

# XD-AR: Challenges and Opportunities in Cross-Device Augmented Reality Application Development

MAXIMILIAN SPEICHER, University of Michigan, USA

BRIAN D. HALL, University of Michigan, USA

AO YU, University of Michigan, USA and Tsinghua University, China

BOWEN ZHANG, University of Michigan, USA and Tsinghua University, China

HAIHUA ZHANG, University of Michigan, USA and Tsinghua University, China

JANET NEBELING, University of Michigan, USA

MICHAEL NEBELING, University of Michigan, USA

Augmented Reality (AR) developers face a proliferation of new platforms, devices, and frameworks. This often leads to applications being limited to a single platform and makes it hard to support collaborative AR scenarios involving multiple different devices. This paper presents *XD-AR*, a cross-device AR application development framework designed to unify input and output across hand-held, head-worn, and projective AR displays. XD-AR's design was informed by challenging scenarios for AR applications, a technical review of existing AR platforms, and a survey of 30 AR designers, developers, and users. Based on the results, we developed a taxonomy of AR system components and identified key challenges and opportunities in making them work together. We discuss how our taxonomy can guide the design of future AR platforms and applications and how cross-device interaction challenges could be addressed. We illustrate this when using XD-AR to implement two challenging AR applications from the literature in a device-agnostic way.

CCS Concepts: • **Human-centered computing** → **Mixed / augmented reality**; *User interface toolkits*;

Additional Key Words and Phrases: Augmented reality; cross-device development; framework; taxonomy

## ACM Reference Format:

Maximilian Speicher, Brian D. Hall, Ao Yu, Bowen Zhang, Haihua Zhang, Janet Nebeling, and Michael Nebeling. 2018. XD-AR: Challenges and Opportunities in Cross-Device Augmented Reality Application Development. *Proc. ACM Hum.-Comput. Interact.* 2, EICS, Article 7 (June 2018), 24 pages. <https://doi.org/10.1145/3229089>

## 1 INTRODUCTION

We are experiencing the proliferation of augmented reality (AR) platforms, devices, and frameworks at a massive scale. Platforms like ARKit [23] and ARCore [22] provide AR on smartphones with only a single standard RGB camera, i.e., without motion or depth cameras that were previously required, and provide opportunities for creating novel applications. These advances have enabled motion tracking and environment sensing on common mobile devices, paving the way for more

---

Authors' addresses: Maximilian Speicher, University of Michigan, 105 S State St, Ann Arbor, MI, 48109, USA, [mspeiche@umich.edu](mailto:mspeiche@umich.edu); Brian D. Hall, University of Michigan, Ann Arbor, USA, [briandh@umich.edu](mailto:briandh@umich.edu); Ao Yu, University of Michigan, Ann Arbor, USA, Tsinghua University, Beijing, China, [ya14@mails.tsinghua.edu.cn](mailto:ya14@mails.tsinghua.edu.cn); Bowen Zhang, University of Michigan, Ann Arbor, USA, Tsinghua University, Beijing, China, [zbw14@mails.tsinghua.edu.cn](mailto:zbw14@mails.tsinghua.edu.cn); Haihua Zhang, University of Michigan, Ann Arbor, USA, Tsinghua University, Beijing, China, [haihua@yeah.net](mailto:haihua@yeah.net); Janet Nebeling, University of Michigan, Ann Arbor, USA, [janetneb@umich.edu](mailto:janetneb@umich.edu); Michael Nebeling, University of Michigan, Ann Arbor, USA, [nebeling@umich.edu](mailto:nebeling@umich.edu).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.

2573-0142/2018/6-ART7 \$15.00

<https://doi.org/10.1145/3229089>

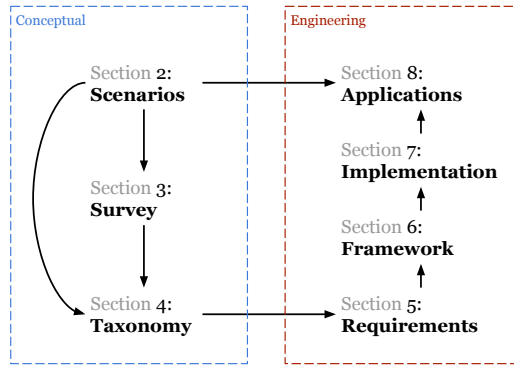


Fig. 1. The two main processes of our research: a) Conceptual: from scenarios to taxonomy; b) Engineering: from requirements to applications.

widespread adoption. However, this rapid growth also leads to a fragmentation of the ecosystem. While most existing research on AR has focused on hard technical problems with AR technologies, such as perception, annotation, visualization, and sensing [1, 2, 10, 24], this fragmentation leads to a variety of new challenges for application developers.

Available frameworks differ in a wide range of characteristics, from system architecture, platform and hardware (in)dependence, to intended scope and provided features. Applications created with one framework typically only run on one kind of device, and porting to another framework to support more devices is cumbersome or prohibitively expensive. For example, applications that rely on marker tracking are developed entirely differently than ones that work with spatial mapping. It is thus becoming increasingly difficult for designers and developers to maintain an awareness of the available options, understand the strengths and weaknesses of different frameworks, and figure out ways to best combine available resources to satisfy application requirements.

In this paper, we present *XD-AR*, a cross-device framework that we designed to study the main issues that must be overcome to allow developers to create AR applications that serve the broad spectrum of hand-held, head-worn, and projective AR devices. *XD-AR* adds to the evolving research on cross-device frameworks [7, 15, 26, 27, 38], but specifically focuses on the new class of rapidly evolving AR devices and platforms such as HoloLens and ARKit, as well as the Tango and ARCore projects, which have been recently merged. It enables developers to quickly create applications that allow HoloLens users to, e.g., place an object in a room, while other users can view and interact with that object through an AR-capable smartphone. *XD-AR* facilitates a wide range of novel cross-device applications using AR, thus enabling exploration of new interesting combinations of AR displays, especially for collaboration in multi-user settings.

To create *XD-AR*, we first studied existing frameworks to obtain a better understanding of the challenges and opportunities in cross-device AR development. Our study was informed by three sources: a) two motivating scenarios that are based on existing work and are challenging to implement because this would require integration of multiple existing frameworks; b) a survey of experienced AR designers, developers, and users to identify their needs and expectations; c) a technical review of the current ecosystem of AR frameworks to provide a better understanding of the conceptual and technical differences underlying existing AR systems.

Figure 1 gives an overview of the flow of the paper and how the different conceptual and engineering parts inform each other. We start by developing the two scenarios in the focus of our investigation and deriving a set of *challenges* in Section 2. The scenarios are also part of our survey with AR developers, designers, and users, from which we extract common *themes* in Section 3.

Next, in Section 4, we present our technical review, from which we derive a *taxonomy of AR frameworks* that captures the main dimensions of existing AR frameworks and is informed by both the scenarios and the survey. Based on these results, in Section 5, we distill five requirements for frameworks like XD-AR. In Section 6, we introduce the XD-AR framework and its various components to support cross-device development. This is followed by a description of our specific implementation of XD-AR in Section 7. Finally, in Section 8, we describe two proof-of-concept applications implementing the two scenarios using XD-AR.

## 2 CHALLENGING SCENARIOS AND AR APPLICATIONS

As the first step in understanding challenges and opportunities of AR, we identified two scenarios based on existing work that are difficult to implement with current frameworks and technologies. In this section, we describe and illustrate these two scenarios and use them throughout the paper to derive technical and design issues.

### 2.1 Scenario 1: Furniture Placement



Fig. 2. AR Furniture Placement: All family members can use their own mobile devices to place virtual models of new furniture in the room.

In the first scenario, a family wants to buy new furniture for their living room (Fig. 2). To decide on which furniture to buy and where it should be placed, they install an AR app on their various devices, including a Tango phone, an iPhone, a Samsung tablet PC, and others. The app allows them to virtually place various types of properly-sized pieces within the room. Furniture can be placed in the room in two ways: either by placing a picture of the furniture from the company's catalog on the floor, or by dragging and dropping a virtual 3D object of the furniture into place using the app itself.

This scenario is based on efforts by IKEA to enable customers to preview digital furniture directly in their own homes, and has a number of pertinent characteristics. First, it is a multi-user, multi-platform and multi-device scenario. Second, the scenario mostly relies on implicit interaction (moving the device around virtual objects), while requiring only limited explicit interaction (actively placing the furniture in the room). Third, the scenario includes both marker-based and marker-less tracking approaches. Finally, it is not specified whether cross-device interaction is supported

(i.e., digital furniture and manipulations thereof are propagated to all other users' devices). Three challenges can be identified:

- (C1) The AR content must be available on different everyday, and potentially low-end, devices.
- (C2) The AR application must provide stable tracking and positioning w.r.t. implicit interaction, i.e., the pieces of furniture should stay in their exact positions when users move around in the room.
- (C3) Depending on the platform and device, the app must decide between marker-based and marker-less tracking (e.g., based on spatial mapping and plane detection, if available).

Overall, this is a moderately complex scenario that is aimed towards the needs of average consumer end-users.

