptable for this task. This is shown by Hoang et al. [20] who employed AR for human-robot collaboration in shared workspaces. The humans part can manipulate virtual barriers using hand gestures and voice commands to prevent a robot arm from entering specific regions.

b) Maps: In robotics, the physical world is modeled through maps, which are used, for example, in robot navigation and path planning. A typical representation is the occupancy grid map, which splits the environment into cells. Each cell contains a probability value that describes how likely a cell is occupied. Grid maps can either be 2D [21] or 3D [22]. Besides the typical use cases, Sprute et al. [15] have also used them for storing the virtual borders. But with the integration of robots operating in 3D space, such as drones, 2D grid maps reach their limits. 3D grid maps extend the map by a third dimension and come in the form of elevation maps [23], multi-level surface maps [24] or voxel maps [22]. In our case, we use voxel maps as they can precisely model the environment and are computationally efficient.

III. Problem Definition

Our goal is to change the navigational behavior of a 6 DoF mobile robot. A 3D occupancy map models the physical environment and is the basis for the navigation. The voxels are cubes with a fixed size, which are ordered in a 3D grid. The grid is placed in the physical environment, and every voxel stores an occupancy probability and color of the corresponding area. $M(x, y, z) \in [0, 1]$ denotes the occupancy probability of a voxel at the position $(x, y, z) \in \mathbb{R}^3$ and $C(x, y, z) \in [0, 1]^3$ its color.

To change the navigational behavior, we use virtual borders defined by a user. Virtual borders are non-physical borders that restrict the workspace of a mobile robot. While not directly visible to the human eye, the robot respects those regions and does not cross them. A single 3D virtual border V consists of four elements: $V = (P, F, \delta, i)$. The virtual border points $P = \{p_1, ..., p_n\}$ are a set of n points $p_j = (x_j, y_j, z_j) \in \mathbb{R}^3$, relative to the map origin. They mark the boundaries of the border. The second component stores the connections of these points as a set of faces $F = \{f_1, .., f_m\}$. Each face consists of a list of indices $f_j = \{i_1, ..., i_k\}$, referring to the points in P. The order of the elements determines the edges connecting the points. All points of one face must be within the same plane. The points in f_j are ordered counterclockwise, so that the normal of the face points towards the outside of the virtual border. The sum of all faces in F do not need to form a closed shape to make the virtual borders more flexible. The area that should be excluded is given by $i \in \{0, 1\}$. This element is a flag whether the outside ($i = 0$) or the inside ($i = 1$) of the object should be excluded. In terms of not closed shapes, that means that for $i = 0$ the relevant area is the intersection of all points on the side opposite to the normal one of the faces. The third component $\delta \in [0, 1]$ is the occupancy probability value, that should be written into the map.

IV. Implementation

The proposed framework uses AR to visualize and control a robot's workspace in 3D. The framework consists of three main components: An AR application as frontend, a mapping module, and a robot with its motion controller. The framework is designed to encapsulate the modules to make it flexible and allow replacing individual modules easily. The center of the framework is the mapping module which is

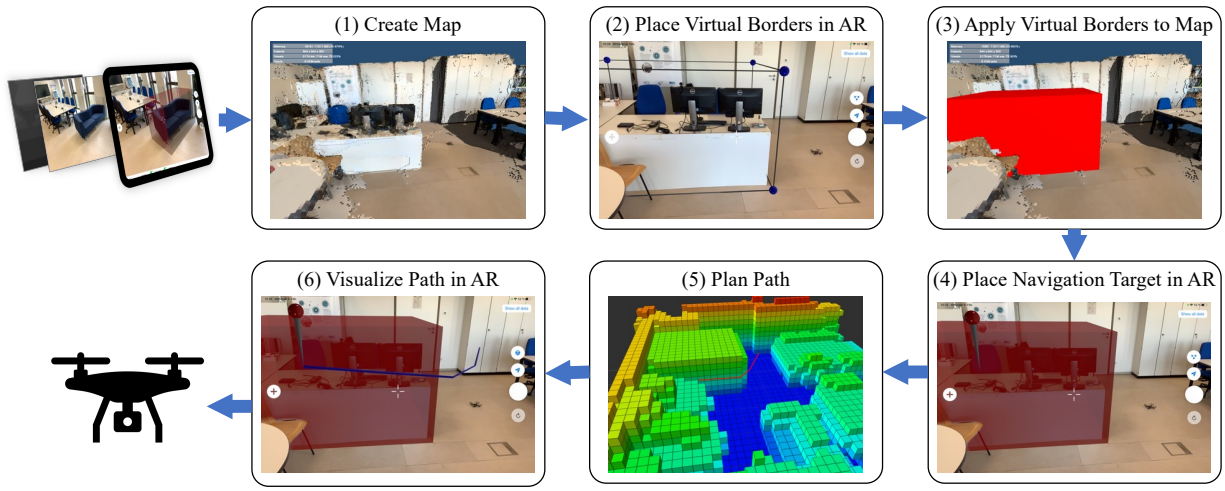

Fig. 2. The general procedure to place virtual borders and send the drone to a target position.

