The Trouble with Augmented Reality/Virtual Reality Authoring Tools

Michael Nebeling* Maximilian Speicher†

University of Michigan School of Information
https://mi2lab.com

ABSTRACT

There are many technical and design challenges in creating new, usable and useful AR/VR applications. In particular, non-technical designers and end-users are facing a lack of tools to quickly and easily prototype and test new AR/VR user experiences. We review and classify existing AR/VR authoring tools and characterize three primary issues with these tools based on our review and a case study. To address the issues, we discuss two new tools we designed with support for rapid prototyping of new AR/VR content and gesture-based interactions geared towards designers without technical knowledge in gesture recognition, 3D modeling, and programming.

Keywords: augmented reality, virtual reality, authoring, design, rapid prototyping, 3D modeling, gestures, Wizard of Oz.

Index Terms: Human-centered computing—Interaction paradigms—Mixed / augmented reality

1 INTRODUCTION

Today's AR/VR applications are mostly built by tech-savvy developers. Significant technical skill and programming experience is required to create AR/VR experiences. While tools like Unity and A-Frame have in many ways become the "standard for AR/VR," they still provide a high threshold for non-technical designers and are inaccessible to less experienced end-users. There is a new class of tools for creating basic AR/VR experiences, allowing users to choose from pre-made 3D models and existing scripts for animation and interactivity. However, these tools usually cover only a very limited spectrum of the AR/VR design space and still require programming for more advanced application logic.

In this position paper, we classify existing authoring tools relevant to AR/VR, identify five classes of tools (Fig. 1), and characterize the main issues we see with how the tool landscape has been evolving. Both authors have a track record of research on interactive technologies with a more recent focus on AR/VR [20, 28–30]. For example, they created ProtoAR [20], a tool designed with the vision of making AR/VR prototyping as easy and versatile as paper prototyping, and GestureWiz [30], a *Wizard of Oz* gesture prototyping environment. The second author also contributed to the design and development of HoloBuilder [31] from 2015 to 2017. When he joined the company, the original idea was to create a "PowerPoint for AR," enabling users without specific design and development skills to create AR experiences. For the future, we envision tools as simple yet powerful as PowerPoint or Keynote leveling the playing field for AR/VR.

2 FIVE CLASSES OF AR/VR AUTHORING TOOLS

Interaction design [3, 19, 23] has evolved into a comprehensive, iterative process with four main activities: establish user needs, develop alternative designs, build interactive prototypes, and evaluate prototypes. Any remaining difficult or new critical areas will be subjects of future iterations. A significant part of the process is prototyping alternative designs, typically with several iterations on paper to shape the concept [1, 22, 26], then moving to digital tools to increase the level of fidelity and generate new design insights [16, 17]. For AR, however, paper prototyping was recently found too limiting [7, 20], motivating the development of new techniques that support physical prototyping but integrate well with digital tools.

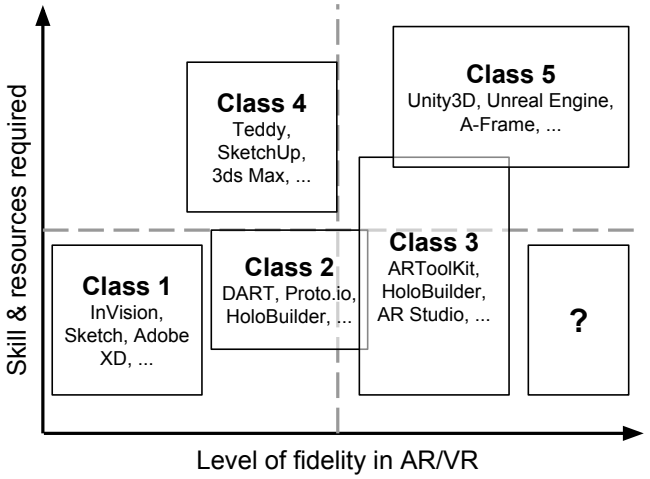

Figure 1: AR/VR authoring tools classified along *(x)* the possible level of fidelity in AR/VR, and *(y)* skill & resources (e.g., 3D models, 360 photos, hardware) required. An "optimal" class of tools would be on the far right and remain low. Tools that combine the principles of our ProtoAR [20] and GestureWiz [30] could be examples for this class.

There is an extensive number of digital tools available for creating interactive prototypes, but across all tools, support for AR/VR content and interactive behavior is limited and scattered. The landscape of existing digital tools can be grouped into five classes of tools.