## 2.2 Scenario 2: Shooter Game

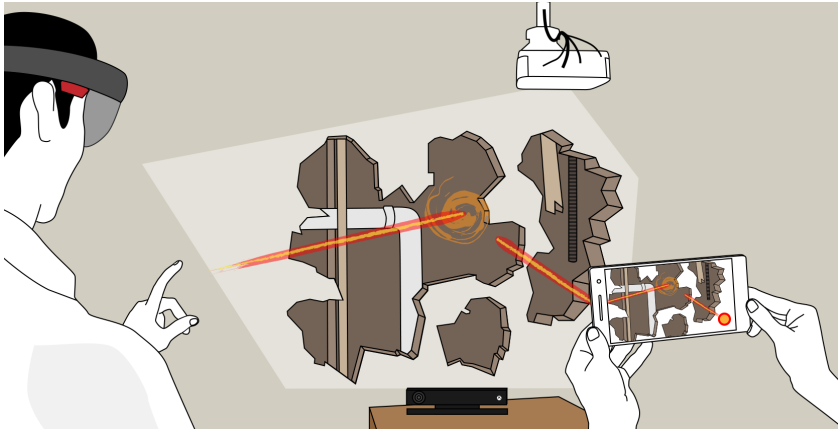


Fig. 3. AR Shooter: multiple users can play a laser shooting game together with AR headsets (left), projection (middle), or hand-helds (right) using mid-air or touch gestures. The image is shown from the perspective of the third person, who is playing using the projection set-up and whose movements are tracked by the Kinect.

In the second scenario, three friends want to play an AR-based shooter game (Fig. 3). Their equipment includes a HoloLens, a RoomAlive setup (projector + Kinect) [17], and a Tango phone, all of which have been placed in a living room. The game allows them to shoot laser beams that destroy the walls and shoot down alien robots floating in the room. The laser beams are activated using an air-tap gesture when using HoloLens, closing a fist in the RoomAlive setup [17], and by tapping the screen of the Tango phone. Each of the friends could be playing using any of the devices. The virtual parts of the environment (destroyed walls, alien robots) are the same for all players and each player can see others' laser beams.

This scenario is based on Microsoft's RoboRaid game for HoloLens, and exhibits its own set of relevant characteristics. First, it is multi-user, multi-platform, and cross-device. Second, it is display-agnostic. Third, it is a highly interactive gaming experience with a great deal of explicit interaction (shooting lasers) based on multiple input modalities. Finally, the AR app requires a spatial understanding of the room to be able to detect the walls. Based on these characteristics, we have identified four additional challenges:

- (C4) The AR content must be available for different displays and output modalities.

- (C5) All digital objects must be the same for everyone, thus the AR content must be synchronized between multiple users using different devices in real time.
- (C6) Depending on the device and input modality, the app must decide which gesture to use for shooting lasers.
- (C7) While each device can map the surroundings independently, there must be a common, global coordinate system for correct placement of the digital objects.

Overall, this is a more complex scenario than the previous one, geared towards designers, developers, and power users.

### 3 SURVEY OF AR TECHNOLOGY EXPERIENCES

To gain a better understanding of the perspectives and experiences that AR designers, developers, and users have developed from working with existing technology, we conducted a survey of challenges with current applications and devices, as well as beliefs about the future of AR. Also, this survey was intended to validate the challenges (C1–7) we derived from the previous scenarios.

#### 3.1 Method

We designed an online survey to collect opinions and expectations from people who have experience with AR applications. Our targeted participants consisted of three groups of people: AR designers/developers, AR users, and those who consider themselves as both.

The survey included 30 questions distributed over three parts: demographic information and AR experience; opinions on current AR frameworks and applications; presentation of two cross-device AR application scenarios (as detailed in the Challenging Scenarios section), followed by questions to identify challenging aspects of using or designing/developing such an application; and questions about an ideal framework and the future of AR. Most questions were optional and open-ended, to allow participants to fully express their opinions.

Before disseminating the survey, we piloted with three AR designers/developers in our research group, estimated the time for completion to be about 20 minutes, and fixed minor issues based on their feedback. We targeted participants with AR experience by email containing an anonymous survey link and a brief introduction, via email lists at university research groups and companies concerned with AR. We also posted requests to social media platforms with vendors and interest groups related to AR frameworks and technologies. Regarding regional demographics, we achieved the highest response rates from Europe and the U.S. The complete survey can be found in Appendix A.

#### 3.2 Results

The survey gained a total of 30 responses with an average completion time of 31 minutes<sup>1</sup>. The 30 participants (23 male, 7 female) ranged in age from 20 to 60 (average age of 34), and chose roles of either designer/developer (13), user (7), or both (12). Participants had backgrounds in academia (15), industry (7), and other areas, while their occupations included student, researcher, physician, developer, and CEO of an AR company. All participants had at least some experience with designing, developing, or using AR applications. About half of our participants had at least one year of experience, having been involved with at least three AR projects or applications. Most participants had used wearable AR devices (22)—like Microsoft HoloLens—and/or hand-held AR devices (16). Generally speaking, our participants came from a wide range of backgrounds, allowing them to provide input from diverse points of view.

We used thematic coding, an exploratory data-driven method to code and analyze the qualitative data [32]. A total of six major themes, each of which was addressed by at least nine participants,

<sup>1</sup>We excluded three outliers to calculate completion time. Their responses are included in the results.

appeared in our analysis: **(T1)** cross-device/cross-platform communication, **(T2)** environmental mapping, **(T3)** obtrusiveness, **(T4)** gestures, **(T5)** tracking, and **(T6)** field of view.

21 of the 30 participants, both designers/developers (d/d) and users, mentioned *(T1) cross-device/cross-platform communication* either as a challenging limitation for AR development, or a must-have for an ideal framework in the future. For instance, 16 of the 28 answers to the first application scenario suggested communication between the devices as a major challenge. P7 (user) asked, “Do these applications communicate with each other – can one person switch/suggest to the other user of mobile app?”, while P12 (d/d) said “Computing shared world anchors is an unsolved problem for current AR technology.” However, few designers or developers were able to propose an effective solution. P9 suggested to switch the devices or upload and download the scene as a whole. Moreover, P2 stated that a good unified development environment for cross-device AR applications is not available, and that this problem would mostly be addressed through hard work, as “this same program would need to be mostly written several different times for the different devices.”

*(T2) Mapping the environment* was mentioned as a major weakness, technical limitation, or particularly challenging part of the application scenarios by 11 participants. To give just one example, referring to the furniture placement application, P4 (both) stated it would be challenging to “[r]ecognize the real world and keep the virtual element on the correct position and scale when moving.”

Regarding the size and design of both, current AR devices and the types of markers used, 10 participants mentioned that these are *(T3) too obtrusive* for everyday use. With respect to the future of AR, 8 of the 26 answers state that participants expect technology to be more suitable for everyday use by being less obtrusive. P3 (user) mentioned, “for AR to take off we need a low-cost display and honestly, something less obtrusive than Google glass or the HoloLens.”

Out of 12 participants who addressed *(T4) gesture recognition*, the majority (9) viewed it as a weakness that requires improvement, while 3 stated that it was an advantage in current frameworks. Issues mentioned included: developing new gestures, customizing existing ones, and more accurate recognition. For instance, P23 (d/d) wrote “it is easy to implement [predefined] gestures in AR frameworks”, but also indicated that custom gestures were challenging. In particular, 3D gestures in AR systems are more sophisticated and complex compared to 2D gestures. P13 (both) suggested that there should be “pre-build components/libraries for gesture recognition” in an ideal AR framework.

Another common theme participants often referred to was *(T5) tracking*. While 5 participants listed device or marker tracking as a strength of current frameworks, there were still 9 who pointed out tracking capabilities need improvement: P5 (both) mentioned “AR marker recognition and tracking in the HoloLens, to tie holograms to real moving objects rather than to the enclosing room” as a weakness of that specific framework. P2 (d/d) suggested an ideal AR framework would have to feature “high-performance tracking of the camera or user.”

Narrow field of view remains an unsolved problem, as 9 participants noted the *(T6) fields of view* of current devices were too limited for useful applications. P8 (user) described the future of AR in 10 years as having “full field of view with the size of normal sunglasses.” With respect to the furniture app, P22 (user) believed the challenging part was to see the whole room furnished, since “you always only see the small screen of your smart phone”, and they suggested this could be addressed by a “360°-panorama which I could pan and zoom so that I can see the whole room furnished”, which, however, is not an AR solution.

For completeness, 5 participants addressed scaling and sizing a virtual object in an AR scene as challenging aspects, particularly with regard to the first application scenario. Concerning challenging applications and the future of AR, 5 participants expected gaming to be the driving force of this technology. Navigation, education, working assistance, and health care, among others, were mentioned several times.