responsible for managing and creating the shared map used by the other modules. It provides interfaces for updating and requesting the map and inserting virtual borders. In our case, the data from the AR device is used to construct the map, but using multiple input devices simultaneously is possible as well. For example, if the robot also captures depth data, it could be used to generate a more holistic map. We also allow the user to place navigation markers for the robot and visualize the resulting path in AR to simulate how the virtual borders affect the navigational behavior of the robot. All interfaces expect the data in a specific format, such as the virtual borders as described in Sect. III. Modules can be exchanged with other modules that implement these interfaces. A visualization of the overall system architecture is given in Fig. 2.

A. The AR Application

The AR application is the access point of the user to the abstract representation of the world as seen by the robot. It should allow a non-expert user to accurately define virtual borders. We use a handheld device because due to the availability of smartphones and tablet computers, most people know how to use them. To achieve our goal, the application must first visualize important virtual objects and allow the user to interact with them. In our scenario, those objects are the virtual borders, a navigation target and a path to this target. The user can create virtual borders to modify the underlying map of the robot. Furthermore, a navigation target can be set or modified and a path from the robot to the target can be shown.

a) Visualization: The application is required to visualize the virtual objects as they only exist virtually and cannot be seen by the user otherwise. In order to accurately position a virtual object in the real world, the object needs to be embedded into the image of the actual room. More specifically, the object is put in the camera image at their position in the real world. Virtual objects behind real objects are occluded using the depth camera. We use two different representations for the virtual borders, depending on their status. During the construction phase, step 2 in Fig. 2, they are represented as wireframes. Wireframes occlude little of the underlying camera image, while showing the dimension of the borders. The vertices are marked with blue spheres, which are the elements the user can interact with. Once a border is applied to the map (Fig. 2, step 3), the wireframe representation changes to semi-transparent planes, as shown in step 4 of Fig. 2. The transparency allows the user to still see the underlying world. The color of the plane indicates whether it is an area the robot is not allowed to enter (red) or not allowed to leave (green). For the navigation target, we use the design of a pin, also shown in step 4 of Fig. 2. The tip of the pin marks the target position. The user can request a path from the robot's navigation planner by pressing the topmost button of the four round buttons on the right side of the screen (Fig. 2, step 4). After planning the path (Fig. 2, step 5), the navigation planner sends the result back to the application, where it is represented as a blue line, as shown in step 6 of Fig. 2. Pressing the same button as before, when a path is present, is the signal for the robot to follow the path to the target.

b) Interaction: The overall concept of our interaction is based on combining simple shapes into complex ones. As mentioned in the introduction, constructing 3D shapes by defining each vertex and edge individually can become exhaustive for complex shapes. Also, it is hard to determine an accurate position of a vertex in the air because of missing reference points. So, we propose to use simple shapes as basis to construct complex virtual borders. Rather than defining every vertex and edge, the shape as a whole is defined. The edges of a shape are predefined and moving vertices results in an update of the connected elements. Furthermore, specific shapes have characteristics, that can help a user determine the position of its vertices and understand the applied changes. For example, in case of a cube, moving one vertex also moves the connected vertices to maintain the orthogonality. When working with a cube on the ground plane, the user knows that the vertices in the air will always be vertically

above the vertices on the ground.

There are two ways to combine the simple shapes into the final bigger shape. The first one takes the union over the inside of all shapes and marks it as occupied, while the second one takes the complement of that union. The first method keeps the robot out of the virtual borders, while the second one makes the robot stay inside them. That can be used to define flight corridors that a drone must take. This way of combining the shapes is straightforward for closed shapes such as the previously mentioned cube. However, using non closed shape has some important characteristics. For example, when using a single 2D plane as basic shape, inserting it in the map will mark the whole area on one side as occupied. Combining multiple planes using the first method allows to define huge areas quickly the robot should avoid. But using the second interaction methods for planes allows defining arbitrary convex shapes. In that case each plane defines one side of that shape.

