How to Safely Augment Reality: Challenges and Directions

Kiron Lebeck, Tadayoshi Kohno, Franziska Roesner
University of Washington
{kklebeck, yoshi, franzi}@cs.washington.edu

ABSTRACT

Augmented reality (AR) technologies, such as those in head-mounted displays like Microsoft HoloLens or in automotive windshields, are poised to change how people interact with their devices and the physical world. Though researchers have begun considering the security, privacy, and safety issues raised by these technologies, to date such efforts have focused on *input*, i.e., how to limit the amount of private information to which AR applications receive access. In this work, we focus on the challenge of *output* management: how can an AR operating system allow multiple concurrently running applications to safely augment the user's view of the world? That is, how can the OS prevent apps from (for example) interfering with content displayed by other apps or the user's perception of critical real-world context, while still allowing them sufficient flexibility to implement rich, immersive AR scenarios? We explore the design space for the management of visual AR output, propose a design that balances OS control with application flexibility, and lay out the research directions raised and enabled by this proposal.

1. INTRODUCTION

Augmented reality (AR) technologies redefine the boundaries between physical and virtual worlds. Rather than presenting virtual content distinct from the physical world, AR applications integrate virtual content "into" the physical world by collecting sensory information about the user's surroundings and overlaying virtual augmentations on the user's view of the real world. These technologies are unique in their potential for immersive mixed reality (i.e., combined physical and virtual) experiences, with applications ranging from assistive technologies to driving aids and home entertainment. Moreover, sophisticated AR applications that run on immersive platforms like Head Mounted Displays (HMDs) and cars are not hypothetical — industry efforts by companies like Microsoft and BMW have demonstrated that these technologies are real and on the horizon [3, 11, 21].

Today's typical AR applications run on conventional devices (e.g., smartphones) in their own isolated contexts. For example, the smartphone application WordLens[1] detects and translates text in real time using the phone's camera. However, as AR platforms continue to evolve and move from mobile phones to more immersive devices such as Microsoft's HoloLens [11], they can provide richer experiences by running *multiple* applications that simultaneously augment a *shared* environment. For example, a HoloLens user may wish to play an AR video game while simultaneously running Skype in the background (applications shown in Figure 1).

While AR applications have diverse potential, they also raise a number of challenging security, privacy, and safety issues [16]. Prior research efforts (e.g., [9, 17]) have focused largely on addressing *input* privacy issues that stem from applications receiving unrestricted sensor access, such as the ability for applications to record videos of a user's sensitive possessions or audio of private conversations. However, little work has looked at the risks and challenges presented by unregulated, and possibly malicious, application *outputs* — for example, a buggy or malicious car application could display distracting images to interfere with the driver's focus. While ensuring the safety and security of an individual application's output is already challenging (as we will see), these issues become particularly salient with the increasingly complex interactions of multiple applications augmenting a shared environment, where applications may attempt to display overlapping content, either coincidentally or maliciously (e.g., to mount a denial-of-service attack).

In this work, we begin to address the safety and security of AR visual output. We lay out design alternatives for visual output models, and in doing so we surface challenges in balancing the flexibility of application functionality with the system's ability to control and mitigate risks from application outputs. We provide initial directions for overcoming these challenges by rethinking the role of the operating system in managing visual AR outputs. Our exploration raises directions and guidelines for future work in this space.

2. BACKGROUND AND MOTIVATION

We begin with several example AR scenarios, followed by the risks and challenges that these scenarios may raise.

2.1 Example AR Systems: The Opportunities

We first present two illustrative case studies that capture the diverse potential of AR technology: automotive AR and HMDs. We focus on these scenarios, rather than mobile phone-based AR, because they are immersive, intended for continuous use, and naturally cater to multiple concurrently running applications. However, mobile phone-based AR scenarios will also face many or all the issues we raise here.