Major Dimension	Minor Dimension	Values	Examples
Hardware & Software	Operating System	Windows	MixedRealityToolkit, RoomAlive, Vuforia
		Android	Tango, ARCore, Vuforia
		iOS	ARKit, Vuforia
	IMU	✓	MixedRealityToolkit, Tango, ARKit
		—	RoomAlive, ARToolKit 6
		Device-Dependent	ARCore, Vuforia
	Regular Camera	✓	MixedRealityToolkit, Tango, ARCore, RoomAlive, Vuforia, ARToolKit 6, ARKit
Environmental Sensing & Understanding	Depth Camera	✓	MixedRealityToolkit, Tango, RoomAlive
		—	ARCore, Vuforia, ARToolKit 6, ARKit
	Other Sensors	✓	MixedRealityToolkit, Tango
		—	ARCore, RoomAlive, Vuforia, ARToolKit 6, ARKit
	Marker	✓	Tango, Vuforia, ARToolKit 6
		—	MixedRealityToolkit, Tango, ARCore, RoomAlive, Vuforia, ARKit
		✓	MixedRealityToolkit, Tango, ARCore, RoomAlive, Vuforia, ARKit
	Markerless	—	ARToolKit 6
		Dynamic	MixedRealityToolkit, Tango, ARCore, Vuforia, ARKit
	Environmental Mapping	Static	RoomAlive
Coordinate Systems & Anchors		—	ARToolKit 6
	Terrain Identification	Plane	MixedRealityToolkit, Tango, ARCore, Vuforia, ARKit
		Floor	MixedRealityToolkit, Tango, RoomAlive
		Ceiling, Walls	MixedRealityToolkit
	Tracking & Positioning (stability)	Low	ARToolKit 6
		Medium	Tango, ARCore, RoomAlive, Vuforia, ARKit
		High	MixedRealityToolkit
	Coordinate Systems	Right-Hand Standard	MixedRealityToolkit, RoomAlive, Vuforia, Tango, ARCore, ARToolKit 6, ARKit
		Left-Hand Standard	MixedRealityToolkit, RoomAlive, Vuforia, Tango, ARCore, ARToolKit 6
		Other Orientations	Tango, ARCore
Explicit Interaction	Anchors	World	MixedRealityToolkit, Tango, ARCore, RoomAlive, Vuforia, ARKit
		Shared World	MixedRealityToolkit
		Spatial	MixedRealityToolkit, Tango, ARCore, RoomAlive, ARKit
	Touch	Object	MixedRealityToolkit, Tango, ARCore, RoomAlive, Vuforia, ARToolKit 6, ARKit
		✓	Tango, ARCore, ARToolKit 6, ARKit
		—	MixedRealityToolkit, RoomAlive
		Device-Dependent	Vuforia
	Gesture	3D	MixedRealityToolkit, RoomAlive
		2D	Tango, ARCore, Vuforia, ARKit
		—	ARToolKit 6
Output to User	Device Movement	✓	MixedRealityToolkit, Tango, ARCore, Vuforia, ARKit
		—	RoomAlive, ARToolKit 6
		✓	MixedRealityToolkit, ARKit
	Voice	—	Tango, ARCore, RoomAlive, Vuforia, ARToolKit 6
		✓	MixedRealityToolkit, Tango, ARCore, Vuforia, ARToolKit 6, ARKit
	Physical Buttons	—	RoomAlive
		—	MixedRealityToolkit, RoomAlive
	Visual Display	See-Through	MixedRealityToolkit, Vuforia
		Touchscreen	Tango, ARCore, Vuforia, ARToolKit 6, ARKit
		Projector	RoomAlive
	Audio	✓	MixedRealityToolkit, Tango, ARCore, RoomAlive, Vuforia, ARToolKit 6, ARKit
	Haptic	Vibration	Tango, ARCore, Vuforia, ARToolKit 6, ARKit
		Device-Dependent	Vuforia
		—	MixedRealityToolkit, RoomAlive

Table 1. Taxonomy in terms of major and minor dimensions, possible values for each dimension, and examples from our review of existing AR frameworks that represent a particular value.

Overall, the above major themes provide an understanding of the current challenges in AR, as described from the perspective of experienced designers, developers, and users. Moreover, they reflect all of the challenges we identified based on the scenarios in the previous section. In the following, together with other sources, we will use this feedback to inform our taxonomy of AR frameworks.

4 TAXONOMY OF AR FRAMEWORKS

To obtain a general understanding not limited to any particular implementation, we have developed a taxonomy of features available in many popular AR frameworks. We began by analyzing existing AR frameworks, reviewed prior research in cross-device and AR fields, and examined the participant responses from our survey. Specifically, our analysis of AR frameworks included Mixed-RealityToolkit (formerly HoloToolkit for HoloLens), ARToolKit, ARKit, Vuforia, Kudan, Wikitude,



MAXST AR SDK, Tango, Argon.js, AR.js, EasyAR, Xzing, RoomAlive Toolkit, and ARCore. Some frameworks are implemented as component-based architectures [3], while others tightly integrate all functionality beyond a unified API, so this taxonomy focuses on conceptual categories rather than implementation-specific details.

From this set of materials, we generated minor dimension categories to encapsulate similar functionality offered by multiple different frameworks, but which were often described or implemented differently between sources. Once a stable set of minor dimensions was achieved, we then grouped together the minor dimensions according to conceptual similarity to form the major dimensions. Finally, we revisited our original analysis of existing frameworks to classify them according to the taxonomy of AR frameworks.

Table 1 shows the taxonomy together with examples of AR frameworks that represent a particular value of each dimension. Note that we decided to lead with dimensions rather than examples to emphasize the conceptual part of our contribution. It is also an attempt to make sure that, despite the rapidly evolving space of AR framework implementations (e.g., Tango was recently merged with ARCore), there is value in our paper as a reference for future AR researchers and developers.

Below we will describe each major dimension of AR frameworks (minor dimensions given in *italics*), highlighting framework differences that must be handled in cross-device development.

#### 4.1 Hardware and Software Restrictions

Most AR frameworks today are heavily tied to hardware and software requirements, including: *operating system* (and version); presence and accuracy of an *inertial measurement unit (IMU)*, which can include accelerometers, gyroscopes, and magnetometers; *regular digital camera*; *depth camera*, which usually includes an IR emitter and receiver; and *other less-common sensors*, such as motion cameras, LIDAR systems, or binocular digital cameras.

This dimension of the taxonomy was also informed by *C1*, *C3*, and *C5* (cf. Challenging Scenarios and AR Applications section), as well as *T1*, *T3*, and *T6* (cf. Survey of AR Technology Experiences section).

#### 4.2 Environment Sensing and Understanding

Environment Sensing and Understanding refers to a wide variety of functionality related to recognizing, learning, understanding, tracking, and positioning an AR device within the physical world [39]. All frameworks support these features to varying degrees, and include *marker recognition*, *markerless techniques*, *environmental (spatial) mapping*, *terrain identification* (finding the floor, ceiling, walls, etc.), *tracking*, and *positioning*.

Though all frameworks and devices support at least some of these features, as AR applications often fundamentally require spatial awareness [30], they do not do so equally. For instance, in RoomAlive the geography of the room is learned once during calibration, no markers are used or supported, and only tracking and positioning of the user is performed as the devices themselves do not move. At the other end of the spectrum, the HoloLens allows complex spatial mapping and supports some terrain identification natively (including distinctly identifying floors, walls, and ceilings in closed environments). Finally, on Tango phones both markers (ARTag [11] and QR Codes) and markerless techniques can be used, and some built-in terrain identification is available.

Previous work has investigated a number of creative solutions to addressing this category of functionality, many of which are not achievable in single-device applications available today. Solutions investigated have included ubiquitous sensing approaches using infra-red [18] and open architecture approaches to fusing a network of sensors [31], which can potentially go beyond single-device SLAM techniques [35].

This dimension was also informed by *C2*, *C3*, and *C5*, as well as *T2*, *T3*, and *T5*.



### 4.3 Coordinate Systems and Anchors

In the world of 3D computer graphics, there are a number of *coordinate systems* to choose from, with each having its own orientation of coordinate axes. Note that this variety of coordinate systems goes beyond the common descriptions of “left-hand” and “right-hand” at times, as these conventions have not been universally adopted. To give just one example, in Tango there are three coordinate systems you might encounter: *a)* Right Hand Local Level, where x-positive points right, y-positive points away from the user, and z-positive points up; *b)* Right Hand Android, taken from the Android Sensor Coordinate System, where x-positive points right, y-positive points up, and z-positive points towards the user; and *c)* if you are developing in Unity you will use a system where x-positive points right, y-positive points up, and z-positive is away from the user. Hence, as an object’s position is generally described as an unlabeled ordered set of three numbers without references to the specific axes, such as (0, 1, −2), something as seemingly simple as communicating the position of an object between devices becomes a challenge.

An additional challenge of coordinate systems is unit translation, or the extent to which units of virtual space can be understood to be equivalent to units of physical space. In both HoloLens and Tango, one unit in virtual coordinate space equates to one meter of physical space (within some margin of error that varies by device and condition). Not all devices and systems provide coordinate mapping this simple, and especially systems without spatial mapping will require special calibration and unit scale translation to allow them to be able to collaborate with other devices.

*Anchors* “are a way for the virtual world to interact with the real world.”<sup>2</sup> They allow a hologram’s position to be bound to the physical geometry of the environment. HoloLens provides such generic anchors, as well as a special world anchor that can be communicated to other HoloLens devices. This world anchor, which is synchronized across HoloLenses, serves as a shared reference point in the physical environment for virtual object placement. However, as non-HoloLens devices do not have the same representation of the physical world, they can not make use of this world anchor. The other extreme is ARToolKit, which does not generate a persistent understanding of the physical environment whatsoever, so anchors are generally limited to being attached to fiducial markers.

