Reality Promises: Virtual-Physical Decoupling Illusions in Mixed Reality via Invisible Mobile Robots

Mohamed Kari
Princeton University

Parastoo Abtahi
Princeton University

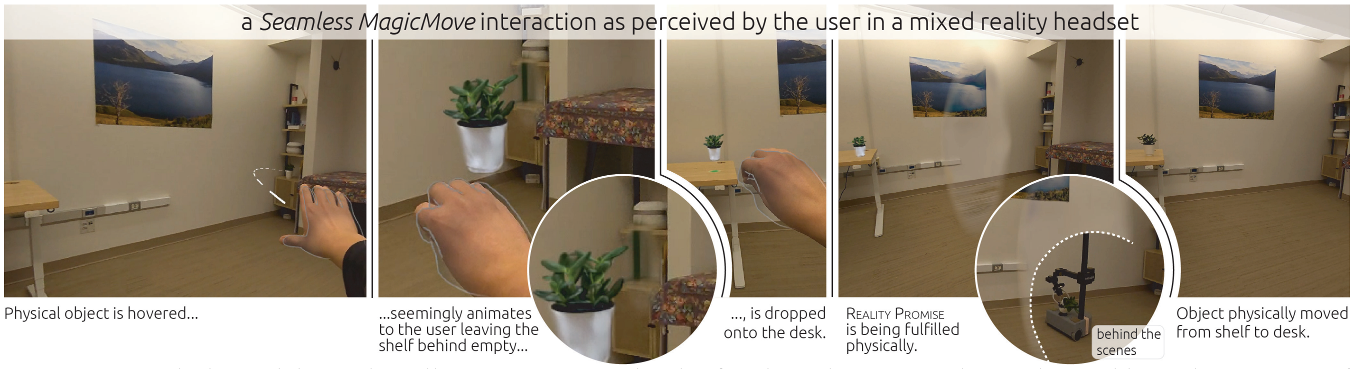
a *Seamless MagicMove* interaction as perceived by the user in a mixed reality headset
Physical object is hovered...
...seemingly animates to the user leaving the shelf behind empty...
..., is dropped onto the desk.
Reality Promise is being fulfilled physically.
behind the scenes
Object physically moved from shelf to desk.

Figure 1: Virtual-physical decoupling illusions create two levels of reality when manipulating the world. On the *experiential level* perceived by the user, objects can be manipulated in ways only virtuality affords. On the *physical level*, a robot replicates virtual manipulations without exposing itself to the user. *Please watch the accompanying video for a full impression of the experience.*

Abstract

Humans incessantly manipulate objects in their environment. Yet, mixed reality systems fall short of enabling seamless manipulations of the physical scene, constraining experiences to virtual effects. In this paper, we present the concept of Reality Promises, the mixed-reality illusion of manipulating the scene in ways only virtuality affords while secretly propagating virtual manipulations to physical reality. By decoupling virtual modes of manipulations from the physical mode of manipulation, Reality Promises create the illusion of manipulating the physical scene instantaneously, using magical forces or fantastical creatures. Concealed from the user, our system directs a mobile robot that manipulates physical objects between dynamic virtual-physical decoupling and recoupling points without revealing itself to the user. To render the robot invisible and physical objects interactable, we introduce a robot-aware 3D Gaussian splat rasterization, shading, and animation system that renders splats co-aligned with the local space into the user's passthrough view where needed. We systematically derive interaction protocols that provide cohesive end-to-end user experiences, such as materializing objects out of thin air or applying user or character-induced virtual forces to physical objects.

CCS Concepts

• **Human-centered computing → Mixed / augmented reality**; • **Computing methodologies → Computer graphics**; • **Hardware → Sensors and actuators**.

Keywords

Mixed reality; augmented reality; computer graphics; robotics;

ACM Reference Format:
Mohamed Kari and Parastoo Abtahi. 2025. Reality Promises: Virtual-Physical Decoupling Illusions in Mixed Reality via Invisible Mobile Robots. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28–October 01, 2025, Busan, Republic of Korea.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3746059.3747660

1 Introduction

Mixed reality (MR) systems enable users to experience fantastical virtual content in their known environment. However, seeing physical laws apply to the physical scene at all times inhibits MR's potential for experiences with a touch of magic. In this paper, we propose to create the illusion of breaking down the limits of physical laws while preserving the sensation of being in physical space by means of *virtual-physical decoupling illusions.* In contrast to previous works in MR [33, 39, 72], which have used perceptual manipulations to produce *visual effects* of manipulating the world, virtual-physical decoupling illusions simultaneously manipulate user perception *and* physical reality, thereby also producing *physical effects.*

Our core idea is to branch the user's perception of physical reality from the current state into a *promised reality*, which they can manipulate in ways only virtuality affords, while concurrently and concealed from the user reproducing the *physical effect* of their virtual manipulation into the physical world. A virtual manipulation creates a Reality Promise that is only resolved once the manipulatory effect is physically fully replicated. To physically fulfill the promise, an invisible mobile robot sets into action and performs the physical manipulation by picking up an object, moving it to the promised position and orientation, and placing it again, all without exposing itself to the user. At the same time, mediated by their video-passthrough MR headset, the user perceives a branched version of physical reality in which it is not a robot, but a *virtual* cause of motion that moves the object to its promised pose. Fig. 2 shows how a virtual bee in a promised reality drops a virtual object, tightly synchronized with the drop by the invisible mobile robot. Seconds later, the promise will be fulfilled, and the virtual object can be recoupled into its physical counterpart again, thereby re-entering full physical reality. Based on the REALITY PROMISE illusion, we demonstrate four distinct end-to-end user experiences.

In a *MagicMake* experience, a user seeing their physical desk and, for example, working in an MR office setup can pinch their thumb and forefinger to spawn a *MagicMake* menu out of thin air. They can flick through the menu until they find an item of their liking, for example, a bottle of coconut water. They can then pull the object out of the menu, place it onto the desk, and then either continue to work on their virtual office panel or start observing how the bottle materializes in front of their eyes until an auditory notification confirms that they can grab the coconut water and drink from it. In a *Seamless MagicMove* experience (shown in Fig. 1), users can reach out for physical objects in the distance and, seemingly telekinetically, instantly move them through space. A minute later, if they were to stand up and walk to the place where they dropped the object in the distance, they would find that it had, in fact, changed its place. In a *De- and Rematerializing MagicMove* experience, the user can take a tissue from a Kleenex box on the coffee table, then point to the box, and perform a gesture to move it back onto the shelf in the distance, then see how the box progressively dematerializes from the coffee table and re-materializes in synchrony on the distant shelf. In a *Character MagicMove* experience, the user can observe how a virtual bee suddenly flies into their view, snatches away the physical chips container the user just ate from to remind them of their diet, and moves it back to a distant, maybe healthier, place.

These experiences, enabled by our REALITY PROMISE system, are based on two underlying technical components. Our *Skynet* component provides fine-grained control for the mobile manipulator robot, tightly integrated with the virtual interaction system and synchronized in lockstep with the user experience. We built *Skynet* with cheap, yet reliable inside-out robot tracking that shares the coordinate system of the user's headset. *Skynet*'s full-scale navigation system enables navigating the shared room and reaching inverse kinematics (IK) targets that are distributed across different surfaces in the scene. *Skynet*'s event monitoring and prediction system provides high-resolution information about the robot's current and planned state over time, thereby informing various interaction-relevant components such as object animations, character animations, or user-facing status notifications.

To visually hide the robot from view, our *SplatiMate* component performs real-time, untethered, on-device robot-aware 3D Gaussian splatting to project a previously captured splat into the user's co-aligned passthrough view where needed. By leveraging *Skynet*'s real-time information about the robot's current pose and its joint configuration, *SplatiMate* obtains the robot's visual-structural control points in 3D space, and from this, chooses the required subset of splats from the overall splat geometry that can hide the robot. Our dedicated stereoscopic vertex and fragment shader pipeline renders the splats coherently onto passthrough, smoothly feathering into the physical scene outside of the robot's vicinity. *SplatiMate* also enables object animations such as the de- and rematerialization effect or pop-up effects as used in the *MagicMake* menu.

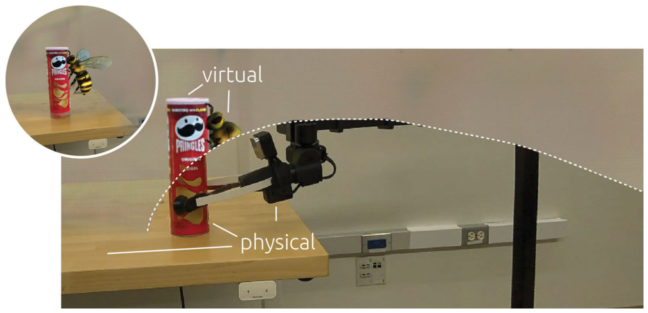

Figure 2: Virtual-physical decoupling and recoupling points are points in time *and* space where physical reality splits into a promised reality or merges with it again. At the shown recoupling point, the bee drops the promised object virtually and in synchronization with the robot dropping the physical counterpart.

Our *RealityGoGo* interaction technique allows users to communicate their manipulation intent to the system. It implements physical object selection through a point-and-pinch gesture as well as object placement through an input-amplified reach-and-drop gesture.

Contributions

In summary, we contribute

- the novel *concept and design space of REALITY PROMISES* for virtual-physical decoupling illusions which manipulate space and perception simultaneously in video-passthrough mixed reality,
- three distinct *MagicMove* and a *MagicMake end-to-end illusionary experience* enabling the motion or creation of matter in ways only virtuality affords while physically replicating the user- or character-evoked effects,
- the *interaction-synchronized mobile robot control engine Skynet* providing user-co-aligned mobile robot manipulation with inside-out tracking, navigation, inverse kinematics, event monitoring, and prediction in tight synchronization with the experience,
- the *on-device & standalone 3D Gaussian splatting system SplatiMate*, which introduces the concept of robot-aware multi-cone-casting for visually hiding the mobile robot and rendering object animation effects, enabled via a dedicated implementation in a stereoscopic and efficient splat vertex and fragment rasterization, shading, and animation pipeline, and
- the *MR-specific object selection and placement interaction technique RealityGoGo* enabling users to grab distant objects and drop them on distant surfaces through *surface-aware amplification*.

2 Related Work

In the following, we review previous research on virtual-physical divergence in virtual reality (VR) and in mixed reality (MR) and briefly consider research that used MR for high-level robot control.

2.1 Virtual-Physical Divergence in VR

Divergence in VR without Physical Control. The divergence between the user's perceived reality and the physical reality is at the core of VR. The display fully substitutes the visual signals emitted by the natural environment, controllers add otherwise physically non-existent haptics, and speakers add only virtually perceived sounds. Within this field, a dedicated branch of research [61] has explored deliberately manipulating the visual sense to indirectly also manipulate other senses such as proprioception [49], thereby aiming to induce an even stronger divergence between perceived and physical reality. *Redirected Touching* [36] and *Haptic Retargeting* [5] leveraged a warp field to offset the user's virtually perceived hand motion from its physical one, thereby aligning a passive prop with a virtual object. *Redirected Walking* [51] employed rotation scaling to systematically inject unnoticed virtual head rotations into physical ones, thereby extending the perceived area of the virtual environment. A perceptual manipulation, purely performed virtually, is found in *Mise-Unseen* [42] which exploited visual inattention to hide changes to the VR scene. *RealitySkins* [53] and *Substitutional Reality* [54] repurposed the physical scene layout for virtual interaction. However, the divergence between perceived and physical reality inducible by these approaches is limited, given that sensory disagreement increases with divergence, thereby eventually raising the user's awareness of the illusions at play.

Divergence in VR with Physical Control. Therefore, following the vision of McNeely's robotic shape displays [43], research has instead explored adapting the user's physical environment to the virtual experience. *Snake Charmer* [3] used a robotic arm to spatially align a physical proxy to a virtual object. *VRHapticDrones* [24] used a drone for force feedback, and *Beyond the Force* [1] leveraged a drone for object qualities beyond force feedback. *InflatableBots* [21] suggested inflatable tubes mounted onto an actuated base. *RoomShift* [57] presented a swarm of furniture-moving robots that enables a number of interaction designs from embodied to controller-based interaction. *HapticBots* [59] and *UltraBots* [16] used small table-top robots to emulate a large haptic surface via touch or ultrasound, respectively.

Conceptually, our work differs in that it is concerned with MR instead of VR. While works in VR usually aim to induce maximal divergence between the virtual experience and the physical reality, only converging the two when needed for the haptic illusion, conversely, Reality Promises aim to maximally preserve the user's experience of being in physical space, only diverging physical and virtual reality when needed for the illusion. Technically, Reality Promises therefore differ in their need to visually remove the robot, necessitating our contribution of robot-aware 3D Gaussian splatting for passthrough MR, not found in VR-related works.