2.1 First Class: Basic Mobile Screens & Interactions

The first class consists of tools for mobile and web designers such as InVision, Sketch, and Adobe XD. Many tools in this class have a concept of multiple screens, support defining active regions in these screens that listen to mouse and touch input to trigger menus or transition to other screens, and enable mobile app previews on smartphones and tablets. While this is sufficient for traditional mobile interfaces, tools in this class lack support for AR/VR content and interaction as they do not interface with AR/VR technologies.

2.2 Second Class: Basic AR/VR Scenes & Interactions

The second class consists of AR/VR counterparts of tools in the first class supporting basic forms of AR/VR content and interactive behavior. An early example in research is DART [18], extending Macromedia Director with scripts for animating 3D avatars useful for prototyping storytelling AR experiences. Recent examples, Proto.io, Halo, and HoloBuilder, aim to be InVision-like tools for AR/VR. Users can upload 360-degree photos to create immersive scenes and define basic interactive behaviors via image maps with anchors that trigger menus or load other scenes. While VR previews are

*e-mail: nebeling@umich.edu
†e-mail: mspeiche@umich.edu

supported with Google Cardboard, AR spatial trackers required for composite views of virtual content and real-world objects are not.

2.3 Third Class: AR/VR Focused Interactions

The third class consists of tools focused on AR camera-based interactions. Early examples from research include ARToolKit [11], Tiles [21], Studierstube [24] and ComposAR [25]. HoloBuilder as well falls into this category since it features marker-based AR functionality. Recent solutions like Facebook's AR Studio or Snapchat's Lens Studio allow users to select from libraries of trackers (face, hand, plane), 2D/3D objects, video masks, and animations to create interactive, shareable camera effects responding to people and objects in their surroundings. There is support for visual programming to add basic animations, logic, and interactivity, but many interface concepts common to tools from the other classes, such as screens, menus, and transitions, are absent.

2.4 Fourth Class: 3D Content

The fourth class consists of 3D content creation tools like Teddy [8] and Lift-Off [10] created by researchers, Google's SketchUp and Blocks, as well as Autodesk's 3ds Max and Maya. Tools in this class support the creation of new 3D objects via digital 3D modeling, animation, and simulation techniques unique to this class of tools. 3D objects can be exported in common 3D file formats supported by other classes of tools except the first. Further, 3D content sharing platforms like Google's 3D Warehouse and Poly have significantly grown in recent years because of the increased need for 3D content due to emerging new technologies for 3D printing as well as AR/VR.

2.5 Fifth Class: 3D Games & Applications

The fifth and last class consists of tools like Unity, Unreal, and A-Frame, which are comprehensive game and application development platforms. Tools in this class come with a visual editor in which scenes with 3D objects can be visually specified and assets such as 3D models, textures, and sounds can be imported and managed. Although most AR/VR device platforms integrate with Unity and Unreal, there is relatively little support specific to AR/VR in them. A-Frame is a relatively new and increasingly popular AR/VR application development layer on top of HTML, CSS, and three.js. Still, new AR/VR content needs to be created with the aforementioned 3D tools and the specification of interactive behavior for AR/VR requires extensive knowledge in 3D graphics and programming languages such as C# or JavaScript. As these constitute primary barriers for non-technical designers [2, 18], this class of tools is geared towards programmers. While they enable high-fidelity interactive AR/VR applications, prototyping alternative designs with them is a lot harder, much more time-consuming, thus significantly more expensive compared to tools in the other classes.

3 The Trouble with Existing Tools

Our brief review of existing digital tools has highlighted three main problems: *(1)* a massive tool landscape, *(2)* most AR/VR projects require tools from multiple classes, and *(3)* significant gaps of tools both within and between classes of tools.

3.1 Massive Tool Landscape

First, there is a complex tool landscape and the sheer number of tools makes it hard for new designers. The significant differences between tools in terms of support for AR/VR content and interactive behavior and the high speed with which the digital tool landscape is evolving still make it difficult even for experienced designers and developers. A significant part of AR/VR applications is 3D content. In the current tool landscape, however, this requires a special class of tools orthogonal to other classes and a completely different skill set and training of designers in 3D modeling, animation, and simulation.

