ProObjAR: Prototyping Spatially-aware Interactions of Smart Objects with AR-HMD

Hui Ye
School of Creative Media,
City University of Hong Kong
Hong Kong, China
huiye4@cityu.edu.hk

Jiaye Leng
State Key Laboratory of Virtual Reality Technology and Systems,
Beihang University
Beijing, China
lengjiaye@buaa.edu.cn

Chufeng Xiao
School of Creative Media,
City University of Hong Kong
Hong Kong, China
chufeng.xiao@my.cityu.edu.hk

Lili Wang
State Key Laboratory of Virtual Reality Technology and Systems,
Beihang University
Beijing, China
wanglily@buaa.edu.cn

Hongbo Fu*
School of Creative Media,
City University of Hong Kong
Hong Kong, China
hongbofu@cityu.edu.hk

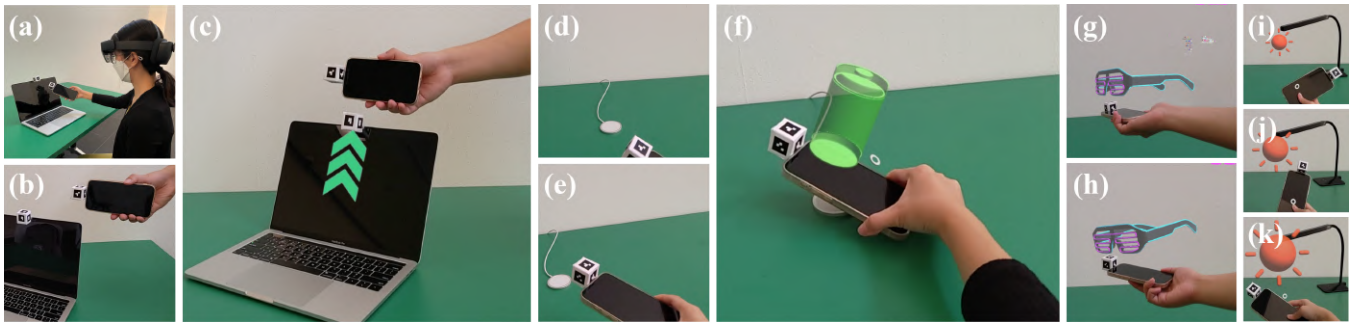


Figure 1: We present *ProObjAR*, an all-in-one system enabling designers to easily prototype spatially-aware interactions among real objects. (a) A designer wearing an Augmented Reality Head Mounted Display (AR-HMD) manipulates real objects (attached with AR markers for tracking) to set up triggering events and the corresponding effects, e.g., "The computer will send data to the mobile phone if the phone appears over the computer". (b)&(c) show the testing result in situ viewed from the AR-HMD, i.e., the user-specified event triggers the data transmission effect (represented by a green arrow). (d)-(k) show three additional prototyping results: (d)-(f) "A phone will be charged if it is placed at a certain position", (g)-(h) "The 3D poses of a phone and a 3D model are synchronous", and (i)-(k) "The lightness of a lamp can be adjusted by changing the orientation of a phone".

ABSTRACT

The rapid advances in technologies have brought new interaction paradigms of smart objects (e.g., digital devices) beyond digital device screens. By utilizing spatial properties, configurations, and movements of smart objects, designing spatial interaction, which is one of the emerging interaction paradigms, efficiently promotes engagement with digital content and physical facility. However, as an important phase of design, prototyping such interactions still remains challenging, since there is no ad-hoc approach for this emerging paradigm. Designers usually rely on methods that require fixed hardware setup and advanced coding skills to script and validate early-stage concepts. These requirements restrict the design process to a limited group of users in indoor scenes. To facilitate the prototyping to general usages, we aim to figure out the design difficulties and underlying needs of current design processes for spatially-aware object interactions by empirical studies. Besides, we explore the design space of the spatial interaction for smart objects and discuss the design space in an input-output spatial interaction model. Based on these findings, we present *ProObjAR*, an all-in-one novel prototyping system with an Augmented Reality Head Mounted Display (AR-HMD). Our system allows designers to easily obtain the spatial data of smart objects being prototyped, specify spatially-aware interactive behaviors from an input-output event triggering workflow, and test the prototyping results in situ. From the user study, we find that *ProObjAR* simplifies the design

*Corresponding author.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3580750

procedure and increases design efficiency to a large extent and thus advancing the development of spatially-aware applications in smart ecosystems.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing**; **Ubiquitous and mobile computing systems and tools**; **Interaction design process and methods**; **Mixed / augmented reality**.

KEYWORDS

spatial interaction, smart objects, AR prototyping

ACM Reference Format:

Hui Ye, Jiaye Leng, Chufeng Xiao, Lili Wang, and Hongbo Fu. 2023. *ProObjAR*: Prototyping Spatially-aware Interactions of Smart Objects with AR-HMD. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3544548.3580750

1 INTRODUCTION

Smart objects [31] include both digital devices (e.g., tabletops, mobile phones, tablets, smartwatches, mouses) and non-digital objects (e.g., clocks, cups, pens) that can be embedded to smart space. They are designed to enable ubiquitous computing in smart environments. The rapid advances in technologies have brought new interaction paradigms for smart objects, beyond the boundary of digital screens. In particular, emerging sensing technologies endow smart objects with spatial awareness, which allows interactive spaces to react to the spatial movements and configurations of objects [27]. By manipulating the proximity [4] and spatial configurations [43] of object(s), people can interact with them to facilitate natural and intuitive cognition [20]. Thus such spatially-aware interactions have a wealth of applications from evoking and controlling screen-based functions using device movements [1, 44, 47] to exchanging information among cross/networked devices using proximity [6, 15, 39, 43].

Before the development stage, these applications are established from an iterative design cycle, where prototyping is a significant early-stage phase to enable designers to rapidly traverse and validate design ideas and concepts [19]. Designers typically resort to traditional prototyping approaches such as making videos and paper mockups in their workflow. However, these approaches are difficult to depict the realistic and interactive aspects of spatially-aware interactions for smart objects [3, 32, 51]. Besides, heavy coding tasks are required to define dynamic and interactive spatially-aware object behaviors. These approaches have very high entry barriers, so they are not the first choices of designers for rapid prototyping.

Driven by technologies, digitally mediated design processes have been explored to mitigate the design challenges and lower the design barriers for smart objects interactions [21, 22, 24–27, 37, 41]. However, a majority of the previously proposed design toolkits focus on scripting the communications and UIs among digital devices [21, 22, 24, 25, 41]. Very recently, Jetter et al. [27] propose a sketching- and simulation-based solution to design spatially-aware interactive spaces in VR. It provides a simulated tracking system to prototype spatial interactions of multiple digital devices. Although its design space in VR contains large degrees of freedom, there still exists a gap [3, 32] between the virtual designing and the spatial interactions happening in physical reality of real-world object(s).

Augmented Reality (AR) provides designers with new prototyping opportunities in real-world scenes. However, most of the AR prototyping tools focus on prototyping situated AR experiences from the perspective of end users [35, 42, 49] or spatial interactions of real-world users [48, 51]. Object-oriented prototyping and authoring tools with AR, are mainly designed to discover and localize static things [23] or function controlling over static appliances [38]. The advantages and capabilities of AR prototyping motivate us to explore it as an approach to designing spatially-aware interactions of smart objects, especially with dynamic spatial movements and configurations.

To figure out the existing challenges and underlying needs of the current design workflows for designing spatial interactions of smart objects, we first conduct an empirical study by interviewing expert researchers and designers. Based on the interview findings, we summarize their motivations and methods, prototyping difficulties, and their expectations for an ideal solution. Then, we explore the spatial interaction design space from the dimensions of *Quantity*, *Proximity*, *Movement Form*, and *Interaction Space* for smart objects, and discuss the design space in an input-output model. From the interview findings and the design space, we design and develop *ProObjAR*, an AR-HMD-based prototyping system, which enables designers to obtain the spatial properties of smart objects, specify the spatially-aware interactions from an input-output event triggering workflow, and test the prototyping results efficiently. Specifically, *ProObjAR* allows designers to obtain the 6-DoF poses (i.e., 3D positions and 3D orientations) of one or multiple smart object(s) in real-time. Then designers follow an input-output event-driven workflow to specify the triggering spatial events by performing certain interactions as input events and creating virtual assets or sketches with the specified effects as output effects. Finally, designers can test the prototyping interactions by manipulating the real-world objects easily. To evaluate the proposed system, we conduct a usability study among both professional users with spatial interaction design experience and casual users. From the observation of their prototyping process, the analysis of their diverse prototyping results, and the discussions with them, we find *ProObjAR* is an easy-to-use and efficient prototyping solution for spatially-aware interactions among smart objects.

The main contributions of this work are fourfold:

- From the depth-interviews, we identify the motivations and typical prototyping methods, prototyping challenges and difficulties, and implications of designing spatially-aware interactions for smart objects.
- From the analysis of existing literature and the findings of the empirical study, we explore the design space of the spatially-aware interactions and put the design space in an input-output model.
- According to the previous findings, we propose a prototyping interface for designers to rapidly specify the dynamic spatial interactions in the event-triggering workflow and test the results easily.
- We demonstrate a wide range of potential application scenarios from a preliminary user study to validate the expressiveness and usability of the proposed system.

Table 1: The differences between our work and closely related AR/VR authoring/designing works.