2.2 Virtual-Physical Divergence in MR

Divergence in MR without Physical Control. Within the broader field of MR, *diminished reality* visually removes content from view [12, 22, 45, 46]. *SceneCtrl* [72] enables selecting, moving, deleting, and copying objects in the scene. *Remixed Reality* [39] shows a real-time reconstruction of the user's space in VR, captured through Kinect cameras and manipulable through an underlying voxel-based data structure. *Annexing Reality* [23] overlays physical objects with virtual representations. *TransforMR* [32] transforms a scene's semantics by visually replacing physical objects with virtual alternatives while matching the pose. *Scene Responsiveness* [33] enables the illusion of virtual control over physical space through forces or characters. *Asynchronous Reality* [17] enables users to remain in a physical space representation while only passing through physical space manipulations, e.g., objects delivered to the user's desk, in suitable moments by mediating the visual signal as required.

Reality Promises share the property of *Asynchronous Reality* and *Scene Responsiveness* to programmatically bring objects in and out of the user's experienced–seemingly fully physical–reality as needed. However, in contrast to all aforementioned works, Reality Promises also exert physical control over space by purposefully directing a mobile robot as needed, tightly synchronizing robotic actions with perceptual modifications.

Divergence in MR with Physical Control. Only a few works have considered inducing virtual-physical divergence in MR while actuating the scene simultaneously. Fabre et al. [14] use an AR headset to overlay an avatar over a chess-playing robot arm. Chen et al. [9] showcase the use of a co-aligned point cloud to "look through" a wall and control a robot in a neighboring room. *Reality Rifts* [10] in "analog reality" (not MR) aim to create divergence between perception and physicality by showing physical effects without physical causes through various hidden actuation mechanisms; however, this is not applicable to moving objects to user-selected drop positions. Sugimoto et al. [55] propose the use of diminished reality to partially "un-occlude" the task space of a robot occluded by the robot itself. Similarly, Plopski et al. [48] and its follow-up work in Taylor et al. [60] demonstrate the use of diminished reality to un-occlude a window occluded by a robot arm. Likewise, Cosco et al. [13] un-occlude the table on which the table-top robot is situated.

Similar to the last three described works, Reality Promises hide the robot from view. However, our work differs in multiple ways. *First*, we aim to enable *end-to-end illusionary user experiences*, where we do not only remove the robot but also aim for substitution of an object's cause of motion. *Second*, our three variants of *MagicMove* experiences therefore have control not only over the visibility of the robot but also the visibility of the objects of interest, toggling them in different places between physical, virtual, and hidden visibility across different phases of the interaction to obtain believable decoupling and recoupling illusions between the virtual and the physical world. *Third*, our *MagicMake* experience, enabling to seemingly materialize objects out of thin air, is another result of our experience focus, not realizable with techniques in previous works. *Fourth*, our need for tight interaction synchrony necessitates not only controlling and monitoring robot motions and events, but also predicting robotic actions minutes ahead of time which in turn informs various interaction-centric design components, such as object animations, character animations, or user-facing status predictions, *none of which* found in previous work. *Fifth*, interaction centricity also leads us to implementing an interaction technique that facilitates simple, yet seamless, situated grab and drop operations across arbitrary distances. Furthermore, because we aim to operate at room-scale rather than table-top scale found in previous works, our technical design differs fundamentally in further aspects, namely, *sixth*, on the robot side, we address this room-scale objective by implementing a robotic pipeline that controls a mobile robot with cheap, yet reliable inside-out, user-co-aligned robot tracking, a full-scale navigation system, and thus accounting for IK targets that can be distributed across different surfaces in the scene. And, on the visual side, *seventh*, we address this room-scale objective with our proposed robot-aware 3D Gaussian splatting pipeline instead of relying on techniques which are applicable only in smaller-scale scenarios such as image warping.

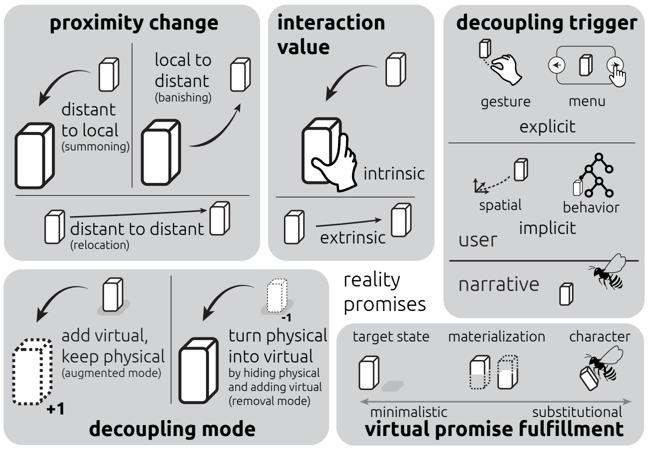
proximity change
local to distant (banishing)
distant to local (summoning)
distant to distant (relocation)
interaction value
intrinsic
extrinsic
decoupling trigger
gesture
menu
explicit
spatial
behavior
implicit
user
narrative
reality promises
add virtual, keep physical (augmented mode)
+1
turn physical into virtual by hiding physical and adding virtual (removal mode)
-1
decoupling mode
target state
materialization
character
minimalistic
substitutional
virtual promise fulfillment

Figure 3: Design space of Reality Promises. Reality Promises enable a range of object interactions that can be triggered and experienced in different ways. In this paper, we implemented all design parameters and design values except implicit user-elicited decoupling triggers.

2.3 Physical Control in MR without Divergence

All of the related work above aims to present a different reality than the physical one to the user's senses. In contrast, the intersectional field of AR plus robotics has evolved into its own comprehensive body of research that aims at augmenting robotic operation, rather than manipulating perception thereof. Suzuki et al. [58] and Walker et al. [63] present comprehensive frameworks for structuring the breadth of the field. Applications reach from AR for trajectory specification and monitoring [2, 19, 25], object-level manipulation [20, 37, 65], robotic behaviors [7, 27, 29, 64], debugging [28], sketching-based target state design [31], and remote collaboration [26] to teleoperation [15]. Our work differs in that it creates an illusionary divergence beyond augmentations.

3 Reality Promises for Virtual-Physical Decoupling

In the following, we introduce a design space of Reality Promises (see Fig. 3) and define our terminology and abstractions in virtual-physical decoupling illusions.

Physical Reality. A video-passthrough head-mounted display (HMD) enables users to continue visually perceiving their physical reality while interacting with virtual content. At the same time, they give control over every light ray that reaches the user's eyes and we leverage this to pass through visual information except in a carefully selected spatiotemporal subset of reality. All of our presented interactions begin by passing through unmodified physical reality to the user, potentially augmented by a non-immersive virtual application, such as a text editing app. Despite working on the app, the user remains aware of their physical reality and its objects.

Object-Body Interaction Value & Object-Body Proximity Change. Different objects offer interactions with different *interaction values* (see Fig. 3, top middle). Interactions with *intrinsic value* require bodily contact with the object, including consuming one-time use goods such as drinks or medication, or using reusable objects such

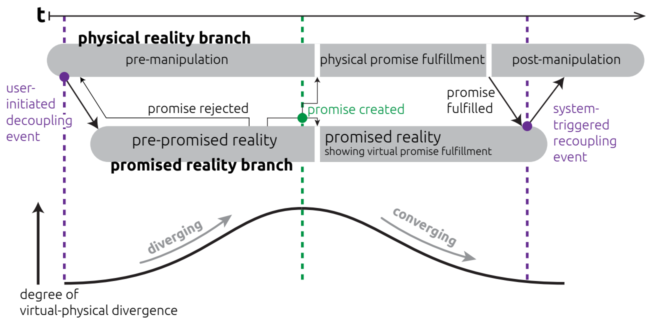

Figure 4: Timeline of a Reality Promise. By decoupling a virtual object from its physical counterpart, physical reality is branched into a *pre-promised reality.* As the user commits to a virtual manipulation, a Reality Promise is created and the user enters a *promised reality.* During *physical promise fulfillment,* the user is shown a substituting *virtual promise fulfillment.* Recoupling joins both branches, merging the virtual object seamlessly into its now again pose-equivalent physical object.

as books or glasses. In contrast, for many objects, the user is not interested in the tangible interaction with the object itself but instead the interaction's *outcome*, such as tidying up, moving decoration, or opening a window. These interactions have *extrinsic value* where the user's intent of the object interaction can be fulfilled outside of their proximity. Intrinsic interactions require body proximity (see Fig. 3, top left) and therefore, assuming a stationary user, entail the object's distant-to-local manipulation (summoning), the local interaction, and potentially a local-to-distant manipulation (banishing) to move it away again. Extrinsic interactions merely require a distant-to-distant manipulation (relocation).

Virtual-Physical Decoupling. As shown in Fig. 4, we want to enable the user to branch from the current state of physical reality into a subjectively different reality by *decoupling* a virtual replica object from its physical parent. In this decoupled branch of reality, the user applies *virtual manipulations* to the object, which are then *replicated physically* outside the user's perception. Virtual-physical decoupling begins with a *decoupling trigger* (see Fig. 3, top right).

We distinguish between user-elicited and narrative-elicited triggers. Narrative-elicited triggers might include a virtual character in a situated MR game that evokes the illusion of manipulating a physical object. Within the area of user-elicited triggers, we further distinguish between implicit and explicit triggers. Implicit triggers range from simple spatial triggers (such as taking a seat on a couch to trigger teleportation of a chips container) to more involved interpretations of user behavior (such as inferring thirst in a user based on their water intake). Explicit triggers include the use of gestures, virtual menus, or speech input.

As soon as decoupling is triggered, the object of interest "splits up" into a physical object and its virtual counterpart making them manipulable independently from each other. We distinguish between different *decoupling modes* (see Fig. 3, bottom left). In *removal mode*, we first hide the physical object from view with the object removal technique, also used to hide the robot. Then, a virtual object is added at the pose of the physical object. As the virtual object starts being manipulated, e.g., floating toward the user's hand, it appears to be leaving only an empty space, thereby strengthening the illusion that the user has control over physical reality. In *augmented mode*, a fully transparent virtual clone is overlaid on the physical object. It becomes opaque only when the user initiates manipulation, transitioning into view over the first few centimeters of the animation path to evoke the sense of extracting a "shadow version" while leaving the physical object visibly behind, thus reinforcing the perception of diverging into an envisioned reality.

Reality Promises & Promised Reality. In programming language design, the concept of asynchronously executed *promises* has been invented and defined as a placeholder for a computation result that will exist in the future [18, 40]. Inspired by this, we define a Reality Promise as a placeholder for a *physical manipulation result* that will exist in the future. As soon as the user commits to a virtual manipulation, the system creates a Reality Promise that will reconcile the induced discrepancy between perceived and physical reality. In our system implementation, a mobile robot navigates to the object of interest, acquires it, transports it to the target location, and places it as specified by the Reality Promise. We refer to this as *physical promise fulfillment.* Simultaneously, the system presents an alternative cause of motion to the user's eyes. We refer to this as *virtual promise fulfillment.* Both the user's perception and the system thereby enter an integrated promised reality as the promise is created, transitioning from a phase of virtual-physical *di*vergence to a phase of virtual-physical *con*vergence (see Fig. 4, bottom).

Pre-Promised Reality. Just picking up an object does not yet declare a desired target state, and therefore nothing can be promised yet. Only by dropping the object does a user commit to a virtual manipulation. Therefore, we distinguish between a *pre-promised* reality, which takes the role of a "reality workspace", and *promised* reality. This is analogous to promises in programming which require instantiating of a promise data structure, before their invocation. To prevent rejection of a non-fulfillable promise, we employ a *rectification* step after the user releases the virtual object, which moves it to the closest valid pose, uprighting it along the gravity axis and moving it onto the closest subjacent robot-reachable surface.

Physical Promise Fulfillment via an Invisible Mobile Robot. While physical manipulation could also be performed by humans or actuated objects, our system implementation automatically controls a versatile mobile robot. We visually remove the mobile robot through a graphics pipeline, thereby obtaining an *invisible mobile robot.*

Virtual Promise Fulfillment via Equifinal Causality Substitution. A Reality Promise specifies the object's source and target pose. While physical reality is being manipulated, we simultaneously fulfill the promise virtually (see Fig. 3, bottom right). Visually removing the robot and the physical object (in removal decoupling mode) enables the substitution of the physical cause of motion for a virtual cause of motion. All conceivable causes of motion must start at the current physical state and lead to the same finally observable physical outcome, defined by the Reality Promise, thereby changing the *mode of the manipulation*, but not the *mission of the manipulation.* Thus, we refer to the virtual promise fulfillment as an *equifinal causality substitution* (see Fig. 5). The virtually presented cause of motion, be it materialization or a monster, can be temporally synchronized with the robot's physical fulfillment, making

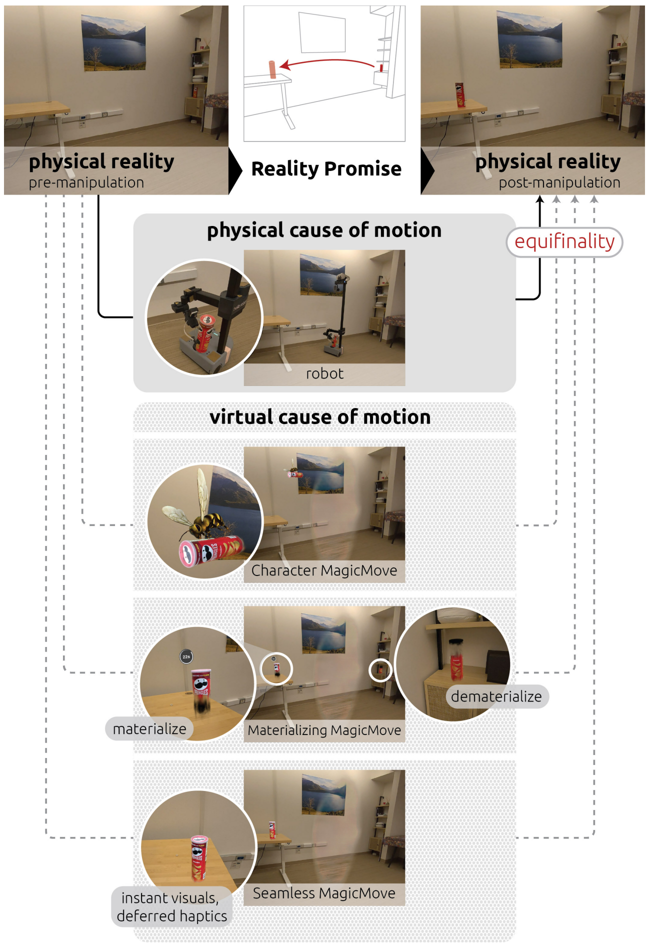

Figure 5: Equifinal causality substitution in *MagicMove.* A reality promise transitions physical reality from one physically valid state into the next, while showing any virtually conceivable modes of transitioning in between by replacing the physical cause of motion for a virtual cause of motion.

sure that both the virtual and the physical object are delivered not only to the same point in space *but also* at the same point in time.