RoomAlive itself works in a significantly different way than the other AR systems we discuss. In this system the environment is learned in advance as part of system calibration, and there is no native use of anchors as only the user moves (tracked by the Kinect) while the physical environment cannot be understood to change or move.

This dimension was also informed by *C5*, *C7*, and *T1*.

### 4.4 Explicit Interaction

Explicit Interaction covers the various user input methods available to AR applications, including *touch*, *gesture*, *device movement*, *physical buttons*, and *voice* [21]. By device movement, we mean that a framework detects the user intentionally moving the head or rotating the phone and the framework generates events accordingly. The primary user interaction methods on the HoloLens are gesture and voice, while in RoomAlive user gestures (hand movement) is dominant.

In terms of gestures, MixedRealityToolkit for the HoloLens supports the basic air tap with one finger and the bloom gesture of opening the palm-up closed hand, as well as composite gestures including tap, hold, manipulation, and navigation. The HoloLens also provides access to raw hand-detection and a higher-level gesture recognition engine to allow custom gestures to be defined. On phone-based AR, such as ARCore and ARKit, most interaction is in the form of moving the phone and touching the screen. In-air gestures on hand-helds could potentially be used more in the future, though increased energy use could further exacerbate battery drain in already process-intensive AR

<sup>2</sup><https://docs.unity3d.com/Manual/windowsholographic-anchors.html> (accessed January 10, 2018).

applications [34], while most current hardware limits the ability to use both front and rear-facing cameras simultaneously. Finally, in projective AR (RoomAlive), all input is gesture and pose based, using Kinect as a stationary input sensor.

This dimension was also informed by *C6*, *T1*, and *T4*.

#### 4.5 Output to User

Output to the User is a universal requirement of AR applications, and current AR frameworks primarily focus on *visual graphics*. The details of how graphics are displayed, and the limitations of what sort of effects are possible, vary tremendously by device. While devices like Tango phones and HoloLens support many complex graphical effects, many commercial headsets are still limited to graphics that can only appear as an overlay like a Heads-Up Display (HUD). Note that HUD-only displays do not fit some definitions of AR, such as those that require virtual objects be “registered in three dimensions” [2], but these sorts of limitations must still be considered in cross-device environments, even if only to exclude these devices from support.

*Audio* is supported in most AR systems, but the visual-focused history of the field has tended to minimize the emphasis on using audio as a fundamental part of an AR application. HoloLens is currently an outlier, as its support for full spatial audio without headphones has been emphasized. Phone-based systems could offer 3D spatial audio experiences if users wore headphones, and Unity and Unreal engines already offer built-in support for spatial audio. On the other hand, projective AR systems like RoomAlive are currently limited to general room speaker systems, which could interfere with voice input and spatial audio use for other people in the room.

There are many *other forms* of providing output to the user, especially through haptic methods like vibration [19], force-feedback [8], and others. This remains a nascent area, and to date has been minimally used or supported in frameworks beyond vibration features natively available in smartphones.

This dimension was also informed by *C4* and *T6*.

### 5 REQUIREMENTS FOR XD-AR

From the scenarios, survey, and analyses discussed previously, we synthesized requirements for our XD-AR framework. These requirements represent the key challenges that any cross-device AR framework will need to overcome. Requirements 1–4 address technical aspects and are therefore *functional requirements* while Requirement 5 addresses user experience and is thus *non-functional*.

**(R1) Transcend device and platform restrictions.** As discussed in Hardware and Software Restrictions, the multiplicity of hardware and software platform configurations creates an exceptionally challenging environment for the design and development of cross-device applications, as location accuracy, tracking stability, and display quality can vary wildly, and access to sensors can be restricted by the device’s software stack or by just not being included on a device.

**(R2) Transcend device-specific sensing.** Based on the analysis of Environment Sensing and Understanding, the multitude of available sensors, along with their own specific processing techniques, poses a unique challenge to cross-device AR development. Any AR cross-device framework will need to find a way to allow devices that function drastically differently to still work together, thus transcending device-specific sensing.

**(R3) Connect coordinate systems.** As shown in Coordinate Systems and Anchors, in current implementations, each device will potentially have its own internal coordinate system, using different anchors and reference points, with even identical devices often developing different coordinate positions and mapping of the physical environment. Cross-device applications

will need to use some method to allow devices, each with their own understanding of the world, to communicate position and orientation in a way that other devices can use and so allow the connection of coordinate systems.

**(R4) Resolve differing interaction modalities.** As analyzed under Explicit Interaction, different devices benefit from—or strictly require—different interaction modalities to be used. Hence, a cross-device approach will require some way to resolve differing modalities (e.g., touch and air-tap) used for the same higher-level events and actions (e.g., placing a virtual table).

**(R5) Resolve varying end-user experience.** As indicated under Output to User, variance of output capability is a challenge for cross-device AR applications since all users, independent of device and platform, shall experience the AR scene in the same way. Thus, there is a need to resolve varying end-user experience.

## 6 XD-AR FRAMEWORK

So far we have developed a taxonomy of AR frameworks and extracted requirements specific to cross-device AR development. These investigations culminated in the design and implementation of our XD-AR framework. Before detailing our implementation in the next section, this section describes XD-AR at the conceptual level in terms of the main components, which we believe represents the minimum set of components for cross-device support in future AR frameworks.

XD-AR is an extensible framework for cross-device AR, intended to assist future designers and developers in their efforts to engineer AR systems in a more platform and device-agnostic way. XD-AR is designed to support a wide variety of architectures, such as client–server (as we chose to do in our proof-of-concept implementation detailed later) or peer-to-peer. It comprises three components: Communication Management, Coordinate Management, and Session Management.

### 6.1 Communication Management

This component is concerned with two key tasks. First, it enables real-time synchronization across devices; e.g., when a HoloLens user air-taps to grab and move a piece of virtual furniture, all other (non-HoloLens) devices must be able to see this interaction in real time at the correct position in the physical environment. Inspired by prior work [29], XD-AR relies on a unified messaging format that abstracts from low-level events (such as the air-tap) and device-specific coordinate representations. The messages emitted by a device are processed by the framework and then relayed to all devices that need to be kept in sync. Second, Communication Management concerns the issues involved in effective networking of devices across many potential complex network topologies. For instance, some devices may not support push messaging protocols, and the server will need to allow such devices to communicate using whatever fallback methods are available, e.g., HTTP long polling.

### 6.2 Coordinate Management

If messages sent to XD-AR are not already translated relative to the shared origin point (i.e., world anchor), XD-AR will perform this translation. Since different devices use different orientations for their coordinate systems, as explained in the Coordinate Systems and Anchors section, XD-AR will detect the type of device a message comes from and then alter the coordinates to fit the target device before relaying the message. This approach allows each device implementation to only be aware of its own coordinate system, thus eliminating the need to alter individual client implementations when a new type of device connects to the system. An implementation could require that individual devices perform translations into a global coordinate system on their own, but nevertheless coordinate systems must be translated in some manner for a working XD-AR system to be achieved.

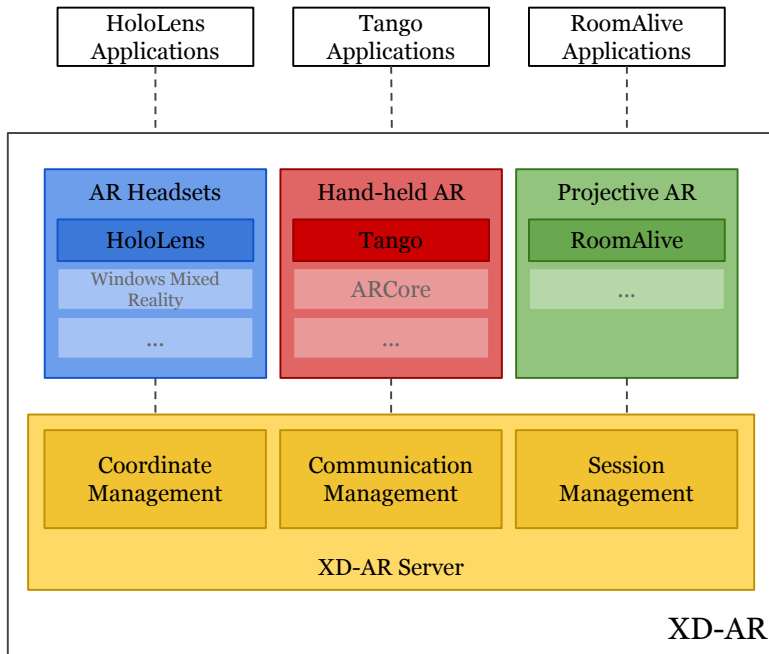


Fig. 4. Example implementation of XD-AR, integrating with HoloLens, Tango, and RoomAlive. Extensions to support Windows Mixed Reality headsets and ARCore are possible directions of future work.

### 6.3 Session Management

Session Management is responsible for three key tasks: device registration, device identification, and virtual object persistence. To illustrate the work of this component, consider an ongoing cross-device session between a HoloLens and a Tango device, with a virtual table placed in the middle of the room. If both devices leave the session, they should be able to rejoin and find the table in the same position that they previously left it.