For the tablet interaction, we implemented a grasping mechanism to modify the basic shapes. In order to grasp and move a vertex, a crosshair placed at the center of the application must be moved over the sphere marking the vertex. The sphere changes its color when the crosshair hovers over it to indicate that it can be grasped. Pressing the grasp button (the plain white button in step 2 of Fig. 2) grasps the currently hovered vertex. The relative position of the vertex to the AR device is fixed while it is grasped. So, moving the tablet makes the vertex move accordingly. The vertex follows the device motion until the grasp button is released and remains at its new position afterward. There are additional markers to make specific tasks easier. The first one is a root marker. Grasping this marker allows moving the object as a whole. There are two modes of how the object follows the device motion. In the first mode, the object only follows the device motion but does not change its orientation. The second mode makes the object follow both, the device's motion and rotation. In addition to the root marker, there are three rotation markers. These markers can be used to rotate the object along one of the three axes.

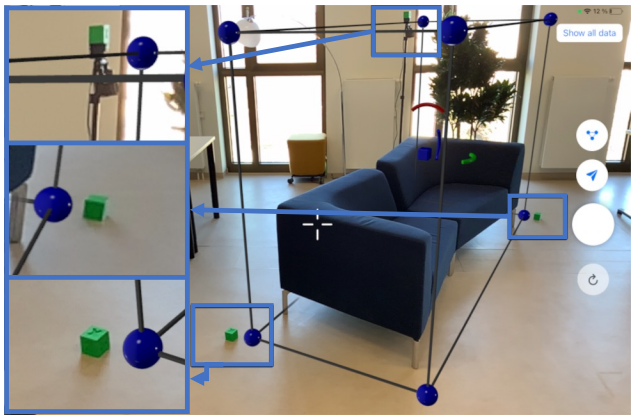

Fig. 3. Experimental setup. Several real green cubes are placed in the scene as reference. This allows us to guide users during virtual border placement experiments as well as to assess the placement quality.

B. Mapping Module

The mapping module manages the map, shared between robot and AR application. The module is responsible for creating and updating the map and inserting virtual borders / virtual objects into the map. The environment is represented as a probability-based occupancy map, as it is well suited for navigation tasks performed by the robot.

a) Construction: The mapping module takes either a point cloud or the raw input of an RGB-D device to update the map. In case of the raw input, the position, orientation, RGB-D image and intrinsic camera parameters are required. The RGB-D image is transformed to a point cloud using the intrinsic matrix and the tablet's pose. The occupancy probability of a voxel $M(x, y, z)$ that contains a point of the point cloud is increased. Additionally, the color of that point is added to the average color of the voxel $C(x, y, z)$. Further, the occupancy probabilities of all voxels in between the point and the camera are decreased.

b) Insertion of Virtual Borders: The voxels inside the virtual borders need to be set to a given probability value. The polygons forming the virtual borders are split into convex polygons to identify the voxels laying within the virtual borders. Per definition, all points inside a convex polygon lay on the same side of the plane defined by a face of the polygon. So, to identify the voxels inside the convex sub-polygons, they must be on the inner side of all faces of that sub-polygon. This test is performed for every voxel of every convex polygon.

C. Drone controller

The drone controller is implemented as ROS 2 package that is responsible for path planning and communicating with the drone. We use a Ryze Tello EDU drone, which includes an SDK with a set of commands to remotely control the drone and receive sensor data from the drone, e.g. battery level, height, camera feed.

a) Path planning: One major part of the package is path planning for autonomous navigation. The input variables of the path planning algorithm are the initial pose of the robot, the goal position and a 3D occupancy map including virtual borders. We use the A* algorithm for path planning to find the shortest path from the robot position to a given goal position in the map. We apply additional post-processing steps to reduce redundant waypoints and to find partial ways if the goal is not reachable. However, the path planner can be easily exchanged if required. The new planner must listen to the ROS topic for navigation target. Then it can request the current map and position of the robot to plan the path. Once it is finished, the planner must simply publish the coordinates of the path's waypoints to another ROS topic. This way, the best planner for a certain task and robot can be applied.

b) Commanding the drone: The drone controller provides ROS topics for taking off, landing or sending a single goal (ROS topic */move_to*) which are directly forwarded to the drone. Furthermore, it provides a ROS topic to send the generated path to, after a user has accepted it in the AR application. For each position in the path, the drone controller

Fig. 4. Three use cases for more complex scenarios of virtual borders. Image (a) and (b) show a combination of multiple cuboid areas the robot should avoid. Image (c) shows two defined working areas connected by a flight corridor the robot should not leave. Area boundaries are again defined by real green cubes. Furthermore, a projector image is used as reference.

sends a command to the *move_to* topic and waits for its successful execution. Then, the next command is sent. This is repeated until the final goal of the drone is reached.