The virtual promise fulfillment phase itself is subdivided into two sub-phases: Before the robot has acquired the physical object of interest, the object has to be visually removed or augmented. After the robot has acquired it, we can reveal the physically empty space.

Virtual-Physical Recoupling. Once the promise has been fulfilled both physically and virtually, it can be resolved by recoupling the virtual object into its physical counterpart. In practice, we smoothly fade out the virtual object over 5 seconds, thereby cross-dissolving the promised reality into the passthrough view, now showing the physical object at the same pose.

4 End-to-End Experiences

In this section, we develop two distinct types of end-to-end user experiences, *MagicMove* and *MagicMake*, informed by our afore-described design space and implemented in our system.

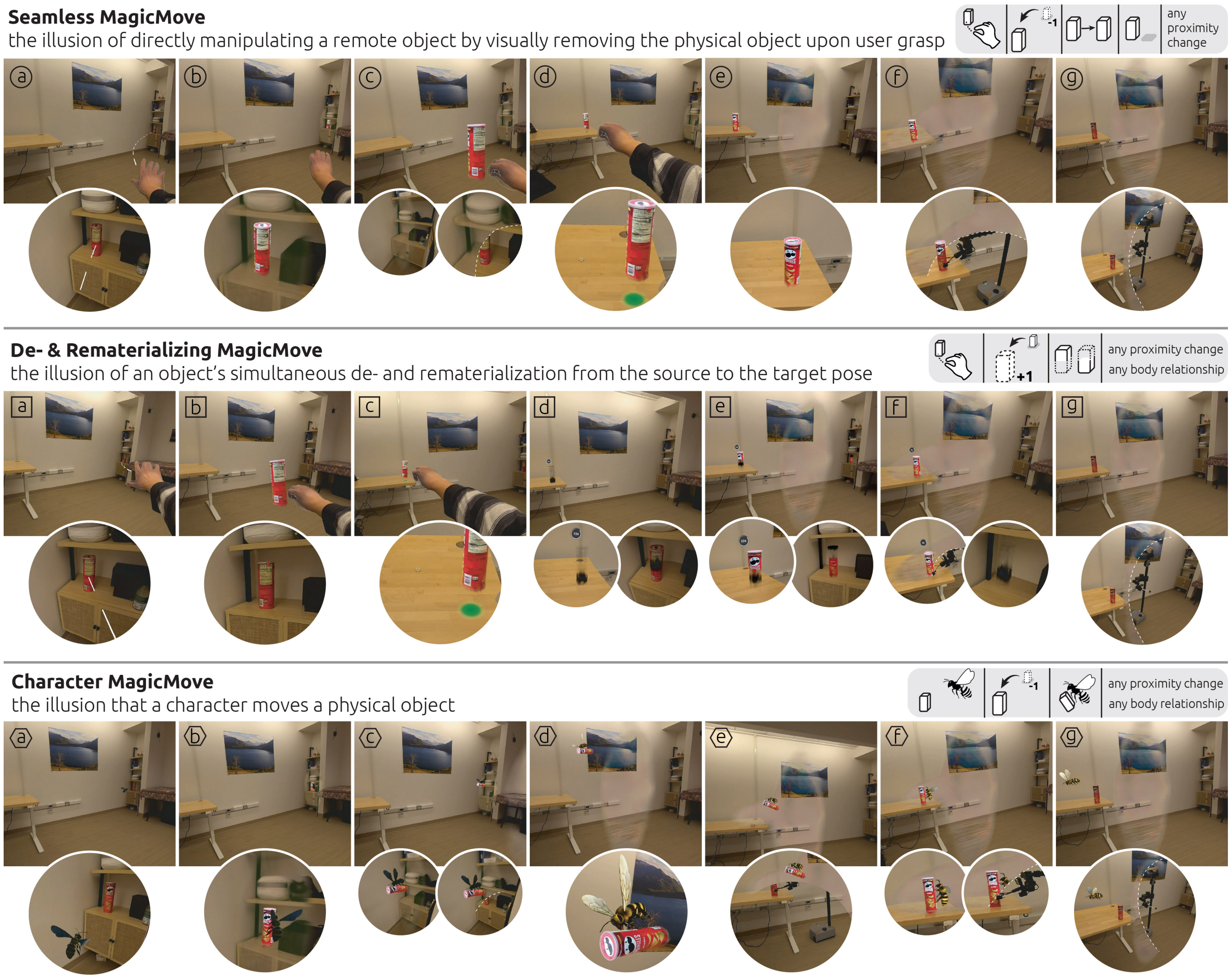
Seamless MagicMove
the illusion of directly manipulating a remote object by visually removing the physical object upon user grasp
-1
any proximity change
De- & Rematerializing MagicMove
the illusion of an object's simultaneous de- and rematerialization from the source to the target pose
+1
any proximity change
any body relationship
Character MagicMove
the illusion that a character moves a physical object
-1
any proximity change
any body relationship

Figure 6: Interaction breakdown of the three variants of *MagicMove* experiences. For a clearer illustration, this figure shows a comparative view with the same source and target poses across all experiences. For each experience, we show the user experience view (square images) and a close-up view (circle images) as well as the experience-specific design space choice (rounded gray boxes). All views were captured on the HMD and are unedited. Circle images revealing a hidden robot or hidden object were captured on device with the reveal lens (dashed virtual-physical separation lines added post-hoc). Screengrabs were taken across multiple runs to collect different views.

4.1 *MagicMove*

In the following, we present our three *MagicMove* variants, an overview of which is shown in Fig. 5, with a detailed breakdown of interactions presented in Fig. 6. We also relate each experience to the design space parameters (marked in *italics*) and outline their interaction protocols (where labels refer to Fig. 6).

4.1.1 Seamless MagicMove.

Design Rationale. We design *Seamless MagicMove* to evoke the illusion that users can telekinetically move objects by gestures. Therefore, we choose a point-and-pinch gesture as a *decoupling trigger*, hiding the physical object as soon as the user starts interacting with the object of interest via removal *decoupling mode.* This pretends the virtual gesture is the cause of object motion and therefore we show no other *virtual promise fulfillment. Seamless MagicMove* is designed to evoke the feeling that the user already "lives" in their promised reality. Once completed, the user's senses never experienced sensory disagreement about the physical whereabouts of the object. Because this experience provides no inherent visual cues of whether a virtually manipulated object is already tangible or not, it suits distant-to-distant manipulations that do not enter *body proximity* and have extrinsic *interaction value.*

Interaction Protocol. ⓐ The user targets the physical object. Using standard selection cone-casting, the most likely object among all selection targets is detected and indicated with a dashed line. ⓑ A thumb-index finger pinch instantaneously hides the physical object visually and inserts the virtual counterpart at its place. ⓒ At the same time, the virtual object starts floating to the user's hand, seemingly leaving only an empty space in the shelf (ⓒ, left circle). The reveal lens (ⓒ, right circle) shows that the physical object is just hidden behind a small visual patch that locally occludes the passthrough view. ⓓ The user uses our surface-aware amplification technique to reach out to the distant desk in order to drop the pre-promised object. A small green disk on the surface (ⓓ, circle) confirms the validity of the drop position. As soon as the user releases their pinch, the object smoothly moves and rotates onto the closest subjacent surface, thereby ensuring robotic reachability and correspondence with the robot's grasping affordance. ⓔ The invisible mobile robot has navigated toward the physical object, grasped it, and, as shown in ⓔ, is on its way to deliver it. As soon as the robot picked up the physical object from the shelf, the occluder patch is disabled, thus revealing the now physically empty shelf. ⓕ The robot is positioning the physical object at the exact same pose as the virtual promise object. The reveal lens (ⓕ, circle) shows the robot at its full extension, a split second after placing the object. ⓖ Invisibly, the robot drives to its home position, while the physical object is now at the promised pose.

4.1.2 De- and Rematerializing MagicMove.

Design Rationale. We design *De- and Rematerializing MagicMove* for summoning objects that the user intends to touch and use. Therefore, we again employ a point-and-pinch-gesture as a *decoupling trigger*, however, this time designed for distant-to-local manipulations that enter *body proximity* for intrinsic *interaction value*. Consequently, we use a de- and rematerialization animation for *virtual promise fulfillment* that conveys the promise is ongoing and how much of it remains. To deepen the materialization metaphor, we show a preparatory scanning-in effect at the physical object pose and scanning-out effect at the virtually promised object pose, only starting the actual de- and rematerialization effect once the robot has acquired the physical object. This allows us to use augmented *decoupling mode*. The scanning-in effect ends as soon as the robot acquires the object, switching over to the de-materialization animation at the now physically empty space.

Interaction Protocol. $\boxed{a}$ We employ the same gesture-based selection technique as before. $\boxed{b}$ Again, as the user selects the object by pinching, a virtual object is inserted at the place of the physical object; however, fully transparent so that the user does not see it yet. Only as the virtual object starts animating toward the user's hand, it gradually becomes opaque across the first few centimeters of its animation path, thereby creating the impression of pulling out a "shadow version" from the physical object. $\boxed{c}$ Again, using our surface-aware amplification technique, the user reaches out to a valid surface and releases it above, causing the object to rectify onto the subjacent robot-reachable desktop. $\boxed{d}$ We then set the virtual object's materialization progress to 0% and, additionally, expand a small duration indicator next to the object, showing the estimated time of arrival (ETA), in this case, the time to full materialization of 72 seconds as depicted ($\boxed{d}$, left circle). Simultaneously, we add a fully dematerialized virtual copy at the pose of the source object, setting its de-materialization progress to 0% ($\boxed{d}$, right circle). Then, both objects start to animate an "empty" scanning effect using a slicer sliding up and down along the object until the robot acquires it. The robot immediately sets into motion, and therefore the ETA countdown begins decreasing accordingly. $\boxed{e}$ As soon as the robot has picked up the object, we switch to the completion animation where the virtual promise object and virtual object at the source pose are synchronized with the progress between pick-up and placement. At the state depicted ($\boxed{e}$, left circle), the robot has covered approximately 40% of its robotic sequence since pick-up, and thus the colored part of the materialization covers approximately 40% of the object's height, while the ETA prediction shows 22 seconds. The slicer continues oscillating between the unfinished volume of the object until completion. The de-materialization of the virtual object at the source pose ($\boxed{e}$, right circle) is informed by the same progress information, thereby showing the inverse volume of the object. $\boxed{f}$ The robot delivers the object to the promised pose. Tightly synchronized, the promised object has fully materialized and the ETA prediction has decreased to 0 seconds ($\boxed{f}$, left circle). The virtual object at the source pose has fully dematerialized. $\boxed{g}$ Within 5 seconds of stowing its arm and while invisibly navigating back home, virtually presented and physical reality can be recoupled by fading out the now fully materialized virtual promise object into the physically delivered object. The virtual object at the source pose has also vanished, leaving only the empty shelf behind, making the illusion of de- and rematerialization perfect. The physical object is now at the promised pose.

4.1.3 Character MagicMove.

Design Rationale. We design *Character MagicMove* to give virtual characters such as virtual pets, creatures, virtually embodied assistants, or a bee in our implementation, control over physical space. The character's interaction serves as a *decoupling trigger* and immediately invokes removal *decoupling mode*. Because the bee can withhold the promised object during the pending promise, it inherently prevents the user from trying to interact too early with the object, therefore suiting objects with intrinsic and extrinsic *interaction value*. By freely choosing its path, it covers any *proximity change*.

As soon as the experience starts (in our implementation, a controller button press), the bee flies toward the virtual object, grasps it, and brings it to a promised pose. In contrast to the previous *MagicMove* interactions, in this experience, the user does not necessarily know where that promised position is. Instead, only the bee (programmed by our system implementation) knows the promised pose, and therefore surprises the user by the actual drop location. We amplify this surprise by not finding the shortest path between source and target pose, but a seemingly random path, changing course a few times mid-way.