	Target User	Goal	Interaction of Interest	AR/VR
ProObjAR [Ours]	Designers	Design objects' spatial interactions	Real-world objects	AR
GesturAR [48]	AR consumers	Create freehand AR applications	Hand gestures	AR
In VR [27]	Designers	Simulate spatially-aware interactive spaces	Virtual objects	VR
Pronto [35]	Designers	Design AR experiences	Augmented videos	AR
ProGesAR [51]	Designers	Design IoT-enhanced functions	Real users	AR

2 RELATED WORK

2.1 Spatial Interactions of Smart Objects

The concept of smart objects was first proposed by Kortuem et al. [31] to enable novel computing applications in smart environments with the capacities of real-world awareness and interactivity. Both the academia and industries have explored diverse spatially-aware interactions across one or multiple smart objects for certain applications, such as lift-to-wake functions of device screens [1], pick-and-drop interaction techniques between laptops, tablets, and wall displays [44], grabbing/throwing/flicking actions on multiple surfaces [47], and the proximity interactions among multiple tablets [36, 39, 43]. Through elicitation studies, the spatially-aware interactions in cross-device/surface configurations are designed for information exchange [15], device presence and connection [33], and other typical cross-device commands [43]. Besides the digital devices, spatially-aware non-digital objects (e.g., cups, trash cans) are also explored in certain applications such as managing photo collections in AR scenes [10]. Although these spatial object interactions have been widely applied in various scenarios, there does not exist a systematic summary of how the interaction space is developed and can be used. We fill in this gap by discussing the potential space of spatial object interaction design from four dimensions. To help better understand the design space and inform further studies, we propose an input-out model that formulates several taxonomies of spatially-interaction design.

2.2 Designing Spatial Interactions of Smart Objects

Designing spatially-aware object interactions is an essential procedure in developing spatial applications. Since it is an emerging interaction paradigm, traditional tools are not specially developed for such design tasks. Driven by technologies, digitally mediated design processes [21, 22, 24–27, 41] have been explored to mitigate the design challenges and lower the design barriers for smart-object interactions. However, a majority of the proposed design toolkits are designed for developers to help script the communications and UIs among digital devices. For example, researchers have proposed various toolkits to foster the designing process for multi-/cross-digital device interactions, such as using a smartphone as look-through lens for tabletop-centric cross-device interaction [22] and an event-driven platform for smartwatch-centric cross-device applications [21]. These toolkits are primarily designed for prototyping the touch-gesture-based interactions in the space of the device screen. To design for spatial interactions beyond the screens, researchers have explored simulation-based methods such as a toolkit that simulates spatially-aware interactions of virtual devices (Table 1) in VR scenes [27]. Although these toolkits provide a large degree of freedoms for scenario variance, they are designed for prototyping in the virtual space instead of the physical reality. In contrast, our system is designed for prototyping spatial interactions of real-world objects. Users can perform real-world manipulation of objects to specify and test the interactive behaviors. Researchers have also proposed prototyping toolkits based on different types of sensors and hardware [28, 39, 46, 50]. However, these toolkits are highly dependent on a fixed scene settled with sensors and hardware. Compared with them, our system *ProObjAR* is more flexible, since it enables users to perform prototyping in situ by wearing an AR-HMD, without the requirement of additional sensor or hardware.

2.3 AR Authoring and Designing Tools

AR provides designers with new prototyping opportunities in real-world scenes. AR technologies allow designers to explore, watch, and test their design concepts by creating virtual effects in and around physical objects, using mobile and wearable devices. It bridges the physical usage space and virtual design space, and thus can facilitate the rapid and easy exploration of spatial awareness for smart objects. However, existing works on AR prototyping focus on user-centered experience prototyping [35, 42, 49], instead of object-oriented interaction prototyping (see the "Interaction of Interest" column in Table 1). In contrast, we aim to provide an interface for prototyping object-centered spatial interactions, where users can trigger the spatial events by performing physical interactions of real-world objects from the AR interface. GesturAR [48] presents an AR-HMD interface for authoring hand-gesture-centric interactions. Different from its target for common AR consumers to create freehand AR applications, we aim to provide designers with a toolkit to design the spatial interactions of smart objects by manipulating real-world objects. Compared to these works, our work enables designers to explore the input design space around (real-world) object(s). We expect our exploration of transferring the concept to objects to open a segment of new opportunities for prototyping AR applications. Other types of object-oriented AR authoring toolkits are mainly designed to discover and localize static things [23] or function controlling over static appliances [38]. Unlike them, our system is applicable to the design of dynamic spatial behavior of smart objects. Several works [8, 16] have explored authoring dynamic movements of robots for human-robot/robot-IoT collaboration tasks. However, there are few works that explore AR prototyping for spatial interactions of smart objects, especially with dynamic spatial movements and configurations. Thus, our work bridges this gap by allowing users to prototype more general types of spatial events of smart objects.

Table 2: Background summary of the interviewees.

ID	Occupation	Working Experience	Coding Skills	Coding at Work	AR Experience	Prominent Project Experience
P1	UX Designer	2.7 years	Medium	No	No	Using device movement and proximity to discover, synchronize, connect devices and wake interactions
P2	Spatial Interaction Designer	1.5 years	High	Yes	Used and designed AR applications	Using device proximity to discover, synchronize, connect, and control devices
P3	Interaction Designer	5.5 years	High	Yes	Used AR applications	Using device proximity and hand gestures to discover, connect, and share data
P4	Researcher	4.0 years	High	Yes	Used and designed AR applications	Using device distance to pair devices

2.4 Visual Programming Interfaces

Specifying interactive behaviors usually requires programming to define the dynamic flows. To reduce the entry barriers, visual programming approaches [5, 7] have been proposed using visual expressions such as diagrams [48], free-hand sketches [13], and icons [2, 12, 45] to define the input, output, and their connection. In particular, some recent AR/VR interfaces [8, 9, 48, 49] integrate visual programming modalities to pair the user actions and the virtual contents. GesturAR [48] presents a trigger-action visual programming interface to author the customized AR hand interactions. CAPturAR [49] allows users to create and experience context-aware human applications through a flexible visual triggering logic. However, these visual programming interfaces explore human- or robot-related applications, while we focus on exploring the spatial interaction of smart objects. In our input-output event triggering workflow, we follow the design of visual programming by providing diagrams, connection lines, and logic operators for defining the input and output modalities so that the different types of spatial events, simulated effects, and their relationships can be defined easily and rapidly.

3 EMPIRICAL STUDY

To better understand the current workflows of smart-object spatial interaction design and identify the involved challenges, we conducted a qualitative, semi-structured interview study with four design experts.

3.1 Interviewees

We recruited four interviewees by purposefully approaching them from our personal and research networks. Three of them were designers (P1-P3) with 1.5-5.5 years of working experience in multiple-device/cross-device/spatial interaction design, and one is a researcher (P4) with 4-year research experience in spatial interaction design. They had been working on several projects on designing spatial interactions (e.g., proximity-based interactions) of one/multiple digital device(s) (e.g., tablets, smartphones, wireless headphones, smartwatches, household IoT devices) for various application scenarios (e.g., device discovery, connection, information exchange, controlling, synchronization). All of them had coding skills, and had utilized programming in their work except P1. All of them except P1 had AR experience including playing AR apps on mobile phones and AR-HMDs and designing AR products. Table 2 summarizes the background of the interviewees.

3.2 Interview Protocol

We conducted semi-structured depth interviews remotely using video or audio calls with the interviewees, with a declaration of research ethics and safety approved by the university. With their agreement, by filling out an informed consent form, the interviews were recorded and later transcribed. Each interview lasted around one hour, and each interviewee was given a gift card for their participation. We first asked the interviewees their background information and experiences, then their project experiences, typical design workflows, common prototyping approaches and toolkits, challenges and difficulties of current prototyping workflows, and finally, their expected improvements to support the rapid prototyping of spatial interaction design for smart objects.

3.3 Data Analysis

We analyzed the interview transcripts using an open coding method [11]. Two authors first coded the transcripts individually and then discussed the individual coding results and made them into a codebook. All the authors then discussed the codes to find the emerging themes. We categorized the themes into three parts: motivations and methods (Section 3.4.1), difficulties (Section 3.4.2), and expectations (Section 3.4.3).

3.4 Findings

3.4.1 Motivations and Methods. Spatially-aware interactions of smart objects have been emerging in recent years. Although their design shares similarities with traditional interaction design, its prototyping process and focus have some differences from traditional ones. In fact, all of the interviewees reported that traditional interaction prototyping focused on validating both interactive flows and visual UIs, but the goal of prototyping spatially-aware interactions for smart objects was to rapidly demonstrate and test the interactive pattern and process. Through prototyping, they could understand the natural interactive flows and learn veritable actions and needs. P1 pointed out that he aimed to "*define the interactive pattern of smart objects and use the spatial relationships to design useful features*" and "*the prototyping focuses on demonstrating the input and output*". P4 emphasized that it was significant to learn "*real-world spatial interactions and the underlying needs*" in the prototyping phase.

Prototyping approaches to spatial interactions also differ from traditional methods. For example, the former focuses on validating objects' dynamic interactive behaviors instead of static states, and thus the sketch-based approaches are rarely utilized. The reported approaches can be divided into three categories:

Virtual demonstrations. To show dynamic spatial movements of devices, all of the interviewees reported their practice of creating dynamic demonstrations (e.g., making slides, videos, and animations, and setting up 3D virtual scenes) and adding animation or sound effects to simulate corresponding UI feedback.