3.2 Design Processes are Unique Patchworks

Second, every new AR application that is being built essentially requires building a unique tool chain as well, where selection of tools highly depends on technical requirements of the next design stage. Developers and engineers can work with tools at the high end of the tool chain (choosing tools like Unity) because they have the training and experience. Yet, these are usually out of reach for non-technical designers. Even worse is that users with less training often end up working with an even larger patchwork of tools, to try to work around their challenges with more professional tools [12].

3.3 Significant Gaps Within & Between Tools

Third, to move an AR/VR application from early design concepts to interactive prototypes, developing alternative designs and design iterations typically require creating different parts and versions of the application with different classes of tools. However, the tool chain is generally optimized in the upwards direction (e.g., tools like Halo export to Unity but not vice-versa). This makes design iterations especially tricky and expensive if they require going back to a previous tool from another class.

4 Case Study

As mentioned earlier, the second author contributed to HoloBuilder, a PowerPoint-inspired tool situated in the second and third classes above, for two years and gained first-hand experience with designing and promoting a platform intended to enable anyone to create AR applications [31]. AR projects in HoloBuilder are based on custom images or point clouds as markers while different states of an app are represented by different "slides", each of which can feature zero or one markers (Fig. 2). Moreover, users are provided with a selection of predefined 3D models, such as arrows, cubes, and spheres. The 3D models added to a "slide" can be enhanced with basic animations and click events that either trigger a transition to a different "slide" or an animation. During his time at HoloBuilder, the second author made three key observations analogous to the problems above.

Figure 2: An AR scene with custom 2D marker in HoloBuilder

First, the vast majority of end-users saw HoloBuilder as a tool for creating virtual tours from 360-degree photos rather than AR content. Many made use of a specific kind of "slide" where a 360-degree photo could be uploaded as the "marker", thus technically still creating an AR scene, but with the camera positioned at the center of a sphere having the 360-degree photo as its inner texture. Different from these end-users, the actual AR capabilities sparked more interest in professional users from industry, most likely due to a combination of higher application requirements and more specific AR use cases. Their AR projects were, however, still marker-based and required a separate mobile app, while 360-degree projects could be viewed directly in the browser. These are examples of confusion about the type of tool *(Problem 1)* and of gaps within the tool

(Problem 3) since these two use cases are disjunct and have little common ground besides the technical basis.

Second, it became evident fairly quickly that users wanted to work with more advanced 3D models than the predefined set that came with the platform, which called for integration with external libraries of 3D models. While users could search for 3D models directly from HoloBuilder, this integration, due to technical restrictions, still required them to download the model in one of the supported formats and then manually drag and drop the downloaded file. This again illustrates the range of tools required *(Problem 1)*, each of which takes care of something else as well as the gap between such tools *(Problem 3)*, which leads to suboptimal integrations.

Third, due to the abundance of 360-degree virtual tours, HoloBuilder's focus shifted from AR to VR projects that could be viewed in, e.g., Google Cardboard. This also led to a strong focus on click interactions, using gaze and dwell typical for VR with Cardboard, ruling out more advanced interactions such as dragging objects in AR. While dragging, as a workaround, could still be partly simulated with, e.g., a click triggering a move animation, it was not possible to design more complex interactions required for many AR applications. Thus, even if a user created a complete AR scene in HoloBuilder, they would still need to use a patchwork of tools for designing interactions beyond clicks (*Problem 2*).

5 *So, what's a better tool?*

AR/VR interfaces are rich in content and afford multiple modalities, including touch, gesture, and speech. The above review of the tool landscape has revealed two key problems: *(1)* creating content for AR/VR interfaces remains difficult [18], and *(2)* specifying interactive behavior requires significant programming or model training [32]. The first problem can be traced back to limited support for early-stage prototyping using physical materials [7]. The second problem is due to the complexity of algorithms required for mobile device tracking [13] and gesture recognition [27], where there are few tools geared towards non-technical designers [9, 33]. In view of these problems, it is important to ask what would make a better tool? Below we characterize the tool support we envision for future tools addressing both problems, before presenting ProtoAR [20] and GestureWiz [30] as examples from our own research.

5.1 Inspiration from DART

The closest to the new tools envisioned by the authors is DART [18]. It shares the authors' goal of leveraging interaction designers existing skills and strengths. Designers use storyboards to organize sketches and images in sequence for the purpose of pre-visualizing animated and interactive content. To meet this need of designers, DART enabled rapid transition from 2D storyboards to 3D animatic actors as placeholders for final 3D content creation, allowing designers to explore interactive stories for new AR experiences. Through integration of Macromedia Director and AR technologies, designers could visually specify, rather than program, relationships between the physical and virtual worlds (e.g., start animation when specified position of the virtual camera is reached, physical object marker tracked, or timer elapsed).