This *faked randomness* draws inspiration from the concept of *forced choice* used by magicians that pull out a seemingly random card from the card deck to then reveal it is the predicted joker card. Apart from our aim to increase the sense of magic, more practically, it also stretches the path the bee takes to as long as is needed for the robot to physically perform its manipulations. Of course, like the magician who carefully engineered his hand motions and made his choice already, under the hood, our system fully specifies the promise as soon as the bee acquires the object.

Interaction Protocol. ⓐ The bee flies toward the physical object. ⓑ As soon as it reaches it, the physical object is visually removed from view and its virtual counterpart is rendered at its place. ⓒ The bee leaves with the attached object, leaving behind a seemingly empty shelf (ⓒ, left circle). The reveal lens (ⓒ, right circle) shows that the physical object is still in its physical place, just hidden from the user's view. ⓓ The bee flies in a non-straight path through the room. In the figure, we show a screengrab toward the end of the path. ⓔ While the bee flew in a more or less constant speed up to this, only varied slightly for more organic motion, at an ETA of 2 seconds, the bee starts moving across the shortest path to the promised pose, thereby ensuring exact spatiotemporal coincidence between the physical object drop from the robot and the virtual object drop from the bee. ⓕ In exact temporal synchronicity, the bee and the robot reach the promised pose. ⓖ The robot drives back to its home pose and the bee flies out of frame, disappearing outside the user's view frustum. The object materialized.

4.2 *MagicMake*

MagicMove interactions enable 1-to-1 object motions from a user-selectable source pose to a promised pose. In contrast, our *MagicMake* interaction enables a 0-to-1 object manipulation where even the source object's pose is abstracted away behind a menu. *MagicMake* thereby provides the illusion of magically making matter out of thin air.

Design Rationale. We design *MagicMake* with a menu as a *decoupling trigger* that can be expanded anywhere in space, through a thumb-forefinger pinch. From the menu, the user can choose objects that are out of sight, potentially even beyond the room, and therefore out of scope for *MagicMove*. Objects obtained through the menu will be manipulated from distant-to-local, entering *body proximity*, and therefore likely also being of intrinsic *interaction value*. As a result, we make use of our materialization animation to communicate the status of promise fulfillment. Because the user pulls the promised object out of a virtual menu, there is no visible physical-to-virtual decoupling transition, however, if the user were to walk to the source object's pose, they would find the physical object (if not acquired yet) in augmented *decoupling mode*.

Interaction Protocol. As shown in Fig. 7, ⓐ the user starts by holding up their hand into empty space. ⓑ By pinching their fingers, they spawn the *MagicMake* menu. The menu expands outward from its round icon in the infinitesimal point of pinch to its full size, oriented toward the HMD. ⓒ The user is not interested in the Coke Zero and therefore flicks through the menu… ⓓ …until finding their object of choice. ⓔ They reach into the menu's depicted object and "pull it out" from the 2D panel into 3D space. Here, we draw inspiration from Tony Stark's holographic JARVIS interface to discover a new element in Iron Man 2 (2010), which allows fluid 2D-to-3D object manipulation, and from the suitcase interaction in the VR version of Resident Evil 4 (2021). ⓕ As soon as they drop the object, it rectifies onto the subjacent surface and starts materializing. The menu icon detaches and smoothly lerps and slerps to a pose next to the object, facing the HMD, while the menu panel contracts into an infinitesimal point again. ⓖ The promised object continues materializing… ⓗ …until materialization is fully completed, the ETA

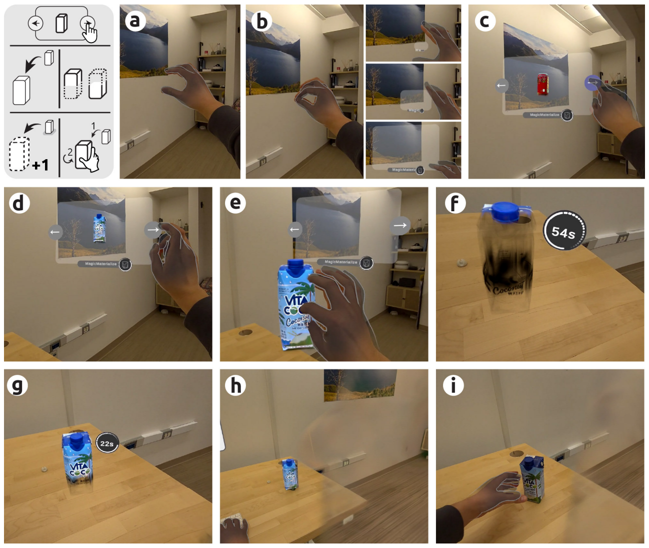

Figure 7: The *MagicMake* menu enables materializing an object out of thin air.

prediction becomes zero, and in sync, the invisible mobile robot reaches out to physically fulfill the promise. ⓘ As the robot retracts its arm with the intention of driving back to its home position, the virtually promised object recouples into the physical object through a fade-in. The user may reach out and have their drink, which the robot delivered from the snack table in the opposite room side.

5 Architecture and Implementation

In this section, we outline our system architecture and implementation with its *Skynet* robotic component, *SplatiMate* Gaussian splatting component, and *RealityGoGo* interaction component.

5.1 *Skynet*: Interaction-Synchronized Mobile Robot Control

5.1.1 Architecture Decisions.

User Headset. We implemented our user application in Unity 2021.3 on macOS 14.7 and deployed it as a standalone application to a commercial Meta Quest 3 device, featuring 2x 4 MP passthrough cameras, *without* Quest Link or server offloading. It therefore runs fully untethered, offering the user full mobility range. We disable Quest's *Physical Features* (incl. *Boundary*, formerly *Guardian*), only re-enabling them when setting the floor height. The Quest Passthrough Camera API is not needed.

Robot. For the robot, we chose Hello Robot Stretch 3, featuring a 2+1 wheel differential drive in the mobile base, vertical lift with a height adjustable to 120 cm (plus base) via a prismatic joint, a telescoping arm horizontally extensible to 76 cm, and a 3 degrees of freedom (DoF) gripper spreading its fingers to 15 cm left-to-right and carrying up to 2.5 kg max payload. Inside the base, the robot carries an onboard Intel NUC 12 Mini PC on which we run the manufacturer's firmware driver, ROS2, Nav2, and a rosbridge-suite server with several custom ROS2 nodes for information flow management. The base LiDAR and RGBD cameras are not needed since we replace them as follows.

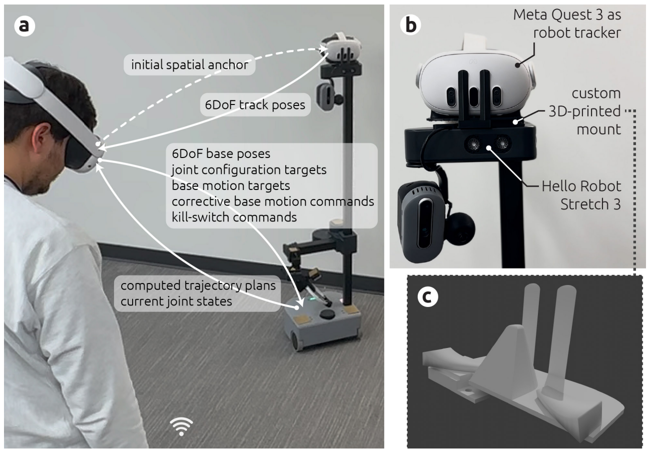

Figure 8: *Skynet*'s system architecture comprises the user headset, the robot-tracking headset, and the robot. ⓐ The user headset communicates with the robot and the robot-tracking device. ⓑ The robot-tracking device is another *Quest 3* headset for inside-out tracking. ⓒ It is attached to a custom 3D-printed mount.

Robot-Tracking Device. User- or character-defined drop poses specify the Reality Promise, which has to be physically fulfilled by the robot. This alone demonstrates and necessitates a shared coordinate system between the user and the robot to convert a user-relative pose into a robot-relative pose, which can then be used for navigation and inverse kinematics (IK).

As one of the most advanced inside-out pose tracking systems available, we chose to leverage *another* Meta Quest 3 headset purely for its visual-inertial SLAM capabilities. Therefore, we modeled and 3D-printed a mount (see Fig. 8c) which we screwed onto the robot's head, providing tight grip after sliding in the robot-tracking headset from overhead (see Fig. 8b). To prevent unintended sleeping, we set the sleep time-out to 1h and taped off the proximity sensor.

Based on an ICP algorithm that aligns the point clouds of each device, we leverage Meta's shared spatial anchor infrastructure to obtain an initial transform between both headsets upon app startup. Using this feature requires a Meta-assigned Quest app ID as point cloud data is stored and computations are performed remotely on Meta infrastructure for privacy assurance, and therefore, we deployed our app once via the Quest developer pipeline for alpha testing through the Quest app store. From then on, we reinstalled the signed version via wire for faster iterations. Deploying the same Unity application as we deployed to the user headset, we implement and switch to *pose tracking mode* based on statically configured device IP addresses, which streams its tracked 6 DoF headset poses at 4 Hz via WiFi to the user headset. Based on this overall architecture, we implemented a comprehensive pose transform pipeline for navigational and joint kinematics control, which will be outlined in detail in the next sections.

Communication. We connect all devices to the same local WiFi network for communication. In summary (see Fig. 8a), the user headset initiates a shared-space session with the robot-tracking headset and from then on receives robot headset poses, transformable to user space, continuously. These poses, as well as all other motion commands, are sent to the robot base computer to update the navigation stack transform hierarchy and send joint trajectories or fine-grained non-navigational motion commands. At the same time, the most recently computed motion plan and the most recently measured joint states are sent from the robot to the headset. Taken together, the user's headset has full information and control about the robot's current and planned state and actions.

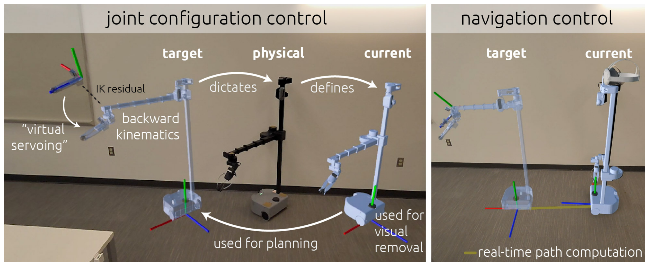

Figure 9: *Skynet*'s information flow for high-level robot control. *Left*: *Skynet* maintains a target robot state representation and a current robot state representation directly on the headset, enabling full IK. *Right*: Based on a virtual object's affordance, *Skynet* plans, executes, and controls for a suitable rotation-aware path under closed-loop feedback from the robot-tracking device.

5.1.2 Robot Joint Control.

Local IK: Custom Problem Setup. Assume the robot is standing on the floor next to a table, which holds a physically promised object, with the arm pointing toward it. Given an object's grasp affordance (i.e., the relative pose between gripper and object at the moment of grasping), the *local inverse kinematics* problem asks for the five robot joint parameters: vertical lift, horizontal arm extension, and 3 DoF gripper pose. Given the Cartesian design of the Stretch 3, we can analytically obtain the 2 positional (lift, extension) and 3 rotational (yaw, pitch, roll) joint parameters through trigonometric and vector-algebraic computations, without the need for numeric IK optimization. We compute the yaw in case the affordance is not perfectly aligned with the plane spanned by the arm and mast to account for navigational imprecision.

Local IK: Headset-Based Solution. We perform these computations directly in our Unity user app. Furthermore, we visualize the result of these computations by exporting the robot's URDF model from manufacturer-provided data, significantly simplifying its mesh geometry for performance in Blender, simplifying the kinematic chain hierarchy, and importing it into our user app. As can be seen in the virtual debug mode in Fig. 9, left, given a desired grasping end-effector state (visualized in the HMD by a gripper with co-located coordinate frame axes in the left of the screengrab), we can thereby accurately compute the state of the robot's target state in the HMD (left robot representation in the figure). Even if the desired end-effector state is out of reach, the best possible state will be computed, leaving only the irreducible IK residual.

Real-Time Implementation: Virtuality-Guided Servoing. Given this target state, we send the computed joint parameters to the physical robot, more specifically to the ROS trajectory action server. While the physical robot performs these actions, in real-time, we obtain the live state of the joints from the robot (see Fig. 9, left, right robot representation). As we will detail later, we make use of this current

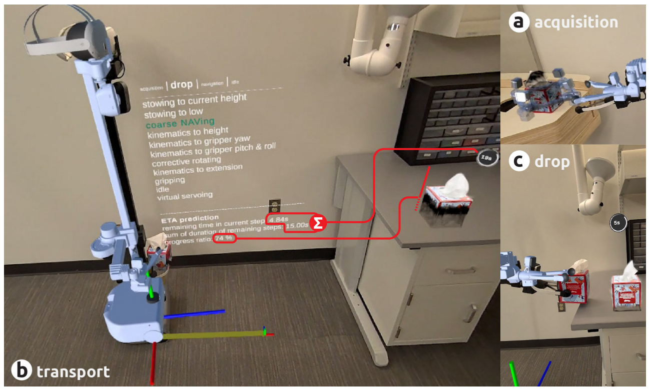
a acquisition
c drop
b transport
acquisition | drop | navigation | idle
stowing to current height
stowing to low
coarse NAVing
kinematics to height
kinematics to gripper yaw
kinematics to gripper pitch & roll
corrective rotating
kinematics to extension
gripping
idle
virtual servoing
ETA prediction
remaining time in current step: 4.84s
sum of duration of remaining steps: 15.00s
progress ratio: 74 %
Σ

Figure 10: *Skynet*'s state machine for a pick-and-place sequence in debug mode. ⓐ The best object-annotated affordance serves as an IK goal. ⓑ The robot is on its way to drop the previously acquired Kleenex box (see gripper). The state machine, visualized in debug mode as a text box next to the vertical mast, is maintained directly on the user headset. The red lines, added post-hoc, show how the state machine's ETA prediction flows directly into the materialization animation (top line: remaining in current step + sum of duration of remaining steps, flowing into the ETA prediction indicator; bottom line: progress in percent). ⓒ The robot delivers the object in sync with the materialization effect.

state representation to compute the accurate occlusion geometry that will visually hide the robot for the Reality Promise illusion.