Physical mockups. Three interviewees (i.e., P1, P2, P4) crafted physical boxes with real-world size and scale to simulate smart objects and allowed people to manipulate the boxes for spatial interactions. Combined with recording videos, the triggering feedback is added by post-processing. They (P1, P3, P4) also reported their practice of attaching sensors to the devices and crafting/programming interactive demos for testing.

Motion capture. Both P1 and P2 reported using motion capture (MoCap) systems to track the pose and movement of digital devices so as to simulate the real usage scenarios and interactions. They obtained the accurate spatial data (e.g., position, orientation, movement) of devices in real-time, but usually analyzed the data using programming offline.

From the reported approaches above, we noticed that these three approaches were with variant levels of fidelity. Designers typically made low-/middle-fidelity prototypes to show the ideas and interactive flows [17, 35]. For the interactive patterns, designers resorted to high-fidelity prototyping approaches, since they needed to obtain the spatial properties of intelligent objects, record and test the spatial interactions by directly manipulating the smart objects in 3D space. Three interviewees (P1-3) considered it necessary to use real-world interactive patterns for demonstration and testing in prototyping. For the interactive feedback, all the participants believed it was less important [35, 51] than the interactive patterns, and thus preferred to simply use visual and sound effects to simulate the feedback.

3.4.2 Difficulties. The interviewees raised several difficulties during their typical prototyping workflows:

Complex and various usage scenarios. All of the interviewees pointed out that the main challenge of prototyping objects' spatial interactions was the complex and various usage scenarios. Existing prototyping tools and approaches can be only applicable for a single fixed indoor scenario (P1), and thus it will cost a lot of time, labor, and money for testing and iterations (P2, P3). P4 said "*actual using cases are more complex than those in the prototyping stage, so we need to adjust the interactions during testing considering multiple using cases. But existing frameworks are not general and applicable for all the scenarios*". P2 thought it necessary to consider misoperation scenarios in prototyping. He especially proposed that "*there are a lot of scenarios to be tested for a natural manipulation of devices. In these scenarios, we need to observe how users perform and how the manipulation of objects would lead to misoperations, and then make adjustments on the prototypes immediately*". He was worried that the existing tools were not easy to be used in different scenes. P1 pointed out that it was fundamental to obtain and utilize spatial data of smart objects in prototyping and they usually used MoCap systems to get such data. But the fixed setup would restrict the prototyping to happen in certain indoor scenes.

Lack of ad-hoc all-in-one tools. The interviewees considered the lack of ad-hoc tools as a significant factor that hindered the current design procedure to be more efficient. P1 mentioned that "*traditional UIs contain complex interaction processes, while spatial interaction is more like a 'single-point' interaction*", so traditional prototyping approaches were not applicable for prototyping objects' spatial interactions. As P3 stated, "*there is no tool that supports the prototyping of spatial interactions among intelligent objects, especially for multiple or cross devices. When designing for multiple- or cross-devices- interactions, I need to write my own programs to negotiate the different spatial states of them. There are no general frameworks, so it is a troublesome problem*". From their viewpoints, designers need to deploy multiple tools to prototype certain features using individual methods, e.g., making animations and videos to show interactive flows (P4), making physical mockups to simulate real-world manipulations (P2), using MoCap systems to get spatial data/states of objects (P1), visualizing and analyzing spatial data by setting up virtual scenes (P3) or programming (P1). These tedious combinations are due to the lack of an all-in-one framework (P3), and thus reduce the prototyping ease and efficiency.

High technical difficulties. According to the background information of the interviewees (Table 2), all of them were capable of programming, and most of them had to code in their workflows. However, they pointed out that such implementations required extensive use of programming. P1 reported that designers needed to write codes to visualize and analyze spatial data of smart objects. He said "*the IDE of MoCap systems is hard to use, so it's hard to utilize the spatial data in real-time for interactive demos*". P4's comments resonated with P1's, "*traditional designers [without coding skills] usually design the interactions from an ideal perspective but need to know the real-world interaction and actions. They usually have to collaborate with technical mates to co-design their results*". P3 thought it tedious to specify the triggering spatial events and corresponding feedback among different devices by coding, as "*current methods don't support the specification of the interaction logic in an easy and clear way so we need to program for their communication*". From their experience, their teams usually recruited employees with technical backgrounds or suggested designers learn to program. These invisible technical requirements increase the entry barriers and design costs to a large degree.

3.4.3 Expectations. Combined with the above-mentioned challenges, the interviewees stated their expectations and standards for an ideal solution to tackle their current difficulties. Here we list their expectations as follows, separated into common and single points:

Common points:

- **C1:** Applicable and general to various using scenarios (All).
- **C2:** Easily and rapidly express design ideas, describe and demonstrate interaction scenarios and flows, and define interactive patterns (input) and feedback (output) in an all-in-one workflow (P1, P2, P3).
- **C3:** No need to write codes (P1, P3, P4).
- **C4:** Design and test the real-world interactive patterns as inputs (P2, P3, P4).
- **C5:** Specify simulated or other simple representations and feedback as outputs (P1, P2).
- **C6:** Obtain, utilize, and analyze spatial and movement data of smart objects rapidly, conveniently, and in real-time (P1, P2).

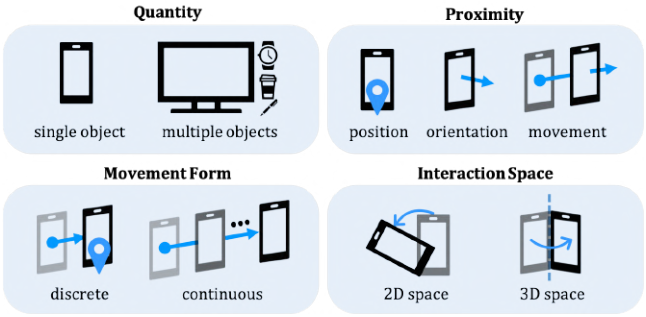


Figure 2: The design space of spatially-aware object interactions is formulated by four dimensions.

- **C7:** Specify and trigger the events among different objects (P1, P3).

Single points:

- **S1:** Applicable for various objects (P2).
- **S2:** No need to set up interaction scenarios (P3).

4 SPATIAL INTERACTION DESIGN

To help better understand spatially-aware interaction prototyping, we first explore a design space of such interactions from four dimensions in Section 4.1. Based on the interview findings and the design space, we propose an input-out interaction model that formulates the different types interactions in Section 4.2.

4.1 Interaction Design Space

Existing literature has explored interaction design spaces such as cross-device interactions [6], proxemic interactions [4], and AR freehand interactions [48]. We discuss the design space of spatially-aware interactions of smart objects following the existing works. Here we summarize the space from four dimensions (Figure 2).

Dimension 1: Quantity. Spatial interactions of smart objects can be classified according to device quantity, i.e., *single-object* or *multiple-object* interaction. The interaction of a single object utilizes the spatial pose or movement of the object itself. The interactions among multiple objects describe their spatial relationships and configurations.

Dimension 2: Proximity. Ballendat et al. [4] discussed the spatial relationships happening among entities (people, devices, and objects) from the measurements of *position*, *orientation*, *distance*, and *movement*. Such proxemic dimensions are still applicable for spatially-aware interactions of one or multiple smart object(s).

Dimension 3: Movement Form. This dimension differentiates spatial interactions: *discrete movement*, i.e., happening once during a period, and *continuous movement*, i.e., continuously changing during a period.

Dimension 4: Interaction Space. Spatial interactions of objects can vary across 2D or 3D dimensions. In the *2D dimension*, objects move in a 2D plane, and thus it is a 3-DoF (degree of freedom) interaction. In the *3D dimension*, objects move and/or rotate in the 3D space so it is a 6-DoF interaction. The movements can also be divided further into moving along an edge, on a plane/surface, and in mid-air.

4.2 Spatial Interaction Model

As pointed out by the interview participants, they expected to define the interactive patterns (input) and feedback (output) easily in the prototyping stage (C2). This expectation resonates with the discussions in GesturAR [51], which explored the input-out model [2, 18, 29, 34] for AR freehand interaction. Following GesturAR [51] and the design space discussed above, we present an input-out model to describe the spatial interactions of smart objects. The *input* is a spatial event (e.g., spatial movement) of a real-world object and the *output* is a certain virtual effect that responds to the spatial event (C4, C5). We divide the *input* into *discrete event* and *continuous event* of smart objects, and the *output* into *discrete effect* and *continuous effect* of virtual assets. Such a paired input event and an output effect compose a *discrete* or *continuous* input-out interaction. Continuous input-out interactions can be further categorized into *synchronous* and *tween* interactions. The former means that the spatial status of the virtual effect is synchronously changed with the spatial status of the smart object, and the latter means that the spatial status of the virtual effect is continuously changed according to the starting and ending status of the object. Each type of input triggers the corresponding type of output (Table 3). Figure 3 shows the examples of each event type and the corresponding effect for a single object and multiple objects (C7).

4.2.1 Single object. For a single smart object, the spatial events (input) and the triggering effects (output) can be categorized from the dimensions of *Movement Form* and *Proximity* with multiple variations, as summarized in Table 3.

Discrete interaction. For discrete position events, the object can be moved to a position range (with range = 0 meaning it comes to a certain position) (Figure 3 (a)) or change its position (Figure 3 (b)). For discrete orientation events, the object can face an orientation range (with range = 0 meaning it faces a certain direction) (Figure 3 (c)) or change its orientation (Figure 3 (d)). These discrete events are mapped to discrete effects like *Appear*, *Disappear*, and *Shake*, which can simulate the real-world effects easily for prototyping.