V. Evaluation

In the following, we provide quantitative and qualitative results of using the proposed framework. We consider the three criteria *accuracy*, *complexity* and *correctness*. Due to the lack of baseline methods to compare our approach to, a more in-depth discussion about the strengths and limitations of the proposed system is given in Section VI. All experiments were performed in a $10m \times 8m$ lab environment. The map and paths are generated during the experiments and we assume that the drone can accurately localize itself and follow a given path. We have chosen to use two basic shapes, a 3D cube and a 2D plane. As described in Sect. IV-A the characteristics of a cube can help the user to determine the positions of its vertices and the plane can be used to construct arbitrary shapes. How such shapes can be combined to construct virtual borders is shown in Sect. V-B.

A. Accuracy

The goal of this section is to answer the question: *Can an inexperienced user accurately define virtual borders in 3D space?* To evaluate this, a user study was performed with ten participants. Before performing the experiment, the participants were asked to answer a questionnaire about their experience with AR and robotics on a 5-point Likert scale. Afterward, every participant was asked to place two virtual borders in two separate runs, one for each basic shape, the cube and the plane. The participants were then asked to rate their experience with the framework in terms of how *intuitive*, *natural*, *easy to learn* and *comfortable* the interaction felt on a 5-point Likert scale.

To evaluate how accurate a participant can position the virtual borders, they should try to place the border in AR as close to real-world markers as possible. These markers are small green cubes that are also visible in the map of the robot. The test setup with the green maker cubes next to the virtual cuboid is shown in Fig. 3. We used the Jaccard Index to assess the accuracy of the user-defined virtual borders compared to the ground truth borders provided by the marker's position in the map: $J(GT, T) = |GT \cap T|/|GT \cup T|$. Here, T refers to the voxels inside the virtual borders inserted by a participant and GT to the voxels within the ground truth borders. There was no familiarization phase before running the experiments to test how the participants adapted to the interaction method.

The results of the questionnaire in the beginning show that the participants have little to no experience with AR (mean: $\mu = 2.1$). The participants are also not familiar with robotics ($\mu = 2.1$) or how modelling is done in CAD tools ($\mu = 2.2$). However, all participants are familiar with mobile devices and how they are used ($\mu = 4.7$), which makes the participants good representatives for the target group.

TABLE I
Results (mean±SD) of user study with ten participants including placement accuracy and processing time.

Virtual Object	Accuracy [in %]	Time [in s]
Cube	85.4 ± 7.6	153 ± 56
Plane	98.3 ± 1.1	102 ± 26

The results of our accuracy study are shown in Tab. I. All participants successfully managed to insert virtual borders at the pre-defined positions with high accuracy (cf. Fig.3 for experimental setup). For the first experiment, i.e. adding a virtual cube, the mean accuracy is 85.4%. Inaccuracies mostly originated from wrongly sized cubes and less from problems with the correct cube orientation. Compared to the cube, the plane experiment achieved a 12.9% higher mean accuracy – mostly due to fewer vertices of this shape, and therefore less degrees of freedom. Furthermore, the average processing time for all participants is reduced by $51s$ with a largely reduced inter-user deviation of $30s$. Reasons are the simpler task as well as the steep learning curve among the participants. For experienced users the processing time decreases even further, typically to less than one minute while maintaining or increasing placement accuracy.

In the concluding questionnaire, the participants rated the interaction method as easy to learn ($\mu = 4.3$) and intuitive ($\mu = 3.8$). Some participants mentioned that their first idea to move the virtual object was to drag it on the screen using a finger, rather than grasping the object with the tablet. However, moving the object by moving the tablet felt natural to the participants ($\mu = 4.5$). Furthermore, the interaction felt comfortable ($\mu = 4.3$), especially holding the tablet with two hands, but some participants noted that they had to move a lot