5.1.3 Robot Navigation Control: From Hand Drop Gesture to Wheel Actuation. In our aforementioned local IK approach, the mobile base pose is presumed fixed. To also move the robot across the 2D map, we expand the above to a global IK approach by first computing an optimal base pose for a given end-effector goal, and second, navigating toward the computed pose, building on our initially proposed robot-tracking setup.

Computing an optimal base pose given an end-effector goal. A Reality Promise specifies a promised object's source pose and promised pose. By annotating the grasping affordance for each virtual object representation (see Fig. 10a) in Unity at design time, and by then aligning these objects with the physical scene after app start-up, the user app is aware of the end-effector goals as a user picks up an object at the annotated poses, or drops an object onto an arbitrary surface, maintaining coordinates in *user-tracking space*. Making simplifying assumptions to resolve base pose underdetermination (arm extension always at 40 cm, gripper yaw always at 0°), we can compute the remaining 3x 1 DoF joint parameters via trigonometry and vector algebra and convert them into joint transform matrices. Finally, we traverse the kinematic chain backwards, obtaining the *base pose in user-tracking space*.

Navigating toward the computed pose. Given a *base pose in user-tracking space*, we need to actuate the two drive wheels. To this end, we first convert the base pose in user-tracking space to a *base-pose in robot-tracking space* using the shared spatial anchor transform between user and robot headsets, which we obtained on app start-up. The robot's root joint, i.e., the mobile base, itself is also tracked in *robot-tracking space* and continuously updated by

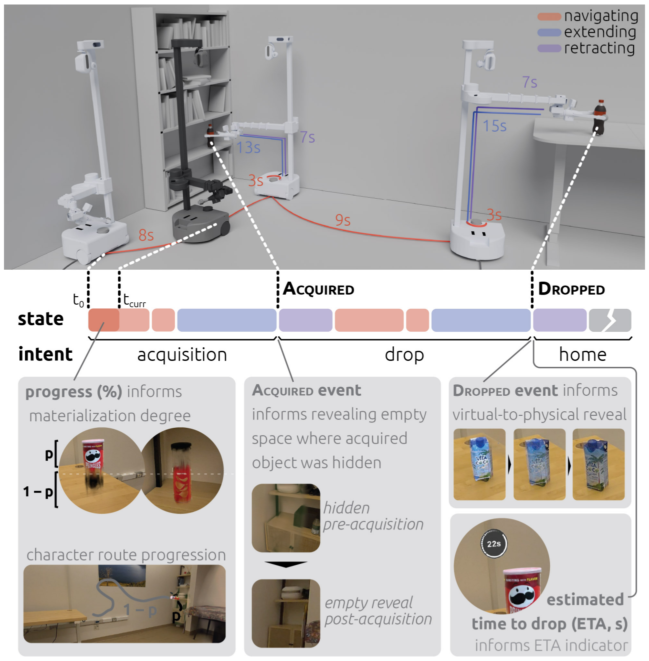
navigating
extending
retracting
7s
15s
13s
7s
3s
3s
8s
9s
t_0
t_{curr}
Acquired
Dropped
state
intent
acquisition
drop
home
progress (%) informs materialization degree
p
1 – p
character route progression
1 – p
p
Acquired event informs revealing empty space where acquired object was hidden
hidden pre-acquisition
empty reveal post-acquisition
Dropped event informs virtual-to-physical reveal
22s
estimated time to drop (ETA, s) informs ETA indicator

Figure 11: *Skynet*'s event prediction & monitoring system is tightly coupled with the rendering and virtual interaction system.

the robot-tracking headset. However, we aim to perform the motion planning as well as the low-level navigation control on the robot base computer in ROS2 Nav2. Therefore, we convert the continuously forwarded robot positional vector and rotational quaternion, and the computed optimal base pose from Unity's coordinate convention (x right, y up, z forward) to Nav2's convention (x forward, y left, z up), before sending this computed pose representation to the robot computer. We fuse the continuously streamed robot pose with the wheel+IMU odometry transform in a custom ROS2 `tf` hierarchy broadcast node, however, resetting the relative transform with each headset-tracked pose update, thereby enabling dead reckoning. Navigational targets are sent to ROS2's `NavigateToPose` action server. As shown in Fig. 9, right, for debugging purposes, at 1 Hz, we obtain the most recent navigation trajectory, apply the inverse convention and transform sequence, and display the trajectory in the user headset.

5.1.4 Promise-and-Robot Integration.

End-to-End State Machine. The aforedescribed navigational and joint kinematics computations provide all building blocks for a full home-to-acquisition-to-drop-to-home sequence. We design a comprehensive finite state machine, maintained in the user app and shown in Fig. 10b. The current idle pose is considered the home pose. As soon as a Reality Promise is defined with its promised object's acquisition and drop pose, the robot transitions into a navigational state (`coarse NAVing`) with *acquisition* intent, performs a fine-grained corrective rotation at the reached base pose target, extends to its full grasp position through a sequence of individual joint actuation states, and repeats the same procedure with a gripped object, now under *drop* intent, until finally dropping the object (see Fig. 10c), switching to *idle* intent and driving back to its

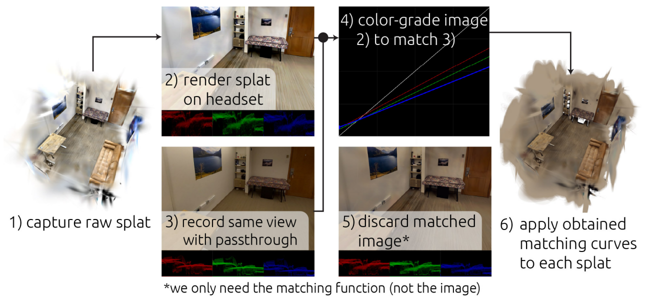
1) capture raw splat
2) render splat on headset
3) record same view with passthrough
4) color-grade image 2) to match 3)
5) discard matched image*
6) apply obtained matching curves to each splat
*we only need the matching function (not the image)

Figure 12: *SplatiMate*'s photometric alignment between the captured splat scene and the passthrough view accounts for color differences in the images from the iPhone capturing camera and the Quest passthrough cameras.

home pose to additionally set and remain in the *idle* state. We add intents as an orthogonal stateful variable, which allows the state machine to distinguish, e.g., between navigating to acquisition vs. drop pose, while reusing the same states and transitions.

Event Monitoring and Prediction. Reality Promises not only require end-to-end execution of the pick-and-place sequence, but also an event system that is tightly integrated with the robot operation and predicts and triggers actions, based on expected and actual acquisition and drop events. As detailed in Fig. 10b, we can compute the ETA to the acquisition or drop event through summation of the estimated duration remaining in the current step plus the estimated duration of all remaining steps. For nearly time-invariant steps, we collected a look-up table providing reference durations, and for navigation, we assume a velocity of 0.3 m/s across the navigation plus a constant of 5 seconds to account for rotations in the path segment's start or end pose. As shown in Fig. 11, based on the ETA and the elapsed duration, we can inform the ETA indicator and compute the overall progress, which informs the materialization degree in the respective *MagicMove* and *MagicMake* experiences. For *Character MagicMove*, this progress represents the route progression degree of the bee character along its path of faked randomness. As soon as a state machine transition fires an *acquired* or *dropped* event, the respective virtual objects at the source and target pose react with their interaction-specific visual recoupling events.

5.2 *SplatiMate*: Robot-Aware 3D Gaussian Splat Rasterization, Shading, and Animation for On-Device Mixed Reality

To hide the robot and physical objects from view, we implement *SplatiMate*, an on-device 3D Gaussian splatting system for MR. *SplatiMate* renders the splat from the perspective of each eye, however, only in the pixel neighborhood where the entity to be hidden is seen by the user. Furthermore, we design *SplatiMate* to render interactable objects, also implementing a custom shader for animating de- and re-materialization effects. *SplatiMate* can render multiple objects jointly, compositing them properly with each other, and alpha-blending the result correctly into passthrough, thereby obtaining a view in which the robot and physical objects are concealed and virtual objects are animated. In the following, we outline the process from splat preparation to splat rendering, and point to the various adaptations from standard Gaussian splatting [35] for our purpose.

5.2.1 *Splat Preparation.*

Step 1: Object & Space Reconstruction. We first captured the space (ca. 200 images) and objects (ca. 15s videos) using an iPhone 14 Pro. We train the splat in Jawset Postshot (v0.5.146), using the MCMC profile, and cap the splat count at 25k splats for scenes and 3k-5k for objects. Training the splat scene takes ca. 3 min, preceded by a camera pose estimation that runs for ca. 10 min. Our implemented renderer is able to properly handle 100k splats without frame drops. However, we cap the splat count for scenes and objects as outlined above to leave sufficient computational budget on the GPU for both removal splats and multiple virtual object splats, animations, feathering in the vertex shader, screen recording, and Unity overhead. Fig. 12, step 1, shows the training result for the space reconstruction from a bird's eye view, rendered for illustration in SuperSplat before exporting to the headset. All interactable objects, shown in the figures of this paper, are also splats.

Step 2: Rectification. The resulting training artifacts lack a meaningful root origin and orientation, and due to depth ambiguity in monocular structure-from-motion, they lack meaningful scale. We measure the objects and space, using a laser range finder for the space, and then, in Blender, employ a Splatting add-on [8], to scale them accordingly, align them upright along the positive y axis, and position them on the xz plane front facing along the z axis. We also cut out excess geometry before exporting a rectified splat. In a custom Python preprocessing script, we convert both the position vector and the rotation quaternion of each splat instance from Postshot's coordinate system in OpenGL convention (x right, y up, z backward) to Unity's coordinate system. Furthermore, in Python, we compute the covariance matrix for each splat instance's scale and rotation ahead of time, thereby taking load off the GPU at runtime. We write out all splats in a CSV data format, more easily readable in Unity than splat PLYs.

Step 3: Color Grading for Space Splats. The difference between the iPhone camera's response function and the passthrough camera's response function leads to different photometric qualities: the rendered scene looks different than the passthrough view. We therefore aim to color-grade the splat such that it matches the passthrough appearance. To this end, we render the full splat from a view in the headset (see Fig. 12, step 2) and, capture the same view with the passthrough cameras (see Fig. 12, step 3). As can be seen in Fig. 12, step 3, the passthrough cameras use less range than the rendered splat (which mirrors the iPhone's higher dynamic range images). Using Adobe Premiere's Lumetri view, we load both images, use tone curves to match the RGB parade scopes and the overall appearance, and export the Premiere's *.look* file. Fig. 12, step 5, shows the matched image, which closely resembles the actual passthrough view in step 3. Discarding the image itself, we then apply the same tone curve to all individual splats in the scene in Python, and export a rectified and *color-matched* splat.

Step 4: End-Effector Annotation for Object Splats. In Unity, we annotate the end-effector position for object splats and add grab manipulability to object splats and the space splat, so that the scene can be aligned on app start-up. Overall, preparation across these four steps totals ca. 4h.

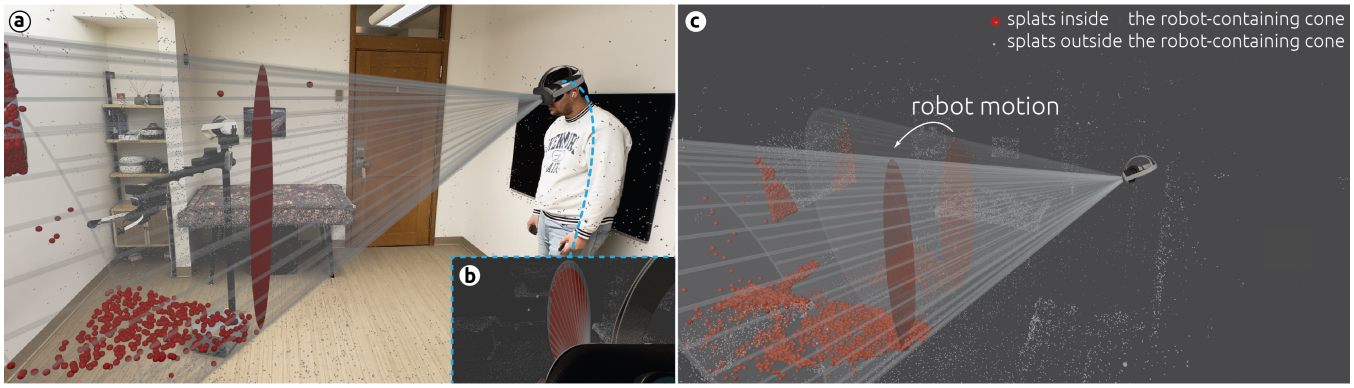
a
b
c
● splats inside the robot-containing cone
· splats outside the robot-containing cone
robot motion

Figure 13: *SplatiMate*'s robot-aware cone-casting finds all splats that must be rendered to cover the robot in view. ⓐ Side view showing how we cast a cone toward the robot from the left eye. ⓑ User-aligned view from slightly above the headset's left-eye display showing the cone along its central axis. ⓒ Bird's eye view showing how the cone automatically tracks the robot as it moves. For simplified illustration, we only show the vertical cone, covering the robot from base to head, omitting the horizontal cone covering the arm, and reduce splats to regular spheres (big red inside the cone, small gray outside) disregarding Gaussian covariance information and color.