Synchronous interaction. Synchronous effects are tightly related to synchronous events from the proximity measurement of position and orientation. For synchronous position or orientation events (Figure 3 (e)(f)), the 2D/3D movement of the virtual content responds to the 3D movement of the object synchronously.

Tween interaction. Similar to animation keyframing, tween interaction describes a continuous interaction by specifying two key statuses (i.e., starting status and ending status) and interpolating the intermediate status according to the specified key statuses (Figure 3 (g)(h)). The two variant positions or orientations of the object can be set as the starting and ending *event tweens*. The two variant effects (e.g., positions, orientations, opacities, and scales) of the virtual contents can be set as starting and ending *effect tweens*. Each pair of event tweens can be mapped to each pair of effect tweens.

4.2.2 Multiple objects. Since the spatially-aware interactions happening among multiple objects are more complex, most of the

Table 3: Taxonomy of input events for a single object and multiple objects as well as the corresponding output effects. Please find the corresponding examples of the events (a)-(l) in Figure 3.

	Single Object			Multiple Objects
	Discrete	Continuous - Synchronous	Continuous - Tween	Discrete
Event	(a) Position range (b) Position change (c) Orientation range (d) Orientation change	(e) Position (f) Orientation	(g) Two key positions (h) Two key orientations	(i) Relative position in box zones (j) Relative position in fan zones (k) Relative identical/opposite/vertical orientation (l) Distance
Effect	Appear Disappear Shake	Synchronous 2D/3D position Synchronous 2D/3D orientation	Tweening position Tweening orientation Tweening opacity Tweening scale	Appear Disappear Shake

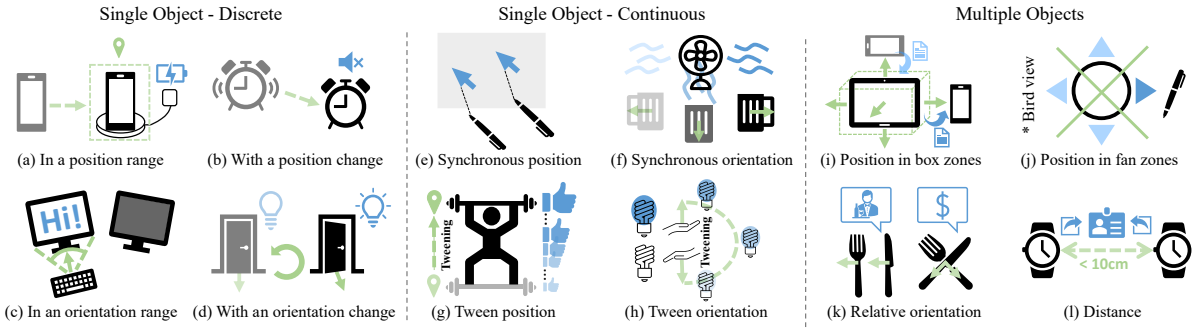


Figure 3: Examples of each type of spatial event and triggered effect of the spatial interaction model. The objects being interacted are in black, the events in green, and the effects in blue.

current applications are based on discrete interactions. So here we classify the spatial events (input) and the triggering effects (output) can be classified from the dimension of relative *Proximity*, as summarized in Table 3.

Relative position. Objects can be manipulated by changing their relative positions. For example, the 3D space around an object can be roughly divided into six zones: front, back, above, below, left, and right (Figure 3 (i)). Other object(s) can be placed in one of the zones to trigger a discrete effect. The 2D space around an object can be roughly divided into four fan zones: left-front, left-back, right-front, and right-back (Figure 3 (j)).

Relative orientation. Objects can be manipulated by changing their relative orientations (Figure 3 (k)). For example, an object can be in identical, opposite, and vertical directions from another object(s).

Distance. Objects can be placed in the 3D space at, less than, or larger than a certain distance (Figure 3 (l)).

Combinations. Besides the interactions of multiple objects discussed above, a combination of multiple events of single objects can also be considered as the interactions among multiple objects. For example, when a chair is dragged from a desk, and a mouse on the table is moved with a position change, the computer will power off.

5 PROOBJAR SYSTEM DESIGN

To support prototyping spatial object interactions in the design space discussed above, we design and develop a prototyping system *ProObjAR*, which provides a single user with an AR-HMD-based interface, implemented on a Microsoft HoloLens 2. Users can follow an input-out triggering workflow to define the input events, output effects, and their relationships from a visual programming interface (C3) for various usage scenarios (C1), and test the results through real-world object manipulation in situ, as shown in Figure 4. We will elaborate on the workflow and the prototyping interface in detail below.

5.1 Input-output Triggering Workflow

We design an input-output triggering workflow based on the proposed spatial interaction model in Section 4.2. In this workflow, users prototype a spatial interaction by specifying an input spatial event and the corresponding output virtual content, and creating the triggering connection between the input and the output. For a single object, any discrete input can be mapped to any discrete output, which leads to 12 (= 4×3) types of interactions. Any synchronous input can also be paired with any synchronous output except two less meaningful/useful pairs (i.e., synchronous 3D position with synchronous 3D orientation, and synchronous 3D orientation with synchronous 3D position), thus leading to 6 types of synchronous interactions. Any tween input can be paired with any tween output, resulting in 8 (= 4×2) types of tween interactions. For multiple objects, the six types of input can be mapped to any discrete input, so it has 18 (= 6×3) types of interactions. We also provide users with the logical operators (i.e., *and*, *or*, *not* operators) to help specify and combine the multiple interactions.

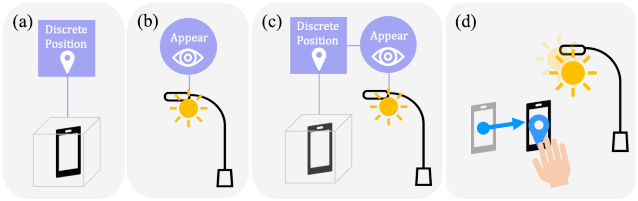


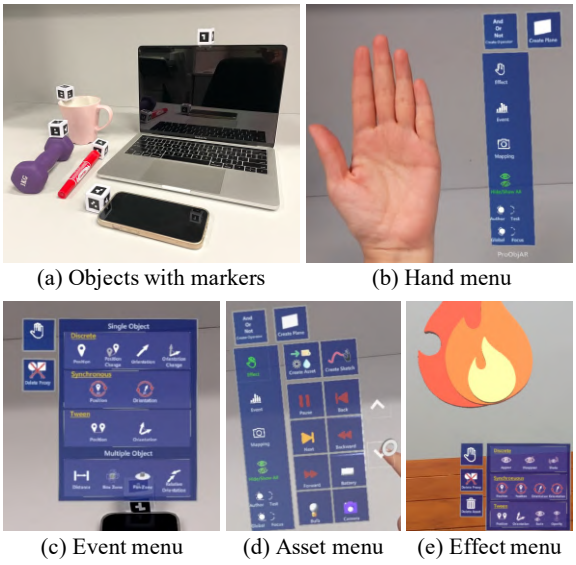
Figure 4: The workflow of *ProObjAR*. (a) The user creates an event (e.g., discrete position in this example) visualized by a square proxy above a real-world object (e.g., a mobile phone). (b) The user creates an asset (e.g., a sun symbol) with a specified effect (e.g., appear) represented by a circular proxy above the asset. (c) The user drags a line to connect the event and effect proxies to specify the triggering connection. (d) The user tests the result by manipulating the object to perform the specified event and thus trigger the effect.

5.2 The Prototyping Interface

ProObjAR interface allows designers to rapidly prototype the spatially-aware interactions from an AR-HMD in an all-in-one input-output triggering workflow.

5.2.1 Obtain spatial information. Various approaches have been proposed to detect the spatial information for scene understanding, for example by using markerless deep-learning-based approaches [40, 49] or marker-based approaches [14]. Due to the limited computing capacities of HoloLens 2, it is hard to get robust tracking of any types of objects that users need using the first type of method, especially with possibly serious hand occlusion. Thus, to achieve instant interaction feedback and easy implementation at low cost (C6), we employ two methods for scene understanding: for the static, fixed, and large planes, we allow users to use an automatically detected plane or manually create a plane, and move and rotate it to a desired pose. This plane can be specified as the interaction space of virtual contents. For moving objects, we utilize an AR-marker-based tracking method [14] to get the 6-DoF pose of an object being tracked. Before prototyping, we prepare enough cube-shape fiducial markers and allow users to attach them to the object surfaces (Figure 5 (a)), and our interface will get the pose data in real-time.

5.2.2 Create input events. We provide a hand menu (Figure 5 (b)) as an overall menu when the user palms up. It contains an event button, an effect button, a mapping button, and several mode switching buttons. The user can start with the creation of either input events or output effects. However, to follow an input-output workflow, here we explain the interface by creating the input events first. Once the user presses the event button near an object, the event menu (Figure 5 (c)) becomes active and appears near the object. Then the user can create the input events from the event menu showing 8 events (i.e., discrete position, position change, discrete orientation, orientation change; synchronous position, synchronous orientation; tween position, tween orientation) for the single object and 4 events (i.e., distance, relative box zone, relative fan zone, relative orientation) for multiple objects. Once the user successfully creates an event, a square proxy with the connected line will be added above the object (Figure 6). Below we elaborate on the steps for each event type.