We note that although the physical displays on which AR content is drawn — in our examples, automotive windshields and HMDs — are fundamentally two-dimensional screens, applications may display both 2D content (e.g., informational overlays) as well as content that appears visually 3D. Dis-

[1] http://questvisual.com/

HotMobile '16, February 23 - 24, 2016, St. Augustine, FL, USA

ISBN 978-1-4503-4145-5/16/02...$15.00

DOI: http://dx.doi.org/10.1145/2873587.2873595

Copyright: Microsoft. Source: http://www.geek.com/microsoft/microsofts-hololens-demo-steals-the-show-at-build-2015-1621727/

Copyright: Microsoft. Source: http://www.engadget.com/2015/10/06/hololens-project-x-lets-you-blast-robot-armies-with-a-ray-gun/

Copyright: TechRadar. Source: http://www.techradar.com/us/news/wearables/mini-partnered-with-qualcomm-to-make-augmented-reality-glasses-1291801

Figure 1: From the left — HoloLens demo showcasing multiple Windows 10 applications; HoloLens demo of a mixed-reality game that maps out the user's physical environment and displays world-aware virtual content; BMW prototype AR glasses that provide the driver with 'X-ray vision' by syncing with the car's external cameras.

played content may be static (e.g., always in the bottom of the screen) or it may need to adjust dynamically as the user moves and/or the real-world environment changes.

Automotive. Recent years have seen substantial growth in automotive AR efforts [12]. For example, technology by Continental[2] overlays 2D information (e.g., current speed limit and navigational arrows) on a car's windshield, and BMW has proposed "X-Ray Vision" (shown in Figure 1) that allows the driver to see through the body of the car [3]. Other efforts aim to display more immersive 3D augmentations. Recent work from Honda Research [23] uses 3D visual cues to aid drivers in making left turns. Honda researchers have also discussed integrating other visual elements into the car's 3D display, such as road signs or street addresses.[3]

Head Mounted Displays (HMDs). HMDs represent another class of AR platforms rapidly gaining traction. Although the idea of AR HMDs has been around for some time [20], products such as Microsoft's HoloLens and Magic Leap's AR headset are leading industry efforts to bring this technology to production. These platforms have the potential to support a range of applications — Microsoft recently showcased several HoloLens applications (Figure 1) such as an AR video player, a 3D weather app, and a video game that places virtual characters into the user's view of the real world.[4,5] These applications highlight the ability of emerging platforms, like HoloLens, to display 3D content that exists in and interacts with some model of the user's world.

2.2 The Risks and Challenges

AR applications can provide uniquely immersive experiences; however, this ability can lead to undesirable consequences if the applications are buggy or malicious. As prior work [6] notes, today's AR apps typically operate with the unrestricted ability to access sensor inputs and render augmentations. Since applications may not always be trusted by the user or by the OS,[6] this model raises a number of security, privacy, and safety issues. Even if AR platforms only allow apps from trusted developers — a policy perhaps especially reasonable for cars — it is still critical to ensure that buggy applications cannot accidentally misbehave.

Input Privacy. Unrestricted sensor access by applications is a major privacy problem. Consider an HMD user in his or her home. Without safeguards, applications may be able to see sensitive information like medications. Prior efforts have focused on addressing visual input privacy (e.g., [9, 10, 14, 17, 22]). For example, one set of strategies leverages trusted OS modules to mediate sensor access for applications. Jana et al. propose the recognizer abstraction [9] as an OS module for fine-grained sensory access control, and Roesner et al. extend this model [17] to accommodate object- or environment-specific access control policies. The left and middle columns of Figure 2 depict the AR system pipeline from [17], which consists of system sensors, system recognizers, an input policy module, and applications. The system sensors collect information from the world, e.g., video from a camera. An OS module then recognizes objects, such as people or stop signs, and the policy module evaluates the relevant privacy (and other) policies to determine which recognized objects to communicate to which applications.