5.2.2 *Splat Rendering.*

Standard Splat Rasterization. We first implemented a custom minimal reference implementation in an OpenGL shader (approx. 500 lines of code in python, of which 150 are GLSL shader lines), consulting existing third-party [34, 70] and original author [35] implementations for splatting and underlying techniques [73]. Then, we manually translated them line by line into Unity's high-level shader language (HLSL), in the process introducing changes for stereoscopic, asymmetric-eye-frustum, passthrough, convention-specific rendering, and differences in gamma correction handling. To account for the lack of geometry shaders and quad meshes on Quest, we use two triangles with 6 corners, 2 of which are shared, to represent the splat quad, thus constructing a "pseudo-mesh" with 4 times as many vertices as splats. We pass splat positions, colors, opacities, and 6D covariance parameters for each vertex into the shader via mesh UV channels and disable frustum culling. For alpha-blending, we outsource splat sorting based on camera distance from far to close into a background thread and hot-swap the resorted splat cloud into the UV channels at 1 Hz. We thereby obtained a standalone & on-device VR viewer for Gaussian splatting, enabling exploration of the captured room and captured objects in life-size.

Robot-Aware Splat Shading via Multi-Cone Casting. In this project, we are not interested in rendering the *full* space scene, but aim to only render the region within which the robot is currently situated. More formally, given the 3D splat geometry, the eye's view position and direction, and the robot's current position, rotation, and joint configuration, we want to render a 2D patch which we can seamlessly composite onto the 2D passthrough image, thereby concealing the robot.

Our idea, visualized in Fig. 13a, is to cast a cone from the user's left or right eye position with an oval cross-section, covering the robot's vertical axis, i.e., covering the robot base, mast, and head. Then, we only render splats that are inside this cone, discarding all other splats. By casting a second cone, encompassing the horizontal arm, and then combining both rendering results, we obtain a patch that fully covers the robot. Since we track the robot, we know its structural control points at all times. By recomputing the required cone geometry at every frame with current structural control points, the robot remains inside the cone, even as it moves (see Fig. 13c).

Fig. 14 provides an insight into the underlying trigonometry. From our robot tracker, we know the pose of the robot and thus its structural control points including its top T and bottom B points in 3D space. From the user's headset, we also know the eye pose E in 3D. Given the triangle $\triangle ETB$, we first compute lengths ET and EB. From the angle bisector theorem, we obtain the angle bisector ratio $r = \frac{ET}{EB}$. Using the section formula, we then compute the bisector's point of incidence as $P^* = \frac{T+rB}{1+r}$. To obtain the cone transform matrix, we compute the look-at rotation from E to P^* representing the cone axis, oriented upward from B to T, and translated to E as the cone apex. In the vertex shader, we transform splats by the cone transform matrix and discard or retain them based on their angle divergence from the central cone axis.

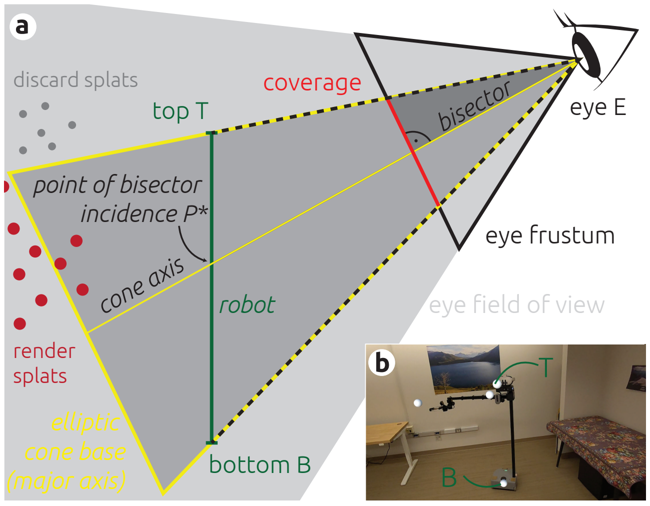

Figure 14: Structural control points of the robot inform our cone geometry computation that automatically shrinks or grows the coverage patch as the user or the robot moves, or as the robot extends or retracts.

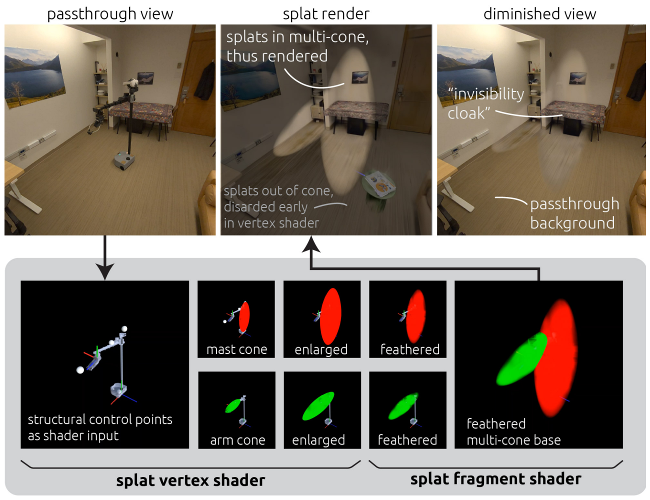

Figure 15: *SplatiMate*'s splat vertex and fragment rasterization & shading pipeline. All images rendered on-device in debug mode. Text labels added post-hoc. The grayed-out out-of-cone render in the top-mid image is shown for illustration, and discarded early in the vertex shader during the user experience only rendering the robot-covering patch.

Our shading pipeline, executed for each eye buffer, is shown in Fig. 15. The vertex shader discards out-of-cone splats already early on by sending them outside the clip space. In the fragment shader, we then use the cone geometry to determine a pixel's normalized squared elliptic radius on the conic section to determine a feathering degree. By feathering, the rendered patch smoothly transitions into the passthrough view. Additionally, in the fragment shader, we minimum-blend the two normalized squared elliptic radii, one for mast ellipse and one for the arm ellipse, to find a smooth handover of the feathering degree at the edge from one rendered ellipse to the other. This means we complement Gaussian splatting's classical alpha-blending along the depth axis with cone-guided alpha-blending of splats along the image axes. Overall, what results is a feathered multi-elliptic shape that covers the full robot as it moves, turns, extends, or retracts, or as the user builds distance or proximity, or as the user turns.

We render the result into an `ARGB32` render texture at a resolution of 1680w × 1760h and blit the result into the respective eye's `OnRenderImage` hook using a custom blitting material that accounts for gamma correction after alpha-blending together all fragments (premultiplied, `Blend One OneMinusSrcAlpha`). While we found that using an `ARGB64` render texture increases the rendering quality considerably, preventing tints or artifacts that are at times noticeable in the patch otherwise, the added performance overhead is infeasible for interactivity, and therefore we stick to `ARGB32`. The final result, composited with the passthrough view, can be seen in Fig. 15, top right.

5.2.3 Splat Animation. To animate the de- and rematerialization effect on virtual objects, we first compute the axis-aligned bounding box for the splat, and then separate it into two slices depending on the progress of promise fulfillment. We use true splat colors for the completed part, while setting the color of the individual splats of the uncompleted part to black and nearly transparent (10%). Furthermore, we slice the incomplete part again into an oscillating middle part, depending on elapsed time, and set it to zero transparency, thereby obtaining a scanner-like effect to represent that the promise is pending.

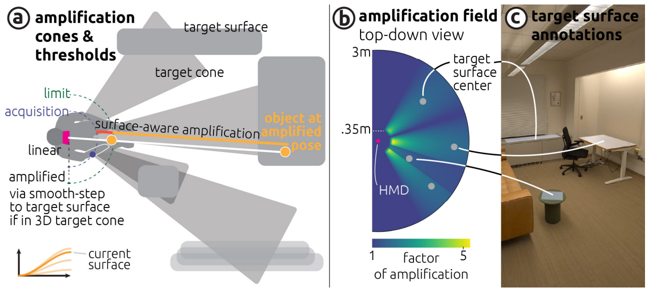

Figure 16: *RealityGoGo*'s surface-aware amplification. ⓐ Top-down view illustration of *RealityGoGo*'s *dynamic* amplification technique which maps a fixed travel distance to the dynamically computed distance to the best-matching drop surface. ⓑ Top-down view plot of the amplification field, visualizing the amplification factor at each possible hand position after pinching at a pinch distance of 30 cm. Amplification kicks in 5 cm farther out, highest in the direction of the farthest surface, over a distance of 15 cm. Surfaces indicated as gray dots. ⓒ HMD view of surface annotations of the sill, desk, and coffee table during space setup.

5.3 *RealityGoGo*: Surface-Aware Amplification for Grab-and-Drop Interactions in MR

Interaction Setup. As the app starts up in a new space, we manually co-align the virtual scene splat with the physical scene, interactable virtual objects with current poses of physical objects, and virtual surfaces with physical surfaces (similar to Meta's room setup feature). Fig. 16c shows an example of three aligned surfaces. These surfaces represent robot-reachable drop surfaces onto which the user may drop a selected object.

Interaction Technique. We aim to enable the user to select distant objects and drop them on a suitable drop surface in their environment. To this end, we implement an interaction technique we refer to as *RealityGoGo*, which introduces four differences from the classical Go-Go interaction technique [50] to adapt it from VR to MR with multiple physical-world target surfaces at different distances. 1) For grabbing, we animate the object toward the user's hand as described, instead of "casting out" a virtual arm or hand. 2) We define the amplification threshold dynamically based on an acquisition point. 3) For dropping, we do not amplify the arm or a virtual hand but simply the object position. This avoids the need to render a visually distracting virtual Go-Go arm onto the passthrough view. 4) We use a target-aware smooth-step function between the acquisition radius and a limit radius that quickly flattens amplification again, thereby eliminating the risk of overshooting, and in particular, allowing users to reach all surfaces across varying distances comfortably. Fig. 16a visualizes the underlying amplification concept.

Step 1: Grab. Shown in Fig. 1a, as the user points towards an object, we use the Quest Interaction SDK's distance hand grab interaction with an object motion provider that animates a virtual twin of the object toward the user's hand. However, we extend it for a drop interaction as follows.

Step 2: Interaction-Wise Calibration. As soon as the object arrives at the user's pinched fingers, we define the current pinch position as the calibrated amplification threshold, referring to this as the *acquisition point.* The distance between the headset and the acquisition point represents a radius around the user.

Step 3: Isometric and Amplified Control. If users pull the object closer to their head to inspect it, hand motion is isometrically mapped to object motion. However, as soon as they reach out again beyond the acquisition point radius (plus a small safety offset of 5 cm to avoid unintended amplification after acquisition), they operate in an amplified regime until they hit the limit radius. The limit radius is computed as the acquisition radius plus 20 cm. While amplification in VR [44, 67] often aims at a subconscious effect, we aim to allow the user to easily and knowingly bridge the distance to remote surfaces, which could be close or very far away. Therefore, based on the reach direction, we cast a 3D cone, find the most likely intended drop surface, and set the amplification distance to the distance toward that surface. As soon as the user reaches out, object motion accelerates and then decelerates again until they reach the limit radius where the smooth step functions return values near 100%, returning to a fully linear regime at the target surface. If users reach back, the object quickly bridges the distance to their body again. If users reach sideways, we smoothly blend into non-amplified regime again (see amplification field in Fig. 16b), or hand them over into the next cone if detected.

Step 4: Drop. As the manipulated object enters the volume above a target surface, a small green surface-aligned disk below the object (e.g., see Fig. 6, $\boxed{c}$) indicates the ability to drop it. As the user releases their pinch, the object animates toward the indicated surface point.

6 Applications

Reality Promises enable new capabilities for a range of users across different application domains. For *entertainment applications*, Reality Promises add the capability of physicalized storytelling to MR games and narratives. For example, a game character can abduct a user's stuffed animal and make it reappear in a different place only if the user solves the game quest. Hiding the robot might increase the game's believability of exerting control over physical space, thus deepening immersion. For *remote collaboration applications*, Reality Promises can allow a remote interlocutor to take control of the local user's space, for example, in an avatar-based AR call between a local user and their friend cooking together in the kitchen. Rendering the remote interlocutor's avatar in lieu of the robot might increase the sense of co-presence. For *health applications*, Reality Promises might enable the capability of behavioral nudging, e.g., by spawning a banana on the table if the user did not have healthy food for two hours, thus presenting a subtle reminder to eat healthily. For *architecture applications*, Reality Promises could contribute to the field of responsive architecture where a space subtly re-configures itself to changing needs.