(a) Objects with markers (b) Hand menu

(c) Event menu (d) Asset menu (e) Effect menu

Figure 5: Object tracking and main menus of *ProObjAR*. (a) We attach cube AR markers on the objects to be interacted for object tracking. (b) The hand menu consists of an event button, an effect button, a mapping button, and several mode-switching buttons. (c) Associated with a selected object, the event menu consists of four-row buttons for defining 12 types of spatial events. (d) The asset menu involves an asset repository and a sketching button for creating freehand sketches. (e) Associated with a selected asset, the effect menu provides users with 11 types of virtual effects.

Certain position or position range. The user moves the object of interest to a desired triggering location and presses the position button. Then a bounding box will be shown at the center of the object, with a square proxy connecting with the object through a line to represent the created event (Figure 6 (a)). The user can adjust the size of the bounding box using a pinch gesture to change the position range.

Certain orientation or orientation range. The user manipulates the object to face a certain direction and presses the orientation button. Then an arrow will appear from the object center with the facing direction the same as the object's (Figure 6 (b)). The user can also specify an orientation range by rotating the object in another direction and press the orientation button again to create another arrow. The orientation range is formed from the smaller angle between the two arrows.

Position or orientation change. By pressing the position change or orientation change button, a spatial change event can be created, visualized by a square proxy above the object. It means when the object's position or orientation is changed, the event will happen.

Synchronous 3D position/orientation. The user can create an event that uses the 3D position or 3D orientation of the object

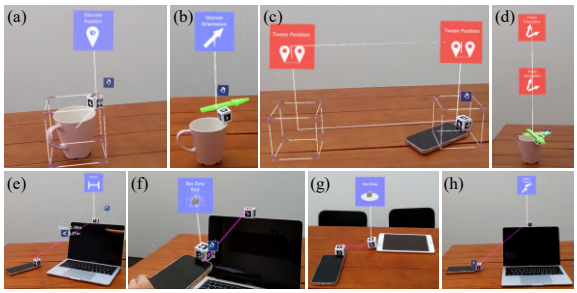


Figure 6: Examples of user-specified input events for a single object (a-d) and multiple objects (e-h). (a) Discrete position proxy shown as a blue square connected with a bounding box indicating the defined object position event. (b) Discrete orientation proxy with an arrow indicating the defined object orientation event. (c) Tween position proxies with two bounding boxes indicating the defined starting and ending positions. (d) Tween orientation proxies with two arrows indicating the defined starting and ending orientations. (e) Distance event is defined with a line connecting two objects, with a specified distance value, a symbol, and the current distance value shown on the line. (f) Relative box zone event is defined by placing an object in one of the six zones around a target object (i.e., the laptop is at the back of the phone in this example). (g) Relative fan zone event is defined by placing an object in one of the four zones around a target object. (h) Relative orientation event is defined by placing two objects in an identical, opposite, or vertical direction.

as a trigger to drive the virtual content's spatial movement in a synchronous manner.

Tween position/orientation. The user can add tween events by specifying the starting and ending spatial states (i.e., *event tweens*) of the objects. For tween position events, the user moves the object to a position and presses the button, and a bounding box is added there. Then the user repeats this operation in another position (Figure 6 (c)). When the object is placed in any intermediate position among these two positions, it will trigger such an event. The tween orientation event can be similarly created (Figure 6 (d)).

Distance. Creating a distance event requires the user to first move a *subject* object with a desired distance to a *target* object. Then when pressing the distance button, the user can use a pinch gesture to drag a line from the *subject* object to the *target* object, with the distance value showing on the line and a button for switching between '<' and '>' symbols (Figure 6 (e)).

Relative position. The representative relative position events can be divided into box and fan zone events. The user can create the *box zone* event (Figure 6 (f)) by moving a *subject* object to one of the six box zones (i.e., front, back, above, below, left, and right) around a *target* object, and connect a line between the two objects. A *fan zone* event (Figure 6 (g)) can be specified by moving a *subject* object to one of the four fan zones (i.e., left front, left-back, right-front, and right-back) around a *target* object when the two objects are on the same plane.

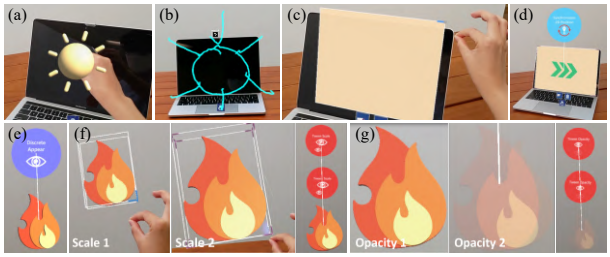


Figure 7: (a) Create an asset by dragging the pre-defined asset from the asset repository. (b) Create an asset by 3D sketching using a fingertip. (c)-(d) Create a plane interactively to work as the active moving area of a virtual asset. (e) Discrete or synchronous effect is added to the asset visualized by a circular proxy above the asset. (f) Tween scale effect is specified by defining two asset scales using a pinch gesture, indicated by two circular proxies above the asset. (g) Tween opacity effect is specified by defining two opacity values from a slider, visualized by two circular proxies above the asset.

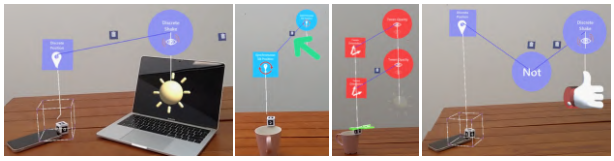
Relative orientation. The user can create the relative orientation event (Figure 6 (h)) by adjusting the orientations of two objects to be identical, opposite, or vertical.

Logic operators. Our system allows designers to create both individual and compound spatial events. We provide users with three logic operators (i.e., *and*, *or*, *not* operators) for quickly creating logical events. Users can add these operators represented by the proxy to the AR scene (Figure 8 (c)).

5.2.3 Create output effects. After creating the input event, the user can create the corresponding output effect. We allow designers to create virtual assets by dragging the pre-built assets from the menu (Figure 7 (a)) or drawing sketches using a fingertip (Figure 7 (b)). Both visual and sound assets are supported, and they are represented by a 2D icon, a 3D model, or a free-hand sketch in the repository. Once the asset is added to the scene, the user can move it, rotate it, and change its scale using a pinch gesture. The user can also attach the asset to an automatically detected plane or surface using ray casting. Besides, the user can interactively create a plane, place it in a certain location and with a certain orientation, and specify it as the active moving area of the virtual asset (Figure 7 (c)(d)). Then the designer can specify the animation effect to the created asset, visualized by a circular proxy with the connected line (Figure 7). Our current system supports 12 types of effects as follows.

Appear, Disappear, Shake. These three discrete effects can be added to the asset, visualized by a circular proxy (Figure 7 (e)). For the sound asset, the "sound playing" effect is associated with the "Appear" and "Shake" effects, and the "stop playing" is linked with the "Disappear" effect.

Synchronous position/orientation. The user selects this effect to make the 2D/3D position/orientation of the virtual asset synchronously changed with the object. To create a synchronous 2D position/orientation event, the designer must select an asset that is already attached to a spatial plane (e.g., the 2D arrow attached to the plane in Figure 7 (d)).



(a) (b) (c) (d)

Figure 8: (a) Map discrete event and effect. (b) Map synchronous event and effect. (c) Map tween event and effect. (d) Map event and effect involving logical operators.

Tween position/orientation/scale/opacity. The user can add tween effects by specifying the starting and ending spatial states (i.e., *effect tweens*) of the assets. For the tween position effect, the user moves the asset to two separate positions. For the tween orientation effect, the user rotates the asset to face two separate directions. For the tween scale effect (Figure 7 (f)), once the user presses the scale button for the first time, a bounding box will be shown at the asset. Then the user can specify two separate asset scales by adjusting the size of the bounding box using a pinch gesture. For the tween opacity effect (Figure 7 (g)), an opacity slider will be displayed below the asset when the user presses the opacity button. Then the user can specify two asset opacity values by adjusting the slider.

5.2.4 Create triggering mappings. To complete an input-output workflow, we allow the user to create the triggering mapping between the input event and the output effect. The input event and the output effect are visualized with the corresponding event and effect proxies in the AR scene. The user can use the pinch gesture to drag a line from the event proxy to the effect proxy for discrete and synchronous events and effects (Figure 8 (a)(b)), and drag two lines to connect the corresponding event and effect proxies for tween events and (Figure 8 (c)). For the events involving logic operation(s), the user connects the event proxy first to the logic operator(s) and then to the target effect proxy (Figure 8 (d)). In the authoring stage, we provide a focus mode that allows users to select to display the related representations of one or multiple interested object(s) while hiding all the lines. When the user pinches an object, the lines related to it will appear and when pinching it again the lines will be hidden.

5.2.5 Test results. After the user authors all the input events, output effects, and their triggering relationships, the user can enter the testing mode. When the user manipulates an object to perform the specified spatial interaction, the connected effect will be triggered. The connection lines between the event proxies, operators, and effect proxies will turn green to indicate the event is successfully triggered, otherwise the lines stay in red. All of the proxies and lines will be hidden when the user presses the "Hide All" button. To avoid the user missing the triggered animation effect, we provide a replay button near the asset for replaying the effect. The user can watch and test the results multiple times and from any view.