Output Safety and Security. While the above works make significant progress towards limiting the flow of sensitive sensor information *to* applications, they do not mitigate malicious or buggy outputs *from* applications. Output may take multiple forms, including visual, audio, and haptic feedback. In this work, we focus on managing visual outputs, because most of today's AR applications display visual content. We propose a new output module for the AR pipeline, shown in the right column of Figure 2. Absent defenses, a buggy or malicious application could overwrite the output of another application, display content that blocks or obscures real-world content, or display content that distracts (and possibly endangers) the user. By exploring the design space for our new output module, we establish a trajectory for defending against such threats. We consider several axes along which malicious or buggy output might occur:

- *Who* (i.e., which application) displayed particular content. Knowing this could, for example, be useful in disambiguating content generated by a phishing application or advertisement from content generated by a legitimate banking or route guidance application.
- *What* kind of content a particular application can draw. For example, should an automotive application be able to draw virtual pedestrians on the road?
- *When* an application can draw, based on the context of the user's actions or environment. For example, could an HMD texting application pop up a full-screen message when the user is doing something potentially hazardous, like walking down stairs?
- *Where* an application can draw, both on the display

[2]http://continental-head-up-display.com/
[3]https://www.youtube.com/watch?v=OgsrFAe_Lgc
[4]https://www.youtube.com/watch?v=gnlIEnHIJ7o
[5]https://www.youtube.com/watch?v=3AADEqLIALk
[6]Like prior work, we assume that the OS is trustworthy and trusted by the user.

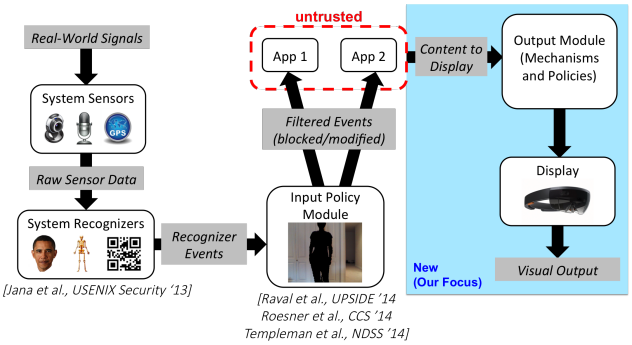

Figure 2: Pipeline for how data flows through an AR system and applications. Prior work focused on the operating system's role in processing and dispatching (possibly privacy-sensitive) input to applications. We extend this pipeline with an ***output module*** *and explore the operating system's role in managing the visual content (i.e., augmentations) that applications wish to display.*

(i.e., with respect to the user's "screen") and within the world (i.e., with respect to specific objects or 3D regions in the world). For example, could an automotive application render an ad on top of a road sign?

To our knowledge, existing research efforts and commercial systems have not explored solutions to address AR output safety and security. However, before AR systems can be widely adopted, it is critical to design mechanisms to ensure they are safe and secure, without significantly hindering the expressiveness of applications. Computer security is a form of risk management, and hence the full spectrum of adversarial actions that we discuss might never manifest. Nevertheless, our goal is to provide a technical foundation for mitigating such threats *if* they do arise.

3. DESIGN SPACE EXPLORATION

In this section, we explore several possible designs for the output module of the AR pipeline (Figure 2), focusing on visual output. The roles of the output module are: (1) to provide a mechanism for applications to specify and display visual content, and (2) to apply safety, security, and privacy policies by interposing on the rendering of this content. Our key questions in exploring this design space are: what role should the operating system play in managing visual output, and how should that role be realized? How can we design an output module to enable flexible application behaviors while mitigating risks from malicious or buggy applications? Our earlier case studies and the above-mentioned risks motivate two important design axes for managing visual AR outputs:

1. *Flexibility:* The ability of honest, non-buggy applications to display AR content. Ideally, an output module does not prevent honest applications from implementing and innovating on desirable AR functionality, including through possible interactions between apps.
2. *Control:* The operating system's ability to prevent applications from drawing malicious or undesirable AR content (e.g., "undesirable" in the automotive context may mean safety-critical distractions of the driver, or occlusion of safety-critical objects like road signs).