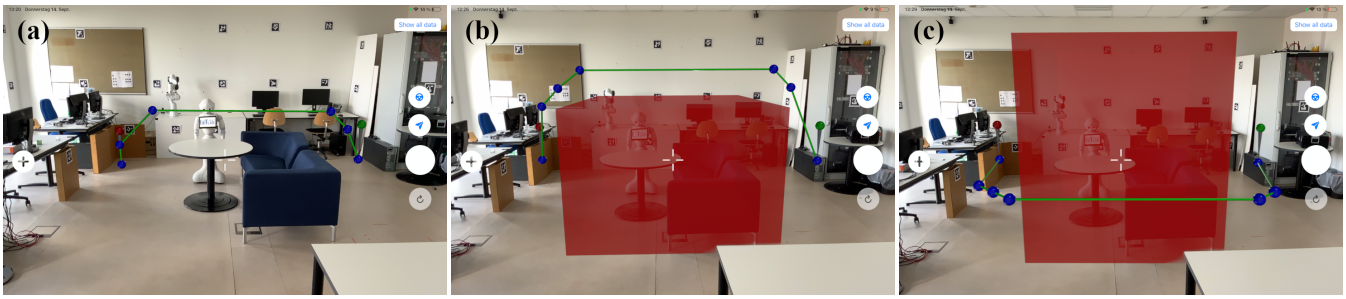

Fig. 5. Capabilities of our proposed approach for a typical use-case including path changes due to virtual border placement in a scene. The initial path in (a) is close to the sitting area. The virtual border in (b) enforces a higher path over the area, while the taller border in (c) results a path around the area.

in the test environment. A factor that may enhance this effect is the tablet's small field of view, which makes it challenging to see the entire scene at once.

B. Complexity

The evaluation of the complexity should answer the question: *Is our approach suited to define complex virtual borders in a workspace?* We define the complexity of a virtual border as the number of arbitrary basic shapes required to construct the overall shape, and their DoF. To test the capability of our solution to define complex virtual borders, a test person was asked to define the virtual borders for three use cases with increasing complexity. The participant was given time before the experiment to get to know the interaction to reduce the learning effect across the experiments. The target areas for the virtual borders are indicated by the marker cubes used in Sect. V-A, where it is shown that it is possible to place the virtual border accurately on such markers.

The three use cases are shown in Fig. 4. For the first case (cf. Fig. 4 a), two squared areas around groups of tables should be marked. The two virtual cubes need to be moved along all three axes and rotated along the vertical *z-axis*. For the second use case (cf. Fig. 4 b), the goal was again to exclude two areas. However, the complexity of the individual shapes is increased. The first shape around a sitting area has a T-shape and must be constructed using two cubes. The other area should prevent the robot from entering the area in front of a projector. A cube must be inserted and rotated along all three axes to neatly cover this area. The last use case (cf. Fig. 4 c), requires defining two workspaces which are connected by an air corridor. The first workspace is a whole section of the room, which can be divided from the rest of the room by a vertical plane. The second workspace has a triangular shape that must be constructed using three vertical planes. This shows how arbitrary shapes can be defined using planes only. To connect the two areas, a cube should be placed and rotated around the *z-axis* so that it reaches inside both areas. All planes must be moved and rotated along all three axes.

The results show that our solution allows defining complex virtual borders using only basic shapes. The test person was able to define the virtual borders for all three use cases. This shows a user can exclude a robot from complex areas, by combing several basic shapes. However, with a growing complexity, also the time to define the virtual border rises.

C. Correctness

The correctness evaluation should answer the question: *Does adding virtual borders change the navigational behavior of the drone?* A drone should avoid virtual borders in its workspace and plan its path accordingly. Fig. 5 demonstrates the capabilities of our approach for a typical use-case. Without virtual borders, the path planner computes the shortest path, which results in a drone flying closely above a sitting area (cf. Fig. 5 a)). Adding a virtual border prevents the drone from flying too close to the area (Fig. 5 b)). Using our framework, a user can also interactively change the virtual box height and visualize different path planning results. For example, the box in Fig. 5 c) is further increased so that the resulting path leads around both the table and the couch. Our results show that the navigational behavior is correctly adapted to avoid the virtual borders.

VI. Discussion and Conclusion

In this work, we developed a framework which exploits an AR application running on an RGB-D device to visualize and modify the workspace of a 6 DoF robot. To this end, a user can define virtual borders that are incorporated in the occupancy map of the robot. This method allows non-expert users to exclude robots from certain areas and adapt the navigational behavior of the robot in human-centered environments, as shown in the evaluation. Contrary to other approaches for 2D virtual borders, we construct the 3D border by combining basic 2D and 3D shapes into more complex shapes. Our experiments have shown that virtual borders with various shapes can be easily defined without the user having to worry about defining ever vertices explicitly or the connections between them. Furthermore, our results showed a steep learning curve for placing and adjusting virtual borders for all participants. We also received positive feedback for the AR application, assessed with an additional questionnaire filled out by the participants, e.g. *"The applications allows a natural interaction with the vertices."* Suggestions for improvement included for example to restrict the motion of vertices to coordinate axes.