6 PRELIMINARY USABILITY STUDY

To evaluate the usefulness, efficiency, and expressiveness of *ProObjAR*, we conducted a preliminary usability study by asking participants to create free-form prototyping results with our system, after getting the approval of the research ethics and safety from the university. We recruited 8 participants (3 males and 5 females, aged 21 to 33, U1-U8) from the university email portal and personal networks. All of them had professional design experiences except U8. Three of them (U1-U3) were postgraduate students majoring in interaction design and with industrial project experiences in designing spatial interactions. Three of them (U4-U6) were Ph.D. students in HCI with both industrial project and research experience in designing spatial interactions of digital devices and household objects. U7 was a Ph.D. student in industrial design with UI/UX design experience. U8 was a casual user without any design skills or experience. They had diverse levels of self-assessed AR experiences as follows: had used AR applications (no: U7, U8; medium: U3, U5, U6; high: U1, U2, U4), had designed AR UIs/applications (no: U3, U5, U7, U8; medium: U6; high: U1, U2, U4), had developed AR applications (no: U3, U5-8; medium: U1; high: U2, U4). None of them had participated in our empirical study (Section 3) or experienced our system before the study. The participants were asked to fill in a consent form, including the study description, purpose, potential risks and benefits, compensation, confidentiality, participation and withdrawal, and questions and concerns before the study. To help participants better understand our system, we conducted the study through two sessions: a training and initial prototyping session and a formal prototyping session on separate days with a 1-2 day(s) interval. We set up this arrangement to reduce the influence on the participants' creativity from the training session. In this case, the participants had more time to think about the design ideas beyond the training examples. After they finished two sessions, they were asked to fill in a questionnaire on System Usability Scale (SUS) on a 5-point scale (1 = strongly disagree to 5 = strongly agree). Besides, we discussed with them their user experience of *ProObjAR* through a small-scale interview. We encouraged them to perform as they wanted and told them that all the comments/scores, no matter whether positive or negative, were welcome.

6.1 Session 1: Training and Initial Prototyping

In this session, we first introduced the study goal, study task, system UI, and system workflow to the participants. We also gave them several result videos as examples. They started to learn the system by reproducing these examples or using *ProObjAR* freely. We guided them on stand-by, i.e., they were allowed to explore the system and workflow, but when they were unclear, we described and told them the next-step interaction (e.g., perform a certain hand gesture to scale an asset). Once familiar with the system, they could further create their desired results freely. The whole session for each participant took about 20 minutes and happened in an indoor scene. All the participants reported that they mastered the system after reproducing/producing 1-2 example(s), and thus learned how to use the system quickly. This was also reflected by the scores of Q7 in SUS (Figure 10 (a)). But for Q4 (i.e., "use the system without the support of a technical person") and Q10 (i.e., "use the system without learning anything new"), the participants had diverse ratings. The

participants without any AR experience (U7, U8) considered the introduction tutorials from a technical person very necessary and asked for more help, compared to those with AR experience. The former group was not familiar with the AR space, so they needed more guidance to perform interactions in such a space (e.g., using a hand gesture to click a button in a 3D UI).

6.2 Session 2: Formal Prototyping

In this session, the participants have thought about their target prototyping results in advance. They led us to the scenes that they wanted to design, and we guided them to stick the fiducial markers to the objects of their interest. Then they started prototyped using *ProObjAR* freely in their desired scenes (Figure 10 (b)). In this session, one of the authors still stood by in case the participants needed any help.

Result and discussion. All of the participants created 21 indoor and outdoor prototyping results in total (Figure 9). The results cover various usage scenarios including facility/appliance/device function controlling (Figure 9 (c)(d)(e)(h)(j)(l)(m)(n)), cross-device information exchange and synchronization (Figure 9 (a)(b)), information discovery, display and presentation (Figure 9 (g)(k)(p)), and personal health and living management (Figure 9 (f)(o)), and activity planning (Figure 9 (i)). We found that most of the prototyping usage scenarios are for facility/appliance/device function controlling using objects' dynamic poses. They expected smart living ecosystems to facilitate living convenience and working efficiency. They turned the objects around them into smart controllers, especially non-digital everyday objects (e.g., cups, wallets, pens). The results of cross-device information exchange and synchronization usually take the advantage of object proximity to define the interaction space. The results of information discovery and presentation utilize the objects' spatial movements to excavate extra space for displaying information beyond the existing reality. The participants prototyped spatial interactions of single objects and among several objects, covering diverse individual and compound events and effects (Table 4). It validates the expressiveness and applicability of *ProObjAR*.

For the prototyping time, each participants spent 2-8 minutes authoring and 2-7 minutes testing the each result (Figure 9). We obtained the following observations: (i) less than 5 minutes were spent authoring a single object's discrete interactions; (ii) about 4-6 minutes were for authoring the interactions between/among 2-3 objects; (iii) synchronous and tween interaction authoring took 4-5 and 5-8 minutes, respectively; (iv) the participants spent more time (5-7 minutes) testing the multi-object and synchronous/tween interactions since they needed to control the movements of multiple objects or a single object continuously to see the effects in the middle interval. From these observations, we found that users can create a prototype of common varying complexity within or around 10 minutes using *ProObjAR* .

The SUS scores rated by the participants were overall good, and Figure 10 (a) shows the rating distribution of every SUS question. The scores show that the participants quickly learned the system from the training session (Q7) and used the system easily (Q3). In the formal session, the participants rarely raised questions about the overall workflow, but some of them tried to confirm their interaction with us from a think-aloud protocol when creating the interactions that were not covered in the training session. The participants reported that the training session and tutorials were necessary (Q4&Q10) to help them learn the system.

Most participants found *ProObjAR* worked better and could facilitate the prototyping of spatially-aware object interactions, especially compared with their commonly used prototyping approaches (e.g., video demonstration and physical mockup). For example, U4 commented *"the system is truly useful because it is designed for spatial object interaction tasks. It is more efficient than making animations by myself"*. The authoring and testing modes were both well received by the participants. *"I like the author-and-test feature since I can instantly watch my prototyping results to get the feedback in time,"* as commented by U3. The participants liked the system generality in various scenarios. *"I like the system because I can design the ideas anywhere, no matter the indoor office and living scenes or outdoor public scenes* (Figure 9 (j))" (U1). This resonated with U4's comment: *"I can test the interactions in multiple usage scenarios."*

Most participants appreciated the input-output event triggering workflow and UI, and thought them convenient and intuitive to define the interaction patterns (Q8). U1 said, *"The interaction logic is very clear, including an input, an output, and their mapping, so it is simple to understand"* (Q3&Q8). U2 also considered the system intuitive since *"I can directly drag and place the blocks and connect them together."* Besides, the participants with experience in event-driven visual programming (U4, U5, U6) especially thought our workflow *"easy to follow" (U6)*, *"commonly used by designers" (U4)*, and *"can be learned quickly" (U5)*. They expected more designers to use *ProObjAR* frequently in their current work (Q1). In addition, some participants (U1, U4, U6, U8) pointed out that our system supported comprehensive event and effect types, which were well integrated (Q5&Q6). *"The options are comprehensive, considering both discrete and continuous interaction states. So it can cover most of the daily operations of a single object or multiple objects"(U5).* The participants also appreciated the feature of prototyping real-world object interaction. *"Testing the real-world object interactions is very necessary, which is usually ignored in the prototyping stage. This system allows me to manipulate the object in the physical space,"* commented U3. U4 also said, *"physical manipulations can help me get veritable feedback."* The casual user U8 especially appreciated to control the real object since *"familiar objects can be used to plan surprising effects" (Figure 9 (i))*. Most of the participants felt confident in using the system (Q9) and satisfied with their prototyping results.

Besides the positive feedback, we also received comments from the participants about improving our system. The participants rated diversely on "the system is simple (Q2)". Those with neutral or negative attitudes mainly thought the system supported a lot of spatial events and effects and thus they needed to learn and remember the functions of multiple menus and buttons. Besides, when creating complex events, our system requires the specification of events one by one. U7 expected a synchronous event with both position and orientation to be created at once. But in our current design, he needs to specify a synchronous position event and an orientation event successively, involving repetitive actions. We will simplify the system by re-designing the event categorization and providing