7 Limitations and Future Work

In the following, we compare our system implementation against the overarching vision of Reality Promises.

Scene Generalizability. Our system is portable and does not require any hardware setup in a new scene: In contrast to most applications and research that offload rendering to a dedicated GPU [30, 52, 62], we designed and implemented our splatting system for full on-device rendering, thereby preventing any wires. Furthermore, by tracking the robot inside-out, we eliminated the need for room-mounted tracking rigs. However, on the software side, any new scene and its objects must be captured, prepared, annotated, and imported into the system. In the future, we see potential to leverage the robot's sensor system for *automatic space and object capture* [38, 41], automatically moving objects out of the way for background scanning and capturing the objects themselves in the process, while leveraging advances in scene understanding [4, 68]. Furthermore, the robot might periodically set out for *automatic inventory discovery*, e.g., informing the *MagicMake* menu about available objects in the surroundings. At the same time, it could scout not only for new but even re-track known but moved objects. The creation of a Reality Promise could also trigger a robotic search for the desired object, rejecting the promise if the robot does not find it. Beyond the robot, this process might even be performed cooperatively with the user headset's sensors, similar to collaborative SLAM. Using the passthrough cameras–very recently opened for developer access in Meta Quest–can enable capturing the space en passant for splat reconstruction, object tracking, and object redetection. Advances in machine learning from transformers [6] to reinforcement learning [69] promise automatic affordance detection. Integrations with online webshops might even open up the way for *long-running promises* that start materializing immediately as the user orders the product and where only the last meters from the door front to the user's already chosen location is fulfilled by the robot.

Visual Fidelity. As part of our space setup procedure, we manually color-matched the reconstructed splat with the passthrough view's appearance ahead of time. Future research into photometric splat alignment [71] can improve the integration of the rendered splat patch with the passthrough view. Furthermore, more powerful GPUs, found in today's high-end devices such as Apple Vision Pro or to be expected from future hardware innovations, afford higher splat counts in a scene and will thus further improve the visual fidelity of the rendered splat.

Robotic Versatility. In this work, we employed the Stretch 3 with a 2-finger gripper on a Cartesian kinematic chain. Practically, this limits reliably performable manipulations to linear sequences with objects measuring 15 cm of diameter at maximum. Advances in dexterity promise opening doors, windows, and drawers, manipulating smaller objects, potentially even enabling dual-manipulator interactions such as opening bottles.

Interaction Versatility. In this paper, we presented pick-and-place-based Reality Promises that comprise *1)* communicating the user's manipulation intent to the system, *2)* virtually rendering the desired manipulation as a promised reality, and *3)* executing the manipulation sequence by means of the robot while visually replacing the physical for a virtual cause of manipulation. Slight system adaptions may enable Reality Promises for calling an elevator with a button press, operating a light switch, closing window curtains, turning a door knob, or pouring liquid from a bottle. In the longer-term future, the same three steps can be considered in order to enable Reality Promises that manage more complex spatial manipulations, e.g., filling the dishwasher. Furthermore, additional interactions can be studied to handle longer fulfillment durations ranging from allowing promise cancellations mid-way to predicting interaction intent based on user behavior and auto-triggering a Reality Promise to minimize the intent-to-fulfillment duration.

Human Factors. *User safety* is a core aspect of any robotic system, more so in a system that visually hides a potentially close robot. While our system technically supports free mobility, as we synthesize the splat from any 6 DoF view, we assume a user that is seated during close robotic operation. For more elaborate approaches of collision prevention, the need for collision course detection arises, e.g., by use of spatiotemporal motion extrapolation to predict potentially dangerous path crossings. Once a collision course is detected, three tactics suggest themselves. First, interrupt *the experience* by breaking the illusion and revealing the robot. Second, diverge *the robot*, i.e., by stopping or re-routing it. Third, diverge *the user* by rendering virtual content into their collision course. The last option is particularly suitable for gaming applications, where the user can be made believe that the diversion is part of the intended story (e.g., a fire breaking out in front of them to make them jump, when in fact, the fire only breaks out to diverge them from colliding with the robot) [11]. Furthermore, investigating *user acceptance* of an invisible mobile robot that might be hidden anywhere in space remains a question for user-study-based future research.

8 Conclusion

Weiser stated that the “most profound technologies are those that disappear” [66], culminating in notions such as Norman's *Invisible Computing* [47]. In this work, we reinterpreted this vision literally and introduced the novel concept of Reality Promises that abstract away all intricacies of robotic operations, including the robot itself, behind a carefully designed “reality interface”. We demonstrated their use to control the user's experience across two branches of reality that are decoupled and later recoupled through a full-scale mobile robotic manipulation system, an on-HMD, robot-aware 3D Gaussian splatting system, and an interaction system for reality manipulations. We demonstrated how to achieve tight conceptual integration and technical orchestration between them to enable experiences of magic motions, materializations, menus, and monsters in an effort to get a step closer to Sutherland's *Ultimate Display* [56], where the computer has control over the existence of matter.

Acknowledgments

We thank Cyrus Vachha for his support in space and object scanning and Lauren Wang for acting in the video. We thank Andreas Fender for fruitful discussions and providing feedback on the manuscript. We thank Ching-Yi Tsai for feedback. This research was supported by the Princeton Presidential Postdoctoral Research Fellowship Program and the Princeton SEAS Innovation Fund.

References

[1] Parastoo Abtahi, Benoit Landry, Jackie Yang, Marco Pavone, Sean Follmer, and James A Landay. 2019. 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*. ACM, 1–13. https://doi.org/10.1145/3290605.3300589

[2] Angelos Angelopoulos, Austin Hale, Husam Shaik, Akshay Paruchuri, Ken Liu, Randal Tuggle, and Daniel Szafir. 2022. Drone Brush: Mixed Reality Drone Path Planning. In *Proceedings of the 2022 17th ACM/IEEE International Conference on Human-Robot Interaction*. IEEE, 678–682. https://doi.org/10.1109/HRI53351.2022.9889504

[3] Bruno Araujo, Ricardo Jota, Varun Perumal, Jia Xian Yao, Karan Singh, and Daniel Wigdor. 2016. Snake Charmer: Physically Enabling Virtual Objects. In *Proceedings of the 10th International Conference on Tangible, Embedded, and Embodied Interaction*. ACM, 218–226. https://doi.org/10.1145/2839462.2839484

[4] Armen Avetisyan, Christopher Xie, Henry Howard-Jenkins, Tsun-Yi Yang, Samir Aroudj, Suvam Patra, Fuyang Zhang, Duncan Frost, Luke Holland, Campbell Orme, et al. 2024. SceneScript: Reconstructing Scenes with an Autoregressive Structured Language Model. In *Proceedings of the 2024 European Conference on Computer Vision*. Springer, 247–263. https://doi.org/10.1007/978-3-031-73030-6_14

[5] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andrew D Wilson. 2016. Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1968–1979. https://doi.org/10.1145/2858036.2858226

[6] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. 2023. Affordances from Human Videos as a Versatile Representation for Robotics. In *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 13778–13790. https://doi.org/10.1109/CVPR52729.2023.01324

[7] Yuanzhi Cao, Zhuangying Xu, Fan Li, Wentao Zhong, Ke Huo, and Karthik Ramani. 2019. V.Ra: An In-Situ Visual Authoring System for Robot-IoT Task Planning With Augmented Reality. In *Proceedings of the 2019 on Designing Interactive Systems Conference*. ACM, 1059–1070. https://doi.org/10.1145/3322276.3322278

[8] Alexandre Carlier. 2024. Gaussian Splatting Blender Addon. https://github.com/ReshotAI/gaussian-splatting-blender-addon.

[9] Jiaqi Chen, Boyang Sun, Marc Pollefeys, and Hermann Blum. 2024. A 3D Mixed Reality Interface for Human-Robot Teaming. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation*. IEEE, 11327–11333. https://doi.org/10.1109/ICRA57147.2024.10611017

[10] Lung-Pan Cheng, Yi Chen, Yi-Hao Peng, and Christian Holz. 2023. Reality Rifts: Wonder-ful Interfaces by Disrupting Perceptual Causality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, 1–15. https://doi.org/10.1145/3544548.3581454

[11] Lung-Pan Cheng, Eyal Ofek, Christian Holz, and Andrew D Wilson. 2019. VRoamer: Generating On-The-Fly VR Experiences While Walking inside Large, Unknown Real-World Building Environments. In *Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces*. IEEE, 359–366. https://doi.org/10.1109/VR.2019.8798074

[12] Yi Fei Cheng, Hang Yin, Yukang Yan, Jan Gugenheimer, and David Lindlbauer. 2022. Towards Understanding Diminished Reality. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, 1–16. https://doi.org/10.1145/3491102.3517452

[13] Francesco I Cosco, Carlos Garre, Fabio Bruno, Maurizio Muzzupappa, and Miguel A Otaduy. 2009. Augmented Touch Without Visual Obtrusion. In *Proceedings of the 2009 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 99–102. https://doi.org/10.1109/ISMAR.2009.5336492

[14] Émilie Fabre, Jun Rekimoto, and Yuta Itoh. 2024. Exploring the Kuroko Paradigm: The Effect of Enhancing Virtual Humans with Reality Actuators in Augmented Reality. In *Proceedings of the 2024 Augmented Humans International Conference*. ACM, 79–90. https://doi.org/10.1145/3652920.3652945

[15] Wen Fan, Xiaoqing Guo, Enyang Feng, Jialin Lin, Yuanyi Wang, Jiaming Liang, Martin Garrad, Jonathan Rossiter, Zhengyou Zhang, Nathan Lepora, et al. 2023. Digital Twin-Driven Mixed Reality Framework for Immersive Teleoperation with Haptic Rendering. *IEEE Robotics and Automation Letters* 8, 12 (2023), 8494–8501. https://doi.org/10.1109/LRA.2023.3325784

[16] Mehrad Faridan, Marcus Friedel, and Ryo Suzuki. 2022. UltraBots: Large-Area Mid-Air Haptics for VR with Robotically Actuated Ultrasound Transducers. In *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. ACM, 1–3. https://doi.org/10.1145/3526114.3561350

[17] Andreas Rene Fender and Christian Holz. 2022. Causality-Preserving Asynchronous Reality. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, 1–15. https://doi.org/10.1145/3491102.3501836

[18] Daniel Friedman and David Wise. 1978. Aspects of Applicative Programming for Parallel Processing. *IEEE Trans. Comput.* 27, 4 (1978), 289–296. https://doi.org/10.1109/TC.1978.1675100

[19] Anna Fuste, Ben Reynolds, James Hobin, and Valentin Heun. 2020. Kinetic AR: A Framework for Robotic Motion Systems in Spatial Computing. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 1–8. https://doi.org/10.1145/3334480.3382814

[20] Pablo Soler Garcia, Petar Lukovic, Lucie Reynaud, Andrea Sgobbi, Federica Bruni, Martin Brun, Marc Zünd, Riccardo Bollati, Marc Pollefeys, Hermann Blum, et al. 2024. HoloSpot: Intuitive Object Manipulation via Mixed Reality Drag-and-Drop. *arXiv preprint* (2024), 1–7. https://doi.org/10.48550/arXiv.2410.11110

[21] Ryota Gomi, Ryo Suzuki, Kazuki Takashima, Kazuyuki Fujita, and Yoshifumi Kitamura. 2024. InflatableBots: Inflatable Shape-Changing Mobile Robots for Large-Scale Encountered-Type Haptics in VR. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. ACM, 1–14. https://doi.org/10.1145/3613904.3642069

[22] Jake Guida and Misha Sra. 2020. Augmented Reality World Editor. In *Posters of the 26th ACM Symposium on Virtual Reality Software and Technology*. ACM, 1–2. https://doi.org/10.1145/3385956.3422125

[23] Anuruddha Hettiarachchi and Daniel Wigdor. 2016. Annexing Reality: Enabling Opportunistic Use of Everyday Objects as Tangible Proxies in Augmented Reality. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1957–1967. https://doi.org/10.1145/2858036.2858134

[24] Matthias Hoppe, Pascal Knierim, Thomas Kosch, Markus Funk, Lauren Futami, Stefan Schneegass, Niels Henze, Albrecht Schmidt, and Tonja Machulla. 2018. VRHapticDrones: Providing Haptics in Virtual Reality Through Quadcopters. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 7–18. https://doi.org/10.1145/3282894.3282898

[25] Eduardo Iglesius, Masato Kobayashi, Yuki Uranishi, and Haruo Takemura. 2024. MRNaB: Mixed Reality-Based Robot Navigation Interface using Optical-See-Through MR-Beacon. *Advanced Robotics* 39, 11 (2024), 663–677. https://doi.org/10.1080/01691864.2025.2508779

[26] Keiichi Ihara, Mehrad Faridan, Ayumi Ichikawa, Ikkaku Kawaguchi, and Ryo Suzuki. 2023. HoloBots: Augmenting Holographic Telepresence With Mobile Robots for Tangible Remote Collaboration in Mixed Reality. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. ACM, 1–12. https://doi.org/10.1145/3586183.3606727

[27] Bryce Ikeda, Maitrey Gramopadhye, LillyAnn Nekervis, and Daniel Szafir. 2025. MARCER: Multimodal Augmented Reality for Composing and Executing Robot Tasks. In *Proceedings of the 2025 ACM/IEEE International Conference on Human-Robot Interaction*. IEEE, 529–539. https://doi.org/10.1109/HRI61500.2025.10974232

[28] Bryce Ikeda and Daniel Szafir. 2021. An AR Debugging Tool for Robotics Programmers. In *4th International Workshop on Virtual, Augmented, and Mixed Reality for HRI at HRI 2021*. 1–4.