Devices joining a session must register and after that, XD-AR must ensure that each device can be uniquely identified, with appropriate data passed to Communication Management as required. Session Management is moreover responsible for ensuring that virtual objects are persisted, so that devices can drop in and out of a session over time without the AR scene being effected. In this way, virtual objects are effectively attached to the physical environment rather than (individual) devices. XD-AR implementations can choose what is worthy of persistence, as suits application requirements. For instance, it might not be necessary to persist full environmental maps if they would be recreated by devices in a new session anyway.

## 7 XD-AR IMPLEMENTATION

Having conceptually described the XD-AR framework, this section describes an example implementation (Figure 4). Our XD-AR implementation supports a variety of current devices and AR client libraries: HoloLens, Tango hand-held phones, and RoomAlive [17]. These were chosen as prominent examples of hand-held, head-worn, and projective AR at the time our implementation was created. ARCore and ARKit were released later and will be discussed in Section 9.2. Our implementation is based on a layered architecture with all generic functionality being handled by the XD-AR server, which takes care of translation between different devices and platforms. On top of the server resides

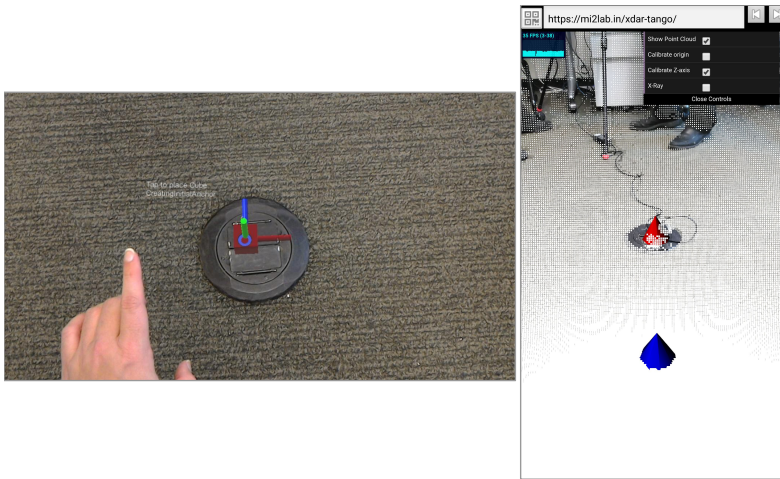


Fig. 5. The calibration process of the shared world anchor on HoloLens (left) and Tango (right). The direction of the z-axis is indicated by the blue features.

a layer of necessary platform-specific components. Device-specific applications (the top layer) can then be realized based on XD-AR as a whole.

Two of the authors implemented XD-AR over a 3-month full-time summer internship. They were computer science undergraduate students with extensive experience developing for HoloLens. While it is difficult to provide an objective metric for development effort, we can assume that anyone intending to develop similar cross-device AR applications without using XD-AR would have to invest a similar amount of work upfront.

## 7.1 XD-AR Server

The XD-AR Server has been implemented according to the framework specification given above, and thus handles coordination between various participating client devices (Figure 4, yellow). Our implementation of the server is based on *Node.js*. We will elaborate in the Discussion section on how it can be extended to support additional device types and platforms.

**7.1.1 Communication Management.** Depending on the specification of the client, XD-AR provides a *WebRTC* data channel or an HTTP GET endpoint for long polling (in case the client does not support push messages). Data is transferred in the form of a one-dimensional arrays of objects. These objects comprise a high-level command (e.g., insert, move, etc.), a corresponding virtual object (e.g., a sphere, chair, table, etc.) or object reference, object position in terms of a three-dimensional vector (in local coordinates), object rotation in terms of a quaternion, coordinate scale in terms of a three-dimensional vector, and their device ID.

**7.1.2 Coordinate Management.** Since different devices use different orientations for their coordinate systems—as explained in Section “Coordinate Systems and Anchors”—the server detects the specification of the device the message comes from based on the device ID, and then alters the coordinates to fit the target device before relaying the message. For this, in a calibration step, our implementation requires all devices to agree on the shared origin point and direction of the z-axis in order to establish the global coordinate system.

**7.1.3 Session Management.** Clients initially contact the server using WebRTC or HTTP, provide their specification (i.e., HoloLens, Tango, or RoomAlive), as well as their calibrated shared world anchor and z-axis (based on the device's local coordinate system) and get assigned a device ID. This is necessary for XD-AR to create a global coordinate system and translate between connected devices. All digital objects added to a scene are persisted to a *MongoDB* database with their global coordinates. Any device joining the session at a later point in time can be provided with those objects, which allows the objects to virtually “stay in place” in the room when the system is shut down and restarted.

## 7.2 HoloLens

The HoloLens part of our implementation (Figure 4, blue) is a *Universal Windows Platform* (UWP) application developed in *Unity* based on the *MixedRealityToolkit*. HoloLens uses spatial mapping and provides its own unique data structure called a *WorldAnchor*, which is tied to a point in the physical space and shared with all other devices of the same kind that enter a session at a later point. The location of the shared world anchor and direction of the z-axis are also communicated to the XD-AR Server upon initialization, after the user has specified both in relation to a physical reference point on which everyone agreed (Figure 5). Communication between individual HoloLens devices happens exclusively via a dedicated HoloLens server, while they communicate with the XD-AR server separately. Updates to the AR scene are sent to XD-AR via HTTP POST while performing long polling on a separate HTTP GET endpoint to receive updates from non-HoloLens devices.

## 7.3 Tango Hand-held Phones

The Tango part of our XD-AR implementation (Figure 4, red) is based on *WebGL* and *three.js*. Tango devices provide their own world anchor, which has to be defined when the corresponding web application is opened (Figure 5). To calibrate this anchor, which needs to be in the same position as the one on HoloLens, the shared point of reference in the physical world is touched within the application. A second point is then touched to establish the shared direction for the z-axis, which is sufficient since we always keep the y-axis of the anchor orthogonal to the floor plane detected by Tango. Based on this, the translation and rotation offset can be calculated as a vector relative to the device's native world anchor by XD-AR. For communication, Tango phones use the WebRTC data channel provided by the XD-AR Server.

## 7.4 Projective AR

The projective AR part of our implementation (Figure 4, green) has been developed based on *Unity* and the *RoomAlive* toolkit. The process to set up this component includes using the *ProCamCalibration* tool provided by *RoomAlive* to calibrate a projector in relation to a Kinect camera, in this way building a 3D mapping of the projective region. Subsequently, in *Unity*, a virtual object is placed on the shared reference point already used by HoloLens and Tango (Figure 5). This special virtual object acts as the world anchor. Then, the *Unity* project is run to display the AR scene based on the previously obtained mapping. Like HoloLens, this part of the system does not support WebRTC, and therefore communicates with the XD-AR server via HTTP in the same way. The position of the shared world anchor in relation to the Kinect origin (which acts as the origin of the local coordinate system) is provided to XD-AR during initialization, so that the framework can translate between global and *RoomAlive* coordinates. This part of our system automatically locks onto the first user detected by the Kinect to distinguish between the *RoomAlive* and the HoloLens and Tango users.



*RoomAlive*. During application operation, Kinect is used to track the position and rotation of the user, and to detect gestures. A dedicated Kinect server receives messages from the attached Kinect cameras and passes the messages to the Unity application. This application then calculates how to display the projected images relative to the position of the user, so as to give the appearance of being registered in 3D relative to the shared origin point. The projectors receive these images from the application and project them accordingly. For full details on the functionality and design of RoomAlive beyond our use of the system, please refer to the original RoomAlive paper [17].

## 7.5 Satisfying XD-AR Requirements

In the following, we revisit the five requirements from Section 5 and explain how each is satisfied by our XD-AR implementation.

(R1) To transcend device and platform restrictions, we used a hybrid approach. We moved as much functionality to XD-AR as was technically feasible, relying on device-specific implementations when necessary. This allowed us to use the unique features and libraries available on each of the available devices, while the server greatly limited the amount of functionality that needed to be re-implemented on each supported device. It has to be noted that this is the only feasible approach since unifying platforms is not scalable at this point in time without extensive vendor support. In this sense, our approach is similar to the one Mozilla's new WebXR project<sup>3</sup> takes.

(R2) Currently, a scalable approach to unified sensing would require reinventing core computer vision libraries. Hence, considering the scope of our project, we relied on device-dependent implementations for Environment Sensing and Understanding. This technique affords the greatest flexibility for developers, as there is no currently available method to use the greater abilities of devices like HoloLens to support, e.g., smartphones (which we intend to pursue as one direction of future work). This is especially true where projective AR applications are concerned, as the methods used are substantially different from other AR systems.

(R3) For Coordinate Systems and Anchors we used a shared physical world reference point, which each device was individually calibrated with, and then each application reported virtual objects relative to this shared reference point. The XD-AR Server handles the translation of coordinate systems between devices, so that adding a new device does not require that every other device's software be updated.

(R4) To resolve differing modalities of Explicit Interaction we used a unified messaging format that abstracts from device-specific low-level events and instead describes higher-level interactions. This way HoloLens can use air-tap gestures, Tango can use touchscreen taps, and projective applications can use hand-close gestures, all representing the same interaction, e.g., shooting a laser beam, without limiting application functionality for all devices. XD-AR serves the purpose of translating between high-level interactions and low-level events based on the devices involved.