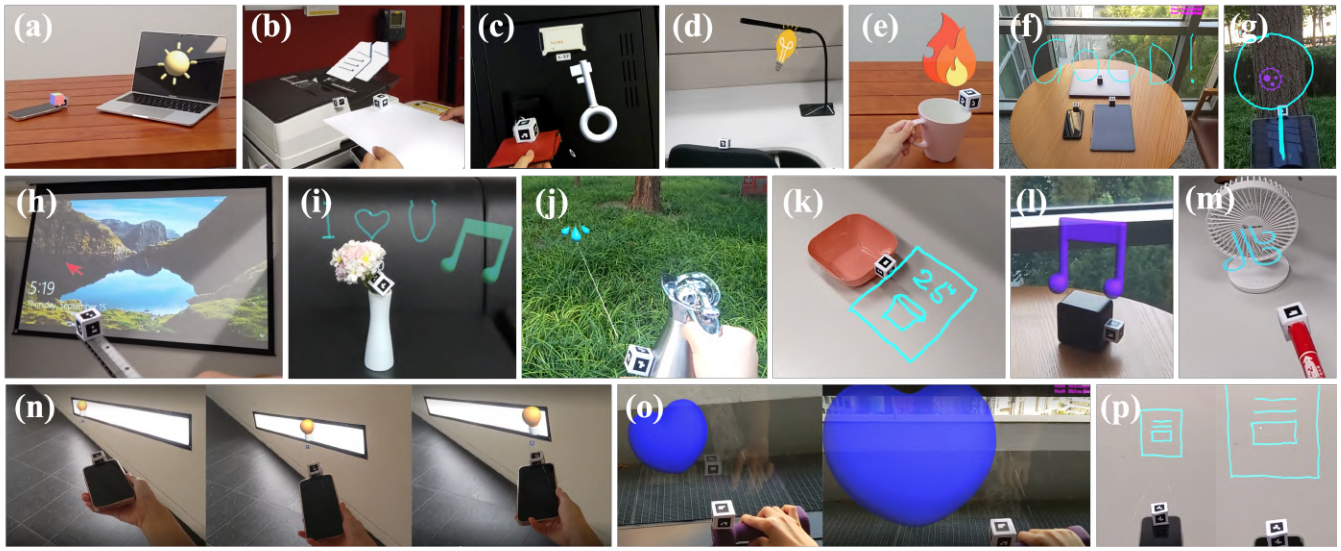


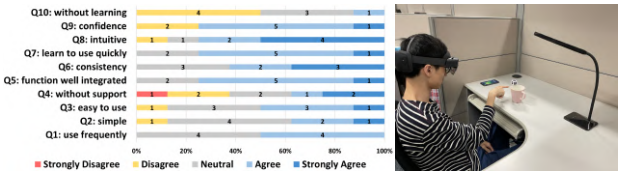
Figure 9: A gallery of representative results from the user study. The study participants designed the following applications: (a) A cross-device connection application: When the phone is near the laptop, the laptop screen turns on. (b) A future-vision intelligent printer: When the user holds a piece of paper facing the printer, the content on the paper will be copied with no need to put the paper in the printer. (c) A smart locker: When the user moves the wallet approaching and in front of the locker, it will be unlocked. (d) A smart workstation: When the chair position changes, the light will be brightened. (e) An auto-heating table: The user moves the cup to a certain location and the cup will be heated. (f) A smart desktop: the desktop will check the cleanliness of objects placed on it. If all the objects are placed facing the same direction, it will display a "GOOD" sign on it. (g) A plant-checking app: The user moves the tablet to check the health of a tree and find a virus-clustered part. (h) A cursor-controlling ruler: The user uses the ruler as a pointer to project the cursor on the projection screen. (i) A smart gift vase: When the user places a flower into the vase, it will play music and project a text message "I Love U" on the background. (j) A smart watering facility: The user moves the watering pot to determine the watering area. (k) A smart desktop: When the tableware is put on the table, a digital menu will appear on the table and follow the tableware. (l) A smart speaker: The Bluetooth speaker faces different directions to play different styles of music. (m) A smart pen: The user uses the pen to control the fan direction. (n) A light-controlling app: The user moves the phone as a slider to adjust the light position in a light tape. (o) A smart fitness room: The user lifts the dumbbell and a heart is shown on the wall, and the heart scale indicates how much power has been used. (p) A projection-controlling phone: The size of the projected content is changed according to the distance between the phone and the wall.

Table 4: Descriptions of user-created prototyping results in Figure 9. The left and right numbers in the "Time" column refer to the authoring time and the testing time for each result, respectively.

No./ Object	Event –>Effect	Time	No. / Object	Event –>Effect	Time
(a) phone, laptop	distance–>shake	4′+3′	(i) flower	position–>appear, play	4′+4′
(b) paper, printer	relative orientation–>appear	4′+2′	(j) watering pot	synchronous 3D position–>2D position	5′+6′
(c) wallet, locker	distance+box zone–>appear	4′+3′	(k) tableware	synchronous 3D position–>2D position	4′+3′
(d) chair	position change–>appear	2′+2′	(l) bluetooth speaker	multiple orientations–>appear (play)	6′+7′
(e) cup	position–>shake	3′+2′	(m) pen	tween 3D orientation–>orientation	7′+5′
(f) phone, tablet, laptop	compound positions –>appear	6′+4′	(n) phone	tween 3D orientation->position	8′+6′
(g) tablet	synchronous 3D position–>3D position	4′+4′	(o) dumbbell	tween 3D position–>scale	5′+3′
(h) pen	synchronous 3D orientation–>2D position	4′+5′	(p) phone	tween 3D position–>scale	6′+4′

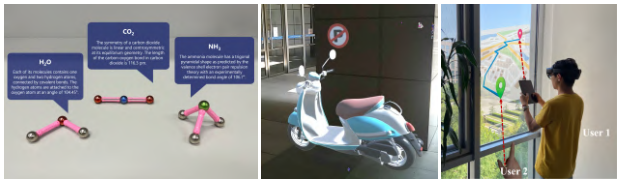
event copy-and-past interactions to reduce such repetitive user actions. U2 found that object tracking was not very robust when the object was attached to reflective surfaces (e.g., glossy screens). He suggested we might adjust the marker size or attach multiple markers to the reflective surfaces to achieve better tracking performance. This also encourages us to design color-pattern markers for more robust tracking on reflective surfaces. U6 suggested including more virtual effects, such as *"non-uniform scaling and freeform translation and rotation."* She said these effects would help demonstrate exaggerated effects [30].

From the prototyping results, we can see that they are object-oriented and are very different from hand-gesture-oriented authoring results in GesturAR [48] and full-body-proximity-oriented prototyping results in ProGesAR [51]. Although objects are manipulated by users' hands, the interactions of interest happen between



(a) (b)

Figure 10: (a) SUS score distribution. The question description here is the key points from the full SUS questions. (b) User prototyping process.



(a) (b) (c)

Figure 11: The potential using scenarios and improvements of *ProObjAR*. (a) The user can manipulate physical balls and sticks to compose different spatial structures to demonstrate the chemical structures of different substances. (b) We will include virtual objects for prototyping in the future. For example, a smart wall can be designed that reminds "no parking" when a motorbike stops here. The designer can control the position of a virtual motorbike model to define events and trigger the effects. (c) An envision for supporting multi-user collaborative prototyping: multiple users are in a shared AR space, and one user watches another user triggering an event by moving the tablet near to the wall and projecting the screen content (e.g., a map) to the wall. They can also manipulate the virtual contents from individual AR views.

objects. Compared to Jetter et al.'s work [27], our results contain both the physical interactions and the virtual effects, thus helping users obtain a faithful usage experience by considering the design context, while [27] can provide more freedom and simulation opportunities in the virtual scenarios.

7 CONCLUSION, LIMITATIONS, AND FUTURE WORK

In this paper, we presented *ProObjAR*, an AR-HMD-based system for prototyping spatially-aware interactions of smart objects. We first figured out the existing issues of the current design workflows from an interview study. From this study, we found that the traditional prototyping approaches often suffered from the challenges of **generality to complex and various usage scenarios**, **lack of ad-hoc all-in-one tools**, and **high technical difficulties**. The interviewees expected a prototyping solution to tackle these challenges by easily allowing them to define and demonstrate the interactive patterns (input spatial events and output simulated effects) in a coding-free manner. To help understand the spatial object prototyping, we then explored a design space from the dimensions of *Quantity*, *Proximity*, *Movement Form*, and *Interaction Space*. From the interview results and the design space, we discussed an input-output interaction model that formulates the taxonomies of spatial interactions. Based on the findings above, we designed and developed the system *ProObjAR*, which allows users to prototype the spatially-aware object interactions from an input-output triggering workflow. Users can specify the input event through real-world object manipulations, output effects using both pre-defined virtual assets and freehand 3D sketches, and their relationships from a visual programming interface. Users can watch and test the prototyping results in AR scenes easily. We validated the usefulness and expressiveness of *ProObjAR* through a two-session usability study.

There are still some limitations in our system. We can improve our system from the following aspects.

Limited event types. Although our system currently supports most types of spatial events in the discussed design space (Section 4.1), there are still some remaining event types that are not involved in *ProObjAR*. In particular, in the cases of relative position and relative orientation events for multiple objects, we have defined only a few common relationships and have not taken more fine-grained conditional customization into account, e.g., user-defined zone areas and certain relative facing angles. We will explore potential interactions that enable customized event specifications such as allowing users to divide the 2D/3D zones by themselves using hand gestures (e.g., cake-cutting gestures). Freeform movement events can also be included by allowing users to perform object movements through demonstrations [48].

Simulated effects. The response effects we support are simple to simulate realistic effects, e.g., the appearance of a virtual asset represents a light brightening, since users expect to use easy and low-cost effects to represent target effects in the prototyping stage (C5 in Section 3.4.3). Due to the powerful simulation capacity and immersiveness of the AR-HMD platform, we can explore more realistic effects at a low cost. For example, we can provide users with a light source and allow them to adjust the color, luminance, and direction to simulate more realistic effects, instead of a 3D sun/bulb model with changing opacity.

Object tracking. We use AR markers to determine the position and orientation of real objects. During our experiments, we found that the recognition accuracy degrades significantly when makers are obscured or under insufficient lighting conditions. We envision more robust tracking algorithms to improve the performance of *ProObjAR*, providing users with a smoother experience. For example, users manipulate objects using their hands in our system, so the hand pose can be utilized to help track the object pose using vision-based approaches.

A small sample of study participants. We invited 8 participants in our usability study, which provided preliminary validations. It is a relatively low sample. So the prototyping results and task completion times might have individual differences, and the SUS score distribution cannot be used to conduct statistical analysis. To tackle these problems, we will recruit a larger sample of participants covering more diverse backgrounds. UX and UI designers from the industries will be invited to use *ProObjAR* to explore whether and how it will benefit their current design tasks. A retrospective study can be conducted with the design experts in Section 3 to verify whether their challenges can be solved using *ProObjAR*. Besides,

more naive and casual users can also be recruited to use the tool for different purposes in their daily lives. For example, users can customize their preferred device connection distances in various places. Teachers or students can specify different spatial combinations of ball-and-stick models to demonstrate or learn the chemical structures of different substances (Figure 11 (a)).