5.2 Overcoming Limitations with Wizard of Oz

However, unlike the new tools envisioned by the authors, existing tools are still bounded by the limitations of AR tracking and interaction approaches [15, 34]. There are two important hard problems.

First, device motion tracking required for devices' internal sensors to understand and track the position relative to the world is still limited. Marker-based and marker-less tracking approaches can be distinguished. The first approach still requires instrumentation of the environment with fiducial markers. The second approach is relatively limited in terms of environmental understanding (e.g., ARKit/ARCore can only detect simple geometry like horizontal and vertical planes).

Second, while a lot of the interaction in AR is implicit (e.g., device motions, user's gaze), support for explicit interactions like touch, gesture, and speech is limited. Extensive research has produced many new tools and libraries focused on gesture recognition. However, most require instrumentation of the environment with external sensors like Kinect or significant time to train the underlying models from sample gestures provided by designers or users [27, 32].

Ultimately, these shortcomings limit designers' ability to rapidly prototype, and users' ability to fully experience, novel AR interfaces. In user interface research, Wizard of Oz [4] is a common technique to circumvent system limitations, by having an experimenter (the "wizard") simulate the behavior of a fully working system. While, in principle, possible for prototyping AR experiences [5, 6], this is hard because of the many degrees of freedom with AR interactions. Having a single wizard simulate all aspects of an AR system is not feasible [18] and using multiple wizards requires extensive training and coordination [14].

5.3 Two Examples from the Authors

5.3.1 ProtoAR: Quick & Easy Capture of AR Content

At CHI 2018, the first author presented ProtoAR [20], a tool he created to support prototyping of mobile AR apps by crafting the main screens and AR overlays from paper sketches and quasi-3D objects from 360-degree captures of Play-Doh models (Fig. 3). The project used a series of student design jams around IKEA's furniture placement AR app called Place. Students started on paper sketching screens and user flow, then used Play-Doh to model miniature versions of furniture they wanted to place, and finally made use of ProtoAR's capture tools to digitize these physical materials and see AR views on smartphones. With ProtoAR, students with no training in 3D graphics and programming were able to generate low-fidelity versions of the IKEA Place AR app in less than 90 minutes.

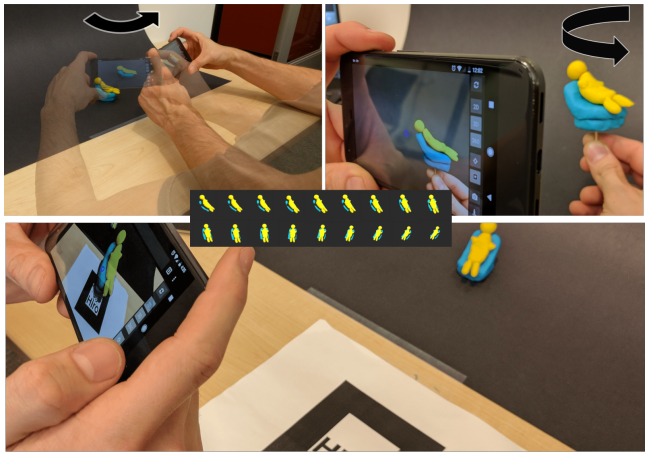

Figure 3: ProtoAR's [20] 360-degree capture of Play-Doh model (top); captured quasi-3D object (middle); marker-based AR preview (bottom)

5.3.2 GestureWiz: Prototyping Gesture Interactions for AR

A second tool the authors also presented at CHI 2018 is GestureWiz [30]. Inspired by Wobbrock et al.'s $1 recognizer's simple and flexible design, it provides a rapid prototyping environment to designers with an integrated solution for gesture definition, conflict checking, and real-time recognition by employing human recognizers in a Wizard of Oz manner (Fig. 4). GestureWiz was designed based on a series of online experiments and user studies in the authors' lab. In one study, 12 participants worked in pairs to co-design and test a novel gesture set. Part of the study required them to split

up and assume the roles of user and wizard to demonstrate and recognize gestures, respectively. GestureWiz implements techniques to manage complex gesture sets by coordinating multiple wizards via live streams, and achieves reasonable accuracy and latency for prototyping purposes. GestureWiz was also shown to support a variety of gesture-based interfaces from the literature that previously required complex system implementations. With GestureWiz, pairs of users and wizards were able to co-design and test a gesture-controlled slideshow prototype in less than 45 minutes.