(R5) To resolve the varying end-user experience provided by different options for Output to User, we had to rely on platform-specific implementations for, e.g., different virtual objects and interactions since we use both Unity and WebGL. Yet, through the aforementioned unified messaging format, our XD-AR implementation abstracts from these by relaying, e.g., a higher-level "laser beam" event, which is then appropriately visualized according to the specific platform.

In general, since it is currently not feasible or scalable to eliminate all device- or platform-specific implementations, we have instead strived to minimize them as much as possible while maximizing generic XD-AR functionality. Thus, our implementation of XD-AR fulfills the five requirements to the largest extent possible.

<sup>3</sup><https://github.com/mozilla/webxr-api> (accessed January 10, 2018).



## 8 PROOF-OF-CONCEPT APPLICATIONS

Finally, to test the overall feasibility and effectiveness of our XD-AR framework, we developed two proof of concept applications: Furniture Placement, and Shooter Game. In the following, we detail the methods we used to overcome the variety of technical and practical limitations that are inherent in AR application development nowadays by making use of XD-AR.

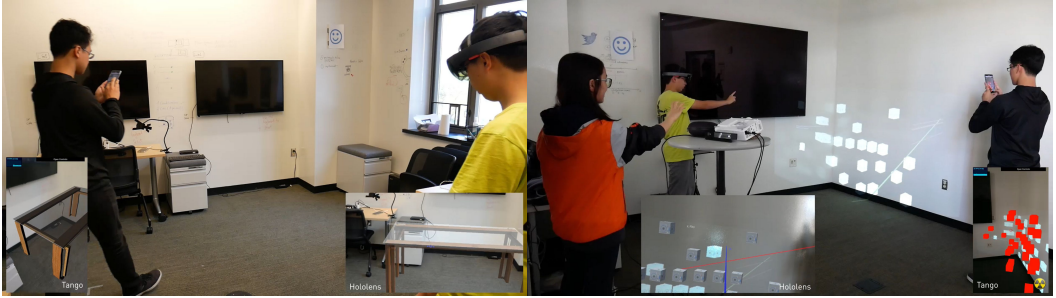


Fig. 6. The two proof of concept applications in use: Furniture Placement (left) and Shooter Game (right).

### 8.1 Technical Specifications

The Furniture Placement app was built using the following technology stacks: *a)* Tango, running a web application based on three.js within the native Chrome Browser on a ZenFone AR, and *b)* HoloLens, running a Unity application based on the MixedRealityToolkit library. Besides the above, the Shooter Game app additionally features *c)* projective AR, in terms of a Unity application based on the RoomAlive toolkit. To tie all of these platforms together and synchronize virtual objects and interactions across devices, we relied on our XD-AR implementation described in the previous section. The use of XD-AR as the backbone of our proof-of-concept applications helps to overcome the challenges identified in Section 2.

### 8.2 Furniture Placement

The Furniture Placement application allows all connected devices to place furniture and view and interact with furniture placed by other users (Figure 6). Users wearing a HoloLens can say the name of a piece of furniture, such as “table” and by air-tapping, the virtual object will appear in the world on the surface the HoloLens’s gaze cursor is pointing to. Similarly, a Tango user can select their choice of furniture from a drop-down menu displayed on their touchscreen, and then tap the screen to place it. A HoloLens user can gaze onto a piece of furniture, air-tap to pick it up, use their gaze to move the furniture elsewhere (constrained by the geometry of the walls and floor), and then air-tap again to place the furniture in the new position. Tango users can tap and drag furniture on their screen as well. All users share the same augmented view of the physical world, can move both their own objects as well as objects placed by other users, and see objects being moved in real time.

In this application, XD-AR takes care of *a)* abstracting from the different low-level events and virtual furniture implementations (and instead using high-level interactions such as “insert table”), as well as *b)* showing all objects at the same place in the physical environment on HoloLens and Tango phones in real time.

### 8.3 Shooter Game

The Shooter Game application allows users of HoloLens, Tango phones, and projective (RoomAlive) systems to play a game together in a shared space (Figure 6). The user playing with RoomAlive can

point at a virtual object and close their hand to shoot a laser which extends outwards from their shoulder to their hand (for more accurate targeting), and then out into the virtual world. Users of the Tango phone move their phone to point their laser, and fire by tapping a button on their screen. Users of HoloLens get to dual-wield lasers, as each hand can air-tap to shoot out its own laser. All lasers are color-coded by user, and every user can see every others user's laser beams. If a laser hits a virtual object, the object is destroyed and disappears from the game world for all users.

Using similar methods as in the Furniture Placement application, users can also place balls or cubes that float in the world, which act as targets for users to shoot at in the game. Any user can place an object in the world, and all users can see the object and shoot at it. Users can also grab hold of an item and move it elsewhere in the environment, so users can setup their own augmented shooting gallery if they like. As before, each device uses its own preferred modalities for interaction.

Again, XD-AR takes care of *a*) abstracting from the different low-level events (air-tap, touch, close hand) and virtual object implementations, as well as *b*) showing all objects in real time at the same place in the physical environment on HoloLens, Tango phones, as well as in RoomAlive.

## 9 DISCUSSION AND FUTURE WORK

After having shown the feasibility of XD-AR by implementing the introductory scenarios, in the following we discuss the limitations of our approach and future directions.

### 9.1 Progressive Enhancement in Augmented Reality

The current state of AR development is in many ways similar to the state of web development and mobile application development, where new features are constantly being introduced by a host of hardware and software vendors, and the majority of devices owned by end-users support only a fractional subset of the capabilities of the latest generation of top-end devices. Progressive enhancement may prove to be even more valuable in AR than in web development, as a growing variety of devices from multiple competing platforms (Android, iOS, Windows, and multi-OS web browsers) could make platform-agnostic application solutions more important than ever.

With an XD-AR framework in place, developers could use progressive enhancement to focus more on content and experience. Low-feature devices could access a limited but useful experience, while additional layers could allow more capable devices additional features, modalities, and performance.

Yet, it is currently not feasible to eliminate all platform-specific parts of such a framework due to a range of vendor-specific restrictions. For instance, ARCore and ARKit do not provide low-level sensor access and would require native implementations for this part. However, with XD-AR, we aim at minimizing these as far as possible.

### 9.2 Extending XD-AR With Additional Devices

Our XD-AR framework has intentionally been constructed to be extensible, allowing easy addition of new devices and platforms as they are released, such as Google's ARCore or Apple's ARKit technology. A new client only needs to be able to define a shared reference point in the physical world for means of calibration, and then communicate commands and object positions to XD-AR relative to this reference point. Adding support for ARCore, for example, is relatively straightforward. ARCore recently added automatic support for sharing World Anchors between devices, but this only works between Android and iOS devices. To integrate ARCore with other XD-AR clients, XD-AR's Coordinate Management module needs to be extended with ARCore's internal coordinate system and unit scale. The Session Management module also needs to add the ARCore device class to the registration process so that XD-AR can, e.g., translate global to device-specific coordinates. Since ARCore is quite similar to Tango, these are the only required changes. The Communication Management module only needs updating for device communication protocols not yet known to

the XD-AR server. Importantly, thanks to XD-AR, none of the other client implementations, such as for HoloLens and Tango, need updating in order to communicate with ARCore devices.

It is difficult to provide time and cost estimates for extending XD-AR with support for new devices and platforms. Our developers that implemented XD-AR's Tango support estimated it to be easy to add, e.g., ARCore support given the relative similarity of the two technologies. As mentioned, ARCore introduced Cloud Anchors for real-time synchronization across ARCore devices, which is akin to HoloLens's shared World Anchors.<sup>4</sup> We would, however, expect more effort for adding devices like *Meta 2* or *Magic Leap* that are based on proprietary platforms.

### 9.3 Alternative Implementation Approaches

In the process of developing the XD-AR implementation discussed here, a number of alternative approaches were considered along the way. Two significant strategies worth noting are: universal combined sensing, where all devices feed a central server that processes the sensor information and then responds back with the results for display; and unified implementation, where a single platform is developed that works similarly on all devices (reminiscent of the “write once, run anywhere” approach attempted by Java). Unfortunately, bandwidth and processing power required to integrate various sensor streams—including high-resolution video—are major limiting factors. Additionally, a variety of vendor-enforced platform restrictions render these approaches infeasible at this time. As described previously, our implementation focused on platform-agnostic components where possible, and used minimal device-specific implementation components when necessary.

Fiducial markers could potentially be integrated into an XD-AR implementation, especially for use as a shared world anchor for systems that do not support or perform well with markerless techniques. This approach could also allow simpler calibration of devices, as the shared world anchor is automatically recognized rather than manually defined. The downside with this approach is that, as shown in Table 1, many AR systems do not natively support markers, and thus many device-specific implementations would need to be expanded.

In terms of coordinate systems, we considered two primary alternatives: having each device translate its local coordinates into a unified coordinate system based on a shared reference point, or having each device report its coordinates in their native form and having the server maintain responsibility for converting coordinates back and forth between devices (the approach we used). Anyone considering the latter approach should pay attention to the additional cognitive overhead this can create in developing and debugging of the server code, especially in troubleshooting divergent device behavior and calibration issues.