Future work will focus on adding additional shapes as building blocks for virtual borders besides cubes and planes. Furthermore, additional devices such as head-mounted AR devices will be integrated in our framework.

REFERENCES

[1] D. Sprute, P. Viertel, K. Tönnies, and M. König, "Learning virtual borders through semantic scene understanding and augmented reality," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4607–4614.

[2] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in *CHI Conference on Human Factors in Computing Systems*, 2022.

[3] G. d. M. Costa, M. R. Petry, and A. P. Moreira, "Augmented reality for human-robot collaboration and cooperation in industrial applications: A systematic literature review," *Sensors*, vol. 22, no. 7, 2022.

[4] Z. Makhataeva and H. A. Varol, "Augmented reality for robotics: A review," *Robotics*, vol. 9, no. 2, 2020.

[5] E. Dorzhieva, A. Baza, A. Gupta, A. Fedoseev, M. A. Cabrera, E. Karmanova, and D. Tsetserukou, "Dronearchery: Human-drone interaction through augmented reality with haptic feedback and multi-uav collision avoidance driven by deep reinforcement learning," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2022, pp. 270–277.

[6] L. Chen, K. Takashima, K. Fujita, and Y. Kitamura, "Pinpointfly: An egocentric position-control drone interface using mobile ar," in *CHI Conference on Human Factors in Computing Systems*, 2021.

[7] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2018, pp. 316–324.

[8] O. Erat, W. A. Isop, D. Kalkofen, and D. Schmalstieg, "Drone-augmented human vision: Exocentric control for drones exploring hidden areas," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 4, pp. 1437–1446, 2018.

[9] R. Möller, A. Furnani, S. Battiato, A. Härmä, and G. M. Farinella, "A survey on human-aware robot navigation," *Robotics and Autonomous Systems*, vol. 145, 2021.

[10] J. Truc, P.-T. Singamaneni, D. Sidobre, S. Ivaldi, and R. Alami, "Khaos: a kinematic human aware optimization-based system for reactive planning of flying-coworker," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4764–4770.

[11] M. Moder and J. Pauli, "Proactive robot movements in a crowd by predicting and considering the social influence," in *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2022.

[12] A. Vega-Magro, L. V. Calderita, P. Bustos, and P. Núñez, "Human-aware robot navigation based on time-dependant social interaction spaces: a use case for assistive robotics," in *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2020.

[13] "Self-propelled random motion lawnmower," Patent US3 570 227A.

[14] M. Wahby, J. Petzold, C. Eschke, T. Schmickl, and H. Hamann, "Collective change detection: Adaptivity to dynamic swarm densities and light conditions in robot swarms," in *Conference on Artificial Life*, 2019.

[15] D. Sprute, K. Tönnies, and M. König, "Virtual borders: Accurate definition of a mobile robot's workspace using augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 8574–8581.

[16] P. Papcun, J. Cabadaj, E. Kajati, D. Romero, L. Landryova, J. Vascak, and I. Zolotova, "Augmented reality for humans-robots interaction in dynamic slotting "chaotic storage" smart warehouses," in *Advances in Production Management Systems. Production Management for the Factory of the Future*, 2019, pp. 633–641.

[17] M. Gu, E. A. Croft, and A. Cosgun, "Ar point&click: An interface for setting robot navigation goals," in *International Conference on Software Reuse*, 2022.

[18] S. M. Chacko, A. Granado, A. RajKumar, and V. Kapila, "An augmented reality spatial referencing system for mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[19] Y. Zhang, C.-H. Zhang, and X. Shao, "User preference-aware navigation for mobile robot in domestic via defined virtual area," *Journal of Network and Computer Applications*, vol. 173, 2021.

[20] K. C. Hoang, W. P. Chan, S. Lay, A. Cosgun, and E. Croft, "Virtual barriers in augmented reality for safe and effective human-robot cooperation in manufacturing," in *31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2022, pp. 1174–1180.

[21] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," *IEEE International conference on Robotics and Automation*, vol. 2, pp. 116–121, 1985.

[22] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[23] M. Hebert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1989, pp. 997–1002.

[24] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.