Real objects v.s. virtual objects. We focus on prototyping spatially-aware interactions of real objects (C4 in Section 3.4.3) in this work. Real-object prototyping can help get and test realistic using experiences but also suffer from the drawbacks of physical dependence. Large and heavy objects are difficult to be manipulated to specify and trigger spatial events by the designer himself/herself. So we would like to include virtual objects in our system, and allow users to place them into physical scenes for prototyping (Figure 11 (b)).

From a single user to collaborative prototyping. The current system is designed for a single user. As pointed out in the interview (Section 3), spatial interaction prototyping sometimes happens among multiple designers and/or developers. We envision an extension of *ProObjAR* for collaborative prototyping in a shared AR space (Figure 11 (c)). In that space, multiple users can wear AR-HMDs to author interactive spatial object behavior together, and watch the triggering effects from a third-person view.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments, and the interview and user study participants for their time. This work was supported by gifts from Adobe and grants from the City University of Hong Kong (Project No. 7005729, 9667234, 7005590, 9229094), the Centre for Applied Computing and Interactive Media (ACIM) of the School of Creative Media, CityU, National Natural Science Foundation of China through Projects 61932003, National Key R&D plan 2019YFC1521102, and Beijing Science and Technology Plan Project Z221100007722004.

REFERENCES

[1] Samsung Electronics America. 2022. *Use the lift to wake feature on Your Galaxy Phone Kernel Description.* Samsung Electronics America. https://www.samsung.com/us/support/answer/ANS00083345/

[2] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-action-circuits: Leveraging generative design to enable novices to design and build circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 331–342.

[3] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K Chilana. 2020. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.

[4] Till Ballendat, Nicolai Marquardt, and Saul Greenberg. 2010. Proxemic interaction: designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 121–130.

[5] Marat Boshernitsan and Michael Sean Downes. 2004. *Visual programming languages: A survey.* Citeseer.

[6] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. 2019. Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices. In *Proceedings of the 2019 chi conference on human factors in computing systems*. ACM, New York, NY, USA, 1–28.

[7] Margaret M Burnett and David W McIntyre. 1995. Visual programming. *COMPUTER-LOS ALAMITOS-* 28 (1995), 14–14.

[8] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. 2019. GhostAR: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 521–534.

[9] 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*. 1059–1070.

[10] Han Joo Chae, Youli Chang, Minji Kim, Gwanmo Park, and Jinwook Seo. 2020. ARphy: Managing Photo Collections Using Physical Objects in AR. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–7.

[11] Kathy Charmaz. 2008. Constructionism and the grounded theory method. *Handbook of constructionist research* 1, 1 (2008), 397–412.

[12] Luigi De Russis and Fulvio Corno. 2015. Homerules: A tangible end-user programming interface for smart homes. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 2109–2114.

[13] Anind K Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive prototyping of context-aware applications. In *International conference on pervasive computing*. Springer, 254–271.

[14] Mark Fiala. 2005. ARTag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. IEEE, 590–596.

[15] Rainhard Dieter Findling, Muhammad Muaaz, Daniel Hintze, and René Mayrhofer. 2016. Shakeunlock: Securely transfer authentication states between mobile devices. *IEEE Transactions on Mobile Computing* 16, 4 (2016), 1163–1175.

[16] Ramsundar Kalpagam Ganesan. 2017. *Mediating human-robot collaboration through mixed reality cues.* Ph. D. Dissertation. Arizona State University.

[17] Danilo Gasques, Janet G Johnson, Tommy Sharkey, and Nadir Weibel. 2019. What you sketch is what you get: Quick and easy augmented reality prototyping with pintar. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6.

[18] Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R Klemmer. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 145–154.

[19] Rex Hartson and Pardha S Pyla. 2012. *The UX Book: Process and guidelines for ensuring a quality user experience.* Elsevier.

[20] James Hollan, Edwin Hutchins, and David Kirsh. 2000. Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7, 2 (2000), 174–196.

[21] Steven Houben and Nicolai Marquardt. 2015. Watchconnect: A toolkit for prototyping smartwatch-centric cross-device applications. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. ACM, New York, NY, USA, 1247–1256.

[22] Zheng Huang and Jun Kong. 2020. A Toolkit for Prototyping Tabletop-Centric Cross-Device Interaction. *International Journal of Human–Computer Interaction* 36, 6 (2020), 536–552.

[23] Ke Huo, Yuanzhi Cao, Sang Ho Yoon, Zhuangying Xu, Guiming Chen, and Karthik Ramani. 2018. Scenariot: Spatially mapping smart things within augmented reality scenes. In *Proceedings of the 2018 CHI Conference on human factors in computing systems*. ACM, New York, NY, USA, 1–13.

[24] Maria Husmann. 2017. *Investigating Tool Support for Cross-Device Development.* Ph. D. Dissertation. ETH Zurich.

[25] Maria Husmann, Nina Heyder, and Moira C Norrie. 2016. Is a framework enough? Cross-device testing and debugging. In *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM, New York, NY, USA, 251–262.

[26] Maria Husmann, Michael Spiegel, Alfonso Murolo, and Moira C Norrie. 2016. UI testing cross-device applications. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*. ACM, New York, NY, USA, 179–188.

[27] Hans-Christian Jetter, Roman Rädle, Tiare Feuchtner, Christoph Anthes, Judith Friedl, and Clemens Nylandsted Klokmose. 2020. " in vr, everything is possible!": Sketching and simulating spatially-aware interactive spaces in virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–16.

[28] Haojian Jin, Christian Holz, and Kasper Hornbæk. 2015. Tracko: Ad-hoc mobile 3d tracking using bluetooth low energy and inaudible signals for cross-device interaction. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 147–156.

[29] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 395–405.

[30] Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. Motion amplifiers: sketching dynamic illustrations using the principles of 2D animation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 4599–4609.

[31] Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, and Daniel Fitton. 2009. Smart objects as building blocks for the internet of things. *IEEE Internet Computing* 14, 1 (2009), 44–51.

[32] Veronika Krauß, Alexander Boden, Leif Oppermann, and René Reiners. 2021. Current practices, challenges, and design implications for collaborative AR/VR application development. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[33] Christian Kray, Daniel Nesbitt, John Dawson, and Michael Rohs. 2010. User-defined gestures for connecting mobile phones, public displays, and tabletops. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*. ACM, New York, NY, USA, 239–248.

[34] David Ledo, Jo Vermeulen, Sheelagh Carpendale, Saul Greenberg, Lora Oehlberg, and Sebastian Boring. 2019. Astral: Prototyping Mobile and Smart Object Interactive Behaviours Using Familiar Applications. In *Proceedings of the 2019 on Designing Interactive Systems Conference*. 711–724.

[35] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–13.

[36] Nicolai Marquardt, Till Ballendat, Sebastian Boring, Saul Greenberg, and Ken Hinckley. 2012. Gradual engagement: facilitating information exchange between digital devices as a function of proximity. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*. 31–40.

[37] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, New York, NY, USA, 315–326.

[38] Nicolai Marquardt, Nathalie Henry Riche, Christian Holz, Hugo Romat, Michel Pahud, Frederik Brudy, David Ledo, Chunjong Park, Molly Jane Nicholas, Teddy Seyed, et al. 2021. AirConstellations: In-Air Device Formations for Cross-Device Interaction via Multiple Spatially-Aware Armatures. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 1252–1268.

[39] Nicolai Marquardt, Ken Hinckley, and Saul Greenberg. 2012. Cross-device interaction via micro-mobility and f-formations. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 13–22.

[40] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. 2021. Deep learning for monocular depth estimation: A review. *Neurocomputing* 438 (2021), 14–33.

[41] Michael Nebeling, Theano Mintsi, Maria Husmann, and Moira Norrie. 2014. Interactive development of cross-device user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 2793–2802.

[42] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. 2018. Protoar: Rapid physical-digital prototyping of mobile augmented reality applications. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–12.

[43] Roman Rädle, Hans-Christian Jetter, Mario Schreiner, Zhihao Lu, Harald Reiterer, and Yvonne Rogers. 2015. Spatially-aware or spatially-agnostic? Elicitation and evaluation of user-defined cross-device interactions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 3913–3922.

[44] Jun Rekimoto and Masanori Saitoh. 1999. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 378–385.

[45] Michael Rietzler, Julia Greim, Marcel Walch, Florian Schaub, Björn Wiedersheim, and Michael Weber. 2013. homeBLOX: introducing process-driven home automation. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. 801–808.

[46] Teddy Seyed, Alaa Azazi, Edwin Chan, Yuxi Wang, and Frank Maurer. 2015. Sod-toolkit: A toolkit for interactively prototyping and developing multi-sensor, multi-device environments. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. 171–180.

[47] Stephen Voida, Mark Podlaseck, Rick Kjeldsen, and Claudio Pinhanez. 2005. A study on the manipulation of 2D objects in a projector/camera-based augmented reality environment. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, New York, NY, USA, 611–620.

[48] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. 2021. GesturAR: An Authoring System for Creating Freehand Interactive Augmented Reality Applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 552–567.

[49] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. 2020. CAPturAR: An Augmented Reality Tool for Authoring Human-Involved Context-Aware Applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 328–341.

[50] Chi-Jui Wu, Steven Houben, and Nicolai Marquardt. 2017. Eaglesense: Tracking people and devices in interactive spaces using real-time top-view depth-sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3929–3942.

[51] Hui Ye and Hongbo Fu. 2022. ProGesAR: Mobile AR Prototyping for Proxemic and Gestural Interactions with Real-world IoT Enhanced Spaces. In *CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–14.