## 10 RELATED WORK

For this work we draw upon two primary bodies of research: *augmented and mixed reality* and *cross-device interfaces*.

### 10.1 Augmented/Mixed Reality Research

AR has been researched for decades, with major survey papers covering the latest in the field extending from 1997 to 2015 [1, 2, 5, 6, 36, 39]. This research to date has necessarily been largely concerned with getting the core technologies to work together in a sufficiently robust way, so as to overcome the fundamental perceptual and human factors challenges [10, 20]. These are non-trivial issues, and involve dozens of open problems across multiple engineering disciplines.

<sup>4</sup><https://blog.google/products/google-vr/experience-augmented-reality-together-new-updates-arcore/> (accessed May 24, 2018).

The recent availability of high-fidelity see-through head-mounted displays, such as the HoloLens, coupled with a new push from smartphone makers to better support AR, means that for the first time we can begin to consider the new design and engineering challenges involved in making these systems work together. No currently identified work in augmented or mixed reality has addressed the challenges of how to engineer useful systems that will support a profusion of AR-enabled devices across a fractured ecosystem of vendors.

Projective AR is a unique category of AR approaches, as it converts the environment itself into a type of display. We make use of RoomAlive as a kind of representative example of what can be done with this technique, which allows network PCs to drive multiple projectors and Kinect devices to create an immersive room-filling AR experience [17]. In addition to entertainment uses, projective systems have been used to provide enhanced practical teaching methods by overlaying anatomy and live annotations over a student [14]. Projective AR has also been shown to have potential in mental health, such as being used to provide clinical treatment of phobias [4, 37]. To the best of our knowledge, there is no prior attempt at supporting AR interfaces that can be viewed using projective displays, such as RoomAlive, together with other types of AR devices.

## 10.2 Cross-Device Interfaces Research

XD-AR also adds to a long stream of research on cross-device interfaces. A number of studies have identified a strong need to facilitate tasks involving multiple devices [9, 28, 33]. Jokela *et al.* [16] provide a taxonomy of interactions and distinguish between resource lending, related parallel use, and unrelated parallel use. Display size, input capabilities, and level of multi-tasking support are the main factors in choosing which device to use in a specific situation. Despite these studies, support in current interfaces is still limited.

To support parallel use of multiple devices, a number of frameworks and tools have been proposed. For example, Conductor [13] demonstrates how to enable users to easily share information, chain tasks across devices, and manage sessions across multiple mobile devices. Panelrama [38], XDStudio [26], Weave [7] and WatchConnect [15] provide development and authoring tools to support specification and design of cross-device interfaces. In related research [25, 30], researchers have also studied desired cross-device designs with end-users and identified patterns of parallel use of smartwatches, phones, and tablets. Finally, a number of projects have focused on infrastructure and approaches range from simple messaging [29] to dynamically migrating interface distribution engines [12].

However, none of the prior work has attempted to support cross-device AR. The closest to our work are standardization movements, such as WebVR, which provide a layer of VR technology on top of existing web technologies. Similar efforts are planned around the idea of WebAR and Mozilla's WebXR. Open-source projects like AR.js and argon.js illustrate how to implement AR interfaces that can run on different mobile devices, but do not provide support for cross-device interactions to the extent provided by XD-AR.

## 11 CONCLUSION

The proliferation of new AR devices and technologies confronts designers and developers with a fragmented ecosystem. Therefore, it is crucial to gain a better understanding of the challenges and opportunities in AR. In this paper, we presented two main contributions.

First, we provided a taxonomy of AR frameworks that both supports developers in choosing the suitable technology for their needs, and allowed us to generate a comprehensive set of requirements for cross-device AR approaches.

Second, we introduced XD-AR, a cross-device augmented reality framework that was designed to study the issues necessary to be solved to provide designers and developers with the means to create innovative applications for a wide variety of platforms and devices.

To inform both the taxonomy and the novel framework, we *a)* identified two particularly challenging scenarios based on existing work; *b)* conducted a survey of AR designers, developers, and users; and *c)* analyzed the state of the art of AR frameworks.

We were able to show the feasibility of our solution by implementing the two challenging scenarios as proof of concept applications on top of XD-AR. They integrate HoloLens, Tango, and RoomAlive, and would be easily extensible to novel technologies like ARKit and ARCore. These applications allow users to, among other things, place furniture in a room using HoloLens and view it on a Tango phone at the exact same position in real time. From this new foundation, we can move towards a future where cross-device AR designers and developers will be able to focus more on the usability and ease of development of the next generation of applications.

## 12 ACKNOWLEDGMENTS

We thank our anonymous study participants. Brian Hall was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1256260.

## REFERENCES

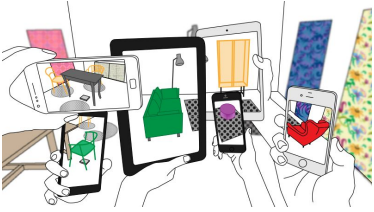
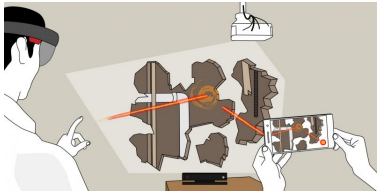
- [1] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. 2001. Recent advances in augmented reality. *IEEE Computer Graphics and Applications* 21, 6 (Nov 2001), 34–47.
- [2] Ronald T Azuma. 1997. A survey of augmented reality. *Presence: Teleoperators and virtual environments* 6, 4 (1997), 355–385.
- [3] Martin Bauer, Bernd Bruegge, Gudrun Klinker, Asa MacWilliams, Thomas Reicher, Stefan Riss, Christian Sandor, and Martin Wagner. 2001. Design of a component-based augmented reality framework. In *Proc. IEEE and ACM International Symposium on Augmented Reality*. 45–54.
- [4] Oliver Baus and Stéphane Bouchard. 2014. Moving from Virtual Reality Exposure-Based Therapy to Augmented Reality Exposure-Based Therapy: A Review. *Front. Hum. Neurosci.* 8, March (2014), 1–15.
- [5] Mark Billinghurst, Adrian Clark, and Gun Lee. 2015. A Survey of Augmented Reality. *Foundations and Trends in Human-Computer Interaction* 8, 2-3 (2015), 73–272.
- [6] Julie Carmigniani, Borko Furht, Marco Anisetti, Paolo Ceravolo, Ernesto Damiani, and Misa Ivkovic. 2011. Augmented reality technologies, systems and applications. *Multimedia Tools and Applications* 51, 1 (Jan 2011), 341–377.
- [7] Pei-Yu (Peggy) Chi and Yang Li. 2015. Weave: Scripting Cross-Device Wearable Interaction. In *Proc. CHI*. 3923–3932.
- [8] Francesco Clemente, Strahinja Dosen, Luca Lonini, Marko Markovic, Dario Farina, and Christian Cipriani. 2017. Humans can integrate augmented reality feedback in their sensorimotor control of a robotic hand. *IEEE Transactions on Human-Machine Systems* 47, 4 (2017), 583–589.
- [9] David Dearman and Jeffrey S. Pierce. 2008. “It’s on my other Computer!”: Computing with Multiple Devices. In *Proc. CHI*. 767–776.
- [10] David Drascic and Paul Milgram. 1996. Perceptual issues in augmented reality. In *SPIE The International Society For Optical Engineering*. 123–134.
- [11] Mark Fiala. 2004. ARTag Revision 1. A Fiducial Marker System Using Digital Techniques. *National Research Council Publication* 47419 (2004), 1–47.
- [12] Luca Frosini and Fabio Paternò. 2014. User Interface Distribution in Multi-Device and Multi-User Environments with Dynamically Migrating Engines. In *Proc. EICS*. 55–64.
- [13] Peter Hamilton and Daniel J. Wigdor. 2014. Conductor: Enabling and Understanding Cross-Device Interaction. In *Proc. CHI*. 2773–2782.
- [14] Thuong Hoang, Martin Reinoso, Zaher Joukhadar, Frank Vetere, and David Kelly. 2017. Augmented Studio: Projection Mapping on Moving Body for Physiotherapy Education. In *Proc. CHI*. 1419–1430.
- [15] Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proc. CHI*. 1247–1256.
- [16] Tero Jokela, Jarno Ojala, and Thomas Olsson. 2015. A Diary Study on Combining Multiple Information Devices in Everyday Activities and Tasks. In *Proc. CHI*. 3903–3912.