[29] Bryce Ikeda and Daniel Szafir. 2024. PRogramAR: Augmented Reality End-User Robot Programming. *ACM Transactions on Human-Robot Interaction* 13, 1 (2024), 1–20. https://doi.org/10.1145/3640008

[30] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. 2024. VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality. In *Proceedings of ACM SIGGRAPH 2024*. ACM, 1–11. https://doi.org/10.1145/3641519.3657448

[31] Hiroki Kaimoto, Kyzyl Monteiro, Mehrad Faridan, Jiatong Li, Samin Farajian, Yasuaki Kakehi, Ken Nakagaki, and Ryo Suzuki. 2022. Sketched Reality: Sketching Bi-Directional Interactions between Virtual and Physical Worlds with AR and Actuated Tangible UI. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. ACM, 1–12. https://doi.org/10.1145/3526113.3545626

[32] Mohamed Kari, Tobias Grosse-Puppendahl, Luis Falconeri Coelho, Andreas Rene Fender, David Bethge, Reinhard Schütte, and Christian Holz. 2021. TransforMR: Pose-Aware Object Substitution for Composing Alternate Mixed Realities. In *Proceedings of the 2021 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 69–79. https://doi.org/10.1109/ISMAR52148.2021.00021

[33] Mohamed Kari, Reinhard Schütte, and Raj Sodhi. 2023. Scene Responsiveness for Visuotactile Illusions in Mixed Reality. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. ACM, 1–15. https://doi.org/10.1145/3586183.3606825

[34] Mark Kellogg. 2023. 3D Gaussian Splatting for Three.js. https://github.com/mkkellogg/GaussianSplats3D.

[35] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023), 139:1–139:14. https://doi.org/10.1145/3592433

[36] Luv Kohli. 2010. Redirected Touching: Warping Space to Remap Passive Haptics. In *Proceedings of the 2010 IEEE Symposium on 3D User Interfaces*. IEEE, 129–130. https://doi.org/10.1109/3DUI.2010.5444703

[37] Oliver Lemke, Zuria Bauer, René Zurbrügg, Marc Pollefeys, Francis Engelmann, and Hermann Blum. 2024. Spot-Compose: A Framework for Open-Vocabulary Object Retrieval and Drawer Manipulation in Point Clouds. In *2nd Workshop on Mobile Manipulation and Embodied Intelligence at ICRA 2024*. https://doi.org/10.48550/arXiv.2404.12440

[38] Yuetao Li, Zijia Kuang, Ting Li, Guyue Zhou, Qun Hao, Zike Yan, Guyue Zhou, and Shaohui Zhang. 2025. ActiveSplat: High-Fidelity Scene Reconstruction Through Active Gaussian Splatting. *IEEE Robotics and Automation Letters* 10, 8 (2025), 8099 – 8106. https://doi.org/10.1109/LRA.2025.3580331

[39] David Lindlbauer and Andy D Wilson. 2018. Remixed Reality: Manipulating Space and Time in Augmented Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 1–13. https://doi.org/10.1145/3173574.3173703

[40] Barbara Liskov and Liuba Shrira. 1988. Promises: Linguistic Support for Efficient Asynchronous Procedure Calls in Distributed Systems. *ACM SIGPLAN Notices* 23, 7 (1988), 260–267. https://doi.org/10.1145/960116.54016

[41] Zhuoyue Lyu, Jackie Yang, Monica S Lam, and James A Landay. 2022. HomeView: Automatically Building Smart Home Digital Twins With Augmented Reality Headsets. In *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. ACM, 1–6. https://doi.org/10.1145/3526114.3558709

[42] Sebastian Marwecki, Andrew D Wilson, Eyal Ofek, Mar Gonzalez Franco, and Christian Holz. 2019. Mise-Unseen: Using Eye Tracking to Hide Virtual Reality Scene Changes in Plain Sight. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, 777–789. https://doi.org/10.1145/3332165.3347919

[43] William A McNeely. 1993. Robotic Graphics: A New Approach to Force Feedback for Virtual Reality. In *Proceedings of the 1993 IEEE Virtual Reality Annual International Symposium*. IEEE, 336–341. https://doi.org/10.1109/VRAIS.1993.380761

[44] Roberto A Montano Murillo, Sriram Subramanian, and Diego Martinez Plasencia. 2017. Erg-O: Ergonomic Optimization of Immersive Virtual Environments. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 759–771. https://doi.org/10.1145/3126594.3126605

[45] Shohei Mori, Okan Erat, Wolfgang Broll, Hideo Saito, Dieter Schmalstieg, and Denis Kalkofen. 2020. InpaintFusion: Incremental RGB-D Inpainting for 3D Scenes. *IEEE Transactions on Visualization and Computer Graphics* 26, 10 (2020), 2994–3007. https://doi.org/10.1109/TVCG.2020.3003768

[46] Shohei Mori, Fumihisa Shibata, Asako Kimura, and Hideyuki Tamura. 2015. Efficient Use of Textured 3D Model for Pre-Observation-Based Diminished Reality. In *Workshops of the 2015 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 32–39. https://doi.org/10.1109/ISMARW.2015.16

[47] Donald Norman. 1998. *The Invisible Computer*. MIT Press.

[48] Alexander Plopski, Ada Virginia Taylor, Elizabeth Jeanne Carter, and Henny Admoni. 2019. InvisibleRobot: Facilitating Robot Manipulation Through Diminished Reality. In *Adjunct Proceedings of the 2019 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 165–166. https://doi.org/10.1109/ISMAR-Adjunct.2019.00-55

[49] Michael I Posner, Mary J Nissen, and Raymond M Klein. 1976. Visual Dominance: An Information-Processing Account of Its Origins and Significance. *Psychological Review* 83, 2 (1976), 157–171. https://doi.org/10.1037/0033-295X.83.2.157

[50] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*. ACM, 79–80. https://doi.org/10.1145/237091.237102

[51] Sharif Razzaque, Zachariah Kohn, and Mary C Whitton. 2001. Redirected Walking. In *Proceedings of Eurographics*. Eurographics Association, 1–6. https://doi.org/10.2312/egs.20011036

[52] Hannah Schieber, Jacob Young, Tobias Langlotz, Stefanie Zollmann, and Daniel Roth. 2025. Semantics-Controlled Gaussian Splatting for Outdoor Scene Reconstruction and Rendering in Virtual Reality. In *Proceedings of the 2025 IEEE Conference Virtual Reality and 3D User Interfaces*. IEEE, 318–328. https://doi.org/10.1109/VR59515.2025.00056

[53] Lior Shapira and Daniel Freedman. 2016. Reality Skins: Creating Immersive and Tactile Virtual Environments. In *Proceedings of the 2016 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 115–124. https://doi.org/10.1109/ISMAR.2016.23

[54] Adalberto L Simeone, Eduardo Velloso, and Hans Gellersen. 2015. Substitutional Reality: Using the Physical Environment to Design Virtual Reality Experiences. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3307–3316. https://doi.org/10.1145/2702123.2702389

[55] Kazuya Sugimoto, Hiromitsu Fujii, Atsushi Yamashita, and Hajime Asama. 2014. Half-Diminished Reality Image Using Three RGB-D Sensors for Remote Control Robots. In *Proceedings of the 2014 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE, 1–6. https://doi.org/10.1109/SSRR.2014.7017676

[56] Ivan E Sutherland. 1965. The Ultimate Display. In *Proceedings of the 1965 IFIP Congress*. IFIP, 1–3.

[57] Ryo Suzuki, Hooman Hedayati, Clement Zheng, James L Bohn, Daniel Szafir, Ellen Yi-Luen Do, Mark D Gross, and Daniel Leithinger. 2020. 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*. ACM, 1–11. https://doi.org/10.1145/3313831.3376523

[58] Ryo Suzuki, Adnan Karim, Tian Xia, Hooman Hedayati, and Nicolai Marquardt. 2022. Augmented Reality and Robotics: A Survey and Taxonomy for AR-Enhanced Human-Robot Interaction and Robotic Interfaces. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, 1–33. https://doi.org/10.1145/3491102.3517719

[59] Ryo Suzuki, Eyal Ofek, Mike Sinclair, Daniel Leithinger, and Mar Gonzalez-Franco. 2021. HapticBots: Distributed Encountered-Type Haptics for VR With Multiple Shape-Changing Mobile Robots. In *Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology*. ACM, 1269–1281. https://doi.org/10.1145/3472749.3474821

[60] Ada V Taylor, Ayaka Matsumoto, Elizabeth J Carter, Alexander Plopski, and Henny Admoni. 2020. Diminished Reality for Close Quarters Robotic Telemanipulation. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 11531–11538. https://doi.org/10.1109/IROS45743.2020.9341536

[61] Wen-Jie Tseng, Elise Bonnail, Mark McGill, Mohamed Khamis, Eric Lecolinet, Samuel Huron, and Jan Gugenheimer. 2022. The Dark Side of Perceptual Manipulations in Virtual Reality. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, 1–15. https://doi.org/10.1145/3491102.3517728

[62] Xuechang Tu, Bernhard Kerbl, and Fernando de la Torre. 2024. Fast and Robust 3D Gaussian Splatting for Virtual Reality. In *Posters of ACM SIGGRAPH Asia 2024*. ACM, 1–3. https://doi.org/10.1145/3681756.3697947

[63] Michael Walker, Thao Phung, Tathagata Chakraborti, Tom Williams, and Daniel Szafir. 2023. Virtual, Augmented, and Mixed Reality for Human-Robot Interaction: A Survey and Virtual Design Element Taxonomy. *ACM Transactions on Human-Robot Interaction* 12, 4 (2023), 1–39. https://doi.org/10.1145/3597623

[64] Jun Wang, Chun-Cheng Chang, Jiafei Duan, Dieter Fox, and Ranjay Krishna. 2024. EVE: Enabling Anyone to Train Robots Using Augmented Reality. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. ACM, 1–13. https://doi.org/10.1145/3654777.3676413

[65] Keru Wang, Zhu Wang, Ken Nakagaki, and Ken Perlin. 2024. “Push-That-There”: Tabletop Multi-Robot Object Manipulation via Multimodal Object-Level Instruction. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference*. ACM, 2497–2513. https://doi.org/10.1145/3643834.3661542

[66] Mark Weiser. 1991. The Computer for the 21st Century. *Scientific American* 265, 3 (1991), 94–104. https://doi.org/10.1038/scientificamerican0991-94

[67] Johann Wentzel, Greg d’Eon, and Daniel Vogel. 2020. Improving Virtual Reality Ergonomics Through Reach-Bounded Non-Linear Input Amplification. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 1–12. https://doi.org/10.1145/3313831.3376687

[68] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. 2023. TidyBot: Personalized Robot Assistance with Large Language Models. *Autonomous Robots* 47, 8 (2023), 1087–1102. https://doi.org/10.1007/s10514-023-10139-z

[69] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. 2023. Learning Generalizable Dexterous Manipulation from Human Grasp Affordance. In *Proceedings of the 6th Conference on Robot Learning*. PMLR, 618–629.

[70] Zhen Xu. 2024. Fast Gaussian Rasterization. https://github.com/dendenxu/fast-gaussian-rasterization.

[71] Ziwen Yuan, Tianyi Zhang, Matthew Johnson-Roberson, and Weiming Zhi. 2024. PhotoReg: Photometrically Registering 3D Gaussian Splatting Models. *arXiv preprint* (2024), 1–7. https://doi.org/10.48550/arXiv.2410.05044

[72] Ya-Ting Yue, Yong-Liang Yang, Gang Ren, and Wenping Wang. 2017. SceneCtrl: Mixed Reality Enhancement via Efficient Scene Editing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 427–436. https://doi.org/10.1145/3126594.3126601

[73] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2001. EWA Volume Splatting. In *Proceedings of the 2001 Conference on Visualization*. IEEE, 29–538. https://doi.org/10.1109/VISUAL.2001.964490

A Failure Mode Analysis of the Pick-and-Place Robotic Pipeline

We performed 20 pick-and-place sequences for 2 objects each (coconut water, chips container). Of these 40 operations, 85% (34x) were completed successfully. The failure modes were as follows:

- 5% (= 2x) failed *grab* (1x coconut water, 1x chips container)
- 2.5% (1x) successful *grab*, but failed *drop* hitting the edge of the table (chips container)
- 2.5% (1x) objects fell down during transport due to bad grip (chips container)
- 5% (2x) the low-level robot driver entered an erroneous state and refused to accept commands.

For successful cases, the distance between physically achieved end pose and virtually defined target pose was 3.4 cm on average.