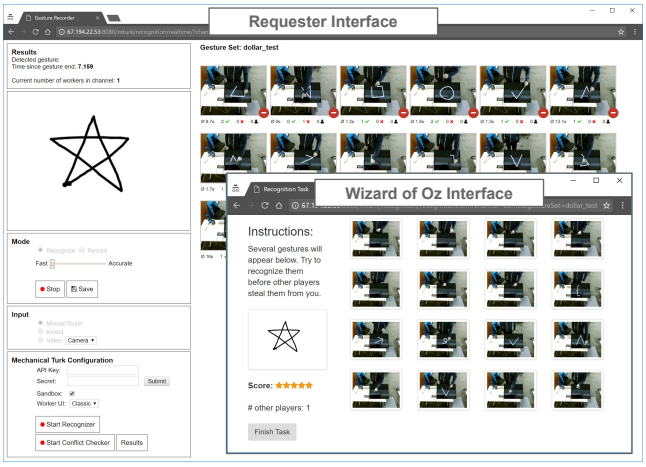

Figure 4: GestureWiz's rapid prototyping interfaces for recording (Requester Interface) and recognition (Wizard of Oz Interface) of arbitrary and potentially multi-modal gesture sets, e.g., single-stroke, multi-stroke, and mid-air 3D (adapted from [30])

6 Conclusion

We presented five classes of AR/VR authoring tools that differ in the levels of fidelity in AR/VR and skill and resources required. The rapidly growing landscape of tools leads to a number of problems for non-technical designers when it comes to creating new AR/VR experiences. For instance, tools that enable designing AR/VR scenes without significant programming knowledge largely do not support 3D modeling or explicit interaction. Inspired by DART [18] and $1 recognizer [33], we started to explore two new tools to overcome these limitations: ProtoAR [20], allowing designers to quickly and easily create 3D models from physical props and integrate them into mobile AR apps, and GestureWiz [30], which leverages Wizard of Oz to provide advanced gesture recognition without model training or programming.

However, there are still many challenges in creating better AR/VR authoring tools. ProtoAR only supports basic AR content and camera interactions. GestureWiz only uses Wizard of Oz for gesture recognition, and relies on external cameras. More complex AR/VR interfaces, however, can involve lots of interactive 3D objects reacting to both users' explicit interactions (via touch, gesture, speech) and implicit interactions (via device camera, inertial sensors, external sensors). These are challenges the authors are currently addressing by integrating the tools and developing new techniques as extensions of ProtoAR and GestureWiz with the ultimate goal of providing powerful examples able to fill the gap marked by "?" in Figure 1.

Code and Data

ProtoAR will be made available to interested workshop participants at `https://protoar.com`. The web site will provide access to the source code, example applications, and instructional materials. We also made code and data of GestureWiz available at `https://github.com/mi2lab/gesturewiz`.

Acknowledgments

Thanks to the co-authors of the ProtoAR [20] paper, Janet Nebeling, Ao Yu, and Rob Rumble.

References

[1] B. Buxton. *Sketching user experiences: getting the design right and the right design*. Morgan Kaufmann, 2010.

[2] M. Conway, S. Audia, T. Burnette, D. Cosgrove, K. Christiansen, R. Deline, J. Durbin, R. Gossweiler, S. Koga, C. Long, B. Mallory, S. Miale, K. Monkaitis, J. Patten, J. Pierce, J. Shochet, D. Staack, B. Stearns, R. Stoakley, C. Sturgill, J. Viega, J. White, G. Williams, and R. Pausch. Alice: Lessons learned from building a 3d system for novices. In *Proc. CHI*, 2000.

[3] A. Cooper, R. Reimann, D. Cronin, and C. Noessel. *About face: the essentials of interaction design*. John Wiley & Sons, 2014.

[4] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of oz studies: why and how. In *Proc. IUI*, pp. 193–200, 1993.

[5] S. Dow, J. Lee, C. Oezbek, B. MacIntyre, J. D. Bolter, and M. Gandy. Wizard of oz interfaces for mixed reality applications. In *Proc. CHI Extended Abstracts*, 2005.

[6] S. Dow, B. MacIntyre, J. Lee, C. Oezbek, J. D. Bolter, and M. Gandy. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 4(4):18–26, 2005.