- [17] Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. 2014. RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-camera Units. In *Proc. UIST*. 637–644.
- [18] Seokmin Jung and Woontack Woo. 2004. UbiTrack: Infrared-based user Tracking System for indoor environment. *International Conference on Artificial Reality and Telexistence (ICAT04)* (2004), 1345–1278.
- [19] Oliver Beren Kaul and Michael Rohs. 2017. HapticHead: A Spherical Vibrotactile Grid Around the Head for 3D Guidance in Virtual and Augmented Reality. In *Proc. CHI*. 3729–3740.
- [20] M. A. Livingston. 2005. Evaluating human factors in augmented reality systems. *IEEE Computer Graphics and Applications* 25, 6 (Nov 2005), 6–9.
- [21] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. ACM, 197–206.
- [22] Matt Miesnieks. 2017. How is ARCore better than ARKit? (2017). <https://medium.com/super-ventures-blog/how-is-arcore-better-than-arkit-5223e6b3e79d>.
- [23] Matt Miesnieks. 2017. Why is ARKit better than the alternatives? (2017). <https://medium.com/super-ventures-blog/why-is-arkit-better-than-the-alternatives-af8871889d6a>.
- [24] Paul Milgram and Fumio Kishino. 1994. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems* 77, 12 (1994), 1321–1329.
- [25] Michael Nebeling and Anind K. Dey. 2016. XDBrowser: User-Defined Cross-Device Web Page Designs. In *Proc. CHI*. 5494–5505.
- [26] Michael Nebeling, Theano Mints, Maria Husmann, and Moira C. Norrie. 2014. Interactive Development of Cross-Device User Interfaces. In *Proc. CHI*. 2793–2802.
- [27] Michael Nebeling, Elena Teunissen, Maria Husmann, and Moira C. Norrie. 2014. XDKinect: Development Framework for Cross-Device Interaction using Kinect. In *Proc. EICS*. 2793–2802.
- [28] Antti Oulasvirta and Lauri Sumari. 2007. Mobile Kits and Laptop Trays: Managing Multiple Devices in Mobile Information Work. In *Proc. CHI*. 1127–1136.
- [29] Jeffrey S. Pierce and Jeffrey Nichols. 2008. An infrastructure for extending applications’ user experiences across multiple personal devices. In *Proc. UIST*. 101–110.
- [30] 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 *Proc. CHI*. 3913–3922.
- [31] Gerhard Reitmayr and Dieter Schmalstieg. 2005. OpenTracker: A flexible software design for three-dimensional interaction. *Virtual reality* 9, 1 (2005), 79–92.
- [32] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. Sage.
- [33] Stephanie Santosa and Daniel Wigdor. 2013. A Field Study of Multi-Device Workflows in Distributed Workspaces. In *Proc. UbiComp*. 63–72.
- [34] Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. 2014. In-air gestures around unmodified mobile devices. In *Proc. UIST*. 319–329.
- [35] Sebastian Thrun and John J Leonard. 2008. Simultaneous localization and mapping. In *Springer handbook of robotics*. Springer, 871–889.
- [36] DWF Van Krevelen and Ronald Poelman. 2010. A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality* 9, 2 (2010), 1–20.
- [37] Maja Wrzesien, Cristina Botella, Juana Bretón-López, Eva del Río González, Jean-Marie Burkhardt, Mariano Alcañiz, and María Ángeles Pérez-Ara. 2015. Treating small animal phobias using a projective-augmented reality system: A single-case study. *Computers in Human Behavior* 49 (2015), 343–353.
- [38] Jishuo Yang and Daniel Wigdor. 2014. Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In *Proc. CHI*. 2783–2792.
- [39] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. 2008. Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR. In *Proc. ISMAR*. 193–202.





<p><b>BACKGROUND</b> What is your background?</p> <p><input type="radio"/> Academia</p> <p><input type="radio"/> Industry</p> <p><input type="radio"/> Both</p> <p><input type="radio"/> Other _____</p> <p>.....</p> <p><b>OCCUPATION</b> What is your current occupation?</p> <p>_____</p> <p>.....</p> <p><b>GENDER</b> What is your gender?</p> <p><input type="radio"/> Prefer not to say</p> <p><input type="radio"/> Male</p> <p><input type="radio"/> Female</p> <p><input type="radio"/> Other _____</p> <p>.....</p> <p><b>AGE</b> What is your year of birth?</p> <p>_____</p> <p>.....</p> <p>End of Block: Background Questions</p> <p>Start of Block: Main Questions</p> <p>Page 5 of 12</p>	<p>AR Software and Technologies (Part 2/3)</p> <p>.....</p> <p><b>NOTE</b> Please answer the following questions from both the AR developer/designer and the AR user's perspective.</p> <p>.....</p> <p><b>AR Frameworks</b></p> <p>In the following, we understand the term "AR framework" very broadly as technical and/or design support to create an AR application. Examples include ARToolKit, Vuforia, Wikitude, ARKit, Tango, HoloToolkit, Argon, etc.</p> <p>.....</p> <p><b>FRAMEWORK</b> What kind of AR frameworks have you worked with?</p> <p>_____</p> <p>_____</p> <p>.....</p> <p><b>COMPROMISE</b> In any of your projects, have you had to significantly change or even drop a planned feature due to a lack of support or incompatibility in the framework? <b>Please explain.</b></p> <p>_____</p> <p>_____</p> <p>.....</p> <p>Page Break</p> <p><b>Pains/Challenges</b></p> <p>.....</p> <p>Page 6 of 12</p>
<p>Q131 AR Software and Technologies (Part 2/3)</p> <p>.....</p> <p><b>APP PROS</b> Based on the AR applications you have used, what is working relatively well?</p> <p>_____</p> <p>_____</p> <p>.....</p> <p><b>APP CONS</b> What is a remaining major challenge/not working well enough so far?</p> <p>_____</p> <p>_____</p> <p>.....</p> <p><b>FRAMEWORK PROS</b> When creating an AR application, what is relatively well supported in existing AR frameworks?</p> <p>_____</p> <p>_____</p> <p>.....</p> <p><b>FRAMEWORK CONS</b> What is a major remaining challenge?</p> <p>_____</p> <p>_____</p> <p>.....</p> <p>Page Break</p> <p>Technical and Design Limitations</p> <p>Page 7 of 12</p>	<p>.....</p> <p><b>LIMITATION</b> For the kinds of AR applications that you have created or used, what was a major technical or design limitation with existing AR frameworks?</p> <p>_____</p> <p>_____</p> <p>.....</p> <p><b>SOLUTION</b> How did you try to resolve this limitation as an AR developer/designer?</p> <p>_____</p> <p>_____</p> <p>.....</p> <p>Page Break</p> <p>Challenging AR Applications (Part 3/3)</p> <p>.....</p> <p><b>CHALLENGING APP</b> What is an AR application that you would find very useful but that you believe is challenging to create?</p> <p>_____</p> <p>_____</p> <p>.....</p> <p><b>REASON</b> What do you think are the main reasons that application is not available yet? (e.g., conceptual, technical, business, etc. reasons)</p> <p>_____</p> <p>_____</p> <p>.....</p> <p>End of Block: Main Questions</p> <p>Page 8 of 12</p>

<div><div>Start of Block: Applications</div><div>Q54 Challenging AR Applications (Part 3/3)</div><div><p>In the following, you will see illustrations of two AR applications. Each illustration details various technical or design aspects that are challenging to support with current AR software and technologies.</p><p>We would like you to first identify the challenges you see, and then explain how you think they can be addressed.</p></div><div>Page Break</div><div><div>AR Furniture Placement</div><div>All family members can use their own mobile devices to place virtual models of new furniture in the room.</div><div></div></div><div>Page 9 of 12</div></div>	<div><div>CHALLENGE1 What are the two most challenging aspects of this AR application? Please explain.</div><div></div><div>SOLUTION1 Describe how you think the above challenges can be addressed (e.g., based on examples of existing AR software and technologies).</div><div></div><div>Page Break</div><div><div>AR Shooter Game</div><div>Multiple users can play a laser shooting game together with AR headsets (left), projection (middle), or handhelds (right) using mid-air or touch gestures.</div><div></div></div><div>Page 10 of 12</div></div>
<div><div>CHALLENGE2 What are the two most challenging aspects of this AR application? Please explain.</div><div></div><div>SOLUTION2 Describe how you think the above challenges can be addressed (e.g., based on examples of existing AR software and technologies).</div><div></div><div>Page Break</div><div><div>Ideal AR Framework</div><div></div><div>IDEAL When creating an AR application, what kind of support would you expect from an Ideal AR framework?</div><div></div></div><div>Page 11 of 12</div></div>	<div><div>RANKING How would you rank order the following AR framework features from most to least important?</div><div><div><input type="checkbox"/> Motion tracking (traditional template square, custom 2D/3D physical object, text, etc.)</div><div><input type="checkbox"/> Environment mapping (plane detection, complex geometry, etc.)</div><div><input type="checkbox"/> Graphics (2D/3D visualization, animation, etc.)</div><div><input type="checkbox"/> Explicit interactions / user input (gestures, voice commands, virtual buttons &amp; controls, etc.)</div><div><input type="checkbox"/> Non-visual support (audio, haptics, etc.)</div><div><input type="checkbox"/> Multi-user / multi-device / multi-platform support</div></div><div>Page Break</div><div><div>Future of AR</div><div></div><div>FUTURE Where do you see AR in 5 years / 10 years from now? Why?</div><div></div><div>End of Block: Applications</div><div>Start of Block: Exit</div><div>END Thank you very much for your time!</div><div></div><div><div>EMAIL Please enter your email address should you wish to enter our lottery for \$20 Amazon gift cards among all registered participants.</div><div></div></div><div>End of Block: Exit</div><div>Page 12 of 12</div></div></div>

Received February 2018; revised April 2018; accepted June 2018