[7] A. J. Hunsucker, K. McClinton, J. Wang, and E. Stolterman. Augmented Reality Prototyping For Interaction Design Students. In *Proc. CHI Extended Abstracts*, 2017.

[8] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proc. SIGGRAPH*, pp. 409–416, 1999.

[9] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. W. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. UIST*, pp. 559–568, 2011.

[10] B. Jackson and D. F. Keefe. Lift-off: Using reference imagery and freehand sketching to create 3d models in VR. *TVCG*, 22(4), 2016.

[11] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proc. IWAR*, pp. 85–94, 1999.

[12] G. Kim. Early Strategies in Context: Adobe Photoshop Lightroom. In *Proc. CHI Extended Abstracts)*, 2007.

[13] G. Klein and D. W. Murray. Parallel tracking and mapping on a camera phone. In *Proc. ISMAR*, 2009.

[14] B. Koleva, I. Taylor, S. Benford, M. Fraser, C. Greenhalgh, H. Schnädelbach, D. vom Lehn, C. Heath, J. Row-Farr, and M. Adams. Orchestrating a mixed reality performance. In *Proc. CHI*, 2001.

[15] E. Kruijff, J. E. Swan II, and S. Feiner. Perceptual issues in augmented reality revisited. In *Proc. ISMAR*, pp. 3–12, 2010.

[16] Y. Lim, E. Stolterman, and J. D. Tenenberg. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Trans. Comput.-Hum. Interact.*, 15(2):7:1–7:27, 2008.

[17] L. Liu and P. Khooshabeh. Paper or interactive?: a study of prototyping techniques for ubiquitous computing environments. In *Proc. CHI Extended Abstracts*, pp. 1030–1031, 2003.

[18] B. MacIntyre, M. Gandy, S. Dow, and J. D. Bolter. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proc. UIST*, 2004.

[19] B. Moggridge and B. Atkinson. *Designing interactions*, vol. 17. MIT press Cambridge, MA, 2007.

[20] M. Nebeling, J. Nebeling, A. Yu, and R. Rumble. ProtoAR: Rapid Physical-Digital Prototyping of Mobile Augmented Reality Applications. In *Proc. CHI*, 2018.

[21] I. Poupyrev, D. S. Tan, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsutani. Tiles: A mixed reality authoring interface. In *Proc. INTERACT*, pp. 334–341, 2001.

[22] M. Rettig. Prototyping for tiny fingers. *Commun. ACM*, 37(4), 1994.

[23] Y. Rogers, H. Sharp, and J. Preece. *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2011.

[24] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnaçao, M. Gervautz, and W. Purgathofer. The studierstube aug-

mented reality project. *Presence: Teleoperators & Virtual Environments*, 11(1):33–54, 2002.

[25] H. Seichter, J. Looser, and M. Billinghurst. Composar: An intuitive tool for authoring AR applications. In *Proc. ISMAR*, 2008.

[26] C. Snyder. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann, 2003.

[27] J. Song, G. Sörös, F. Pece, S. R. Fanello, S. Izadi, C. Keskin, and O. Hilliges. In-air gestures around unmodified mobile devices. In *Proc. UIST*, 2014.

[28] M. Speicher, J. Cao, A. Yu, H. Zhang, and M. Nebeling. 360anywhere: Mobile ad-hoc collaboration in any environment using 360 video and augmented reality. *PACMHCI*, 2:9:1–9:20, 2018.

[29] M. Speicher, B. D. Hall, A. Yu, B. Zhang, H. Zhang, J. Nebeling, and M. Nebeling. XD-AR: challenges and opportunities in cross-device augmented reality application development. *PACMHCI*, 2:7:1–7:24, 2018.

[30] M. Speicher and M. Nebeling. Gesturewiz: A human-powered gesture design environment for user interface prototypes. In *Proc. CHI*, 2018.

[31] M. Speicher, K. Tenhaft, S. Heinen, and H. Handorf. Enabling industry 4.0 with holobuilder. In *Proc. GI*, 2015.

[32] E. M. Taranta II, A. Samiei, M. Maghoumi, P. Khaloo, C. R. Pittman, and J. J. LaViola Jr. Jackknife: A reliable recognizer with few samples and many modalities. In *Proc. CHI*, 2017.

[33] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proc. UIST*, 2007.

[34] F. Zhou, H. B. Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *Proc. ISMAR*, pp. 193–202, 2008.