We now explore how these two axes play out in the design space of visual output models. We first consider two natural output models: the *windowing model* used by traditional platforms (e.g., desktops) and an alternate strawman approach in which applications can *free-draw* AR content. (We consider these models as discrete points in a design space, though of course changes to the implementation of one model could allow it to take on characteristics of another.) We find that windowing overly constrains visual output flexibility, while free-drawing overemphasizes flexibility at the expense of control. We thus introduce a novel model based on *fine-grained AR objects* that provides better flexibility than windows and better control than free-drawing.

We assume that applications can receive input about the real world to make decisions about how and where to display augmentations, possibly dynamically updating these decisions as the user's real-world view changes. For privacy reasons, applications may receive such input with varying degrees of fidelity (e.g., raw sensor input, recognizer events [9], or information about real-world surfaces [25]), but we do not focus on the problem of input privacy in this work.

3.1 Model 1: Windowing

We first consider how the traditional desktop windowing model translates to the AR context. Under this model, the OS delegates control of separate *windows* to applications — rectangular display spaces in which they may draw arbitrary content. Windows are typically isolated from each other, so that one application cannot read content from or manipulate content in another's window. Given that this model is well established in desktop systems, it seems natural to extend it to AR — and, indeed, it appears that HoloLens utilizes such a model in running Universal Windows applications.[7]

With traditional windows, applications can display content in roughly the following manner:

1. The OS gives applications handles to separate windows corresponding to bounded regions of the display. Traditionally, windows are rectangular; in the AR context, they may represent 3D rather than 2D regions that map to the user's real-world environment. Users can typically resize and reposition these windows.
2. Applications render arbitrary content to their respective windows by calling some `draw` function that interfaces directly with graphics hardware.

Under this model, different applications cannot occupy the same display region simultaneously — if two windows overlap, only one can be in focus at a time (i.e., one window occludes the other and receives inputs). Similarly, if one application is in "full-screen" mode, in most of today's systems, this means that only it can display content. Though some windowing implementations allow transparent windows or embedded windows (e.g., iframes on the web), display region sharing with windows is limited to these features.

Control. Under this model, application outputs are isolated, and content is rendered at the window granularity. This gives the system coarse-grained control: applications can only draw inside their own windows, and they cannot autonomously reposition or resize those windows. As a result, an application cannot draw over arbitrary regions of the screen (e.g., the road viewed through an AR windshield) unless the user or the system has explicitly placed its window there. However, the system is not able to enforce finer-grained properties, like where the application can draw *within* the window. It also cannot enforce requirements on arbitrary content the application draws inside its window.

[7] https://msdn.microsoft.com/en-us/library/windows/apps/dn726767.aspx

Flexibility. The windowing model also presents flexibility challenges for multiple applications running simultaneously. While it may suffice for applications whose content fits naturally inside a bounded window, such as the HoloLens video player, it may not suit others. Consider the HoloLens game in Figure 1. If the objects and characters in the game are intended to behave like real-world objects (e.g., to move about within and interact with the user's world), the application needs to render contextually with respect to the user's world. If the application does so in full screen mode, it either prevents other applications from rendering any content at all or, if multiple applications can overlay partially transparent full-screen windows, its content may arbitrarily overlap or conflict with the content of other applications in a way that cannot be controlled at a fine granularity by the OS.

3.2 Model 2: Free-Drawing

Given the flexibility limitations of traditional windowing, we next consider a point in the design space that is intentionally flexible, allowing applications to "free-draw" anywhere within the user's view, at any time. More precisely:

1. The OS gives applications handles to a shared window corresponding to the device's full display.
2. The applications each call a `draw` function to display 2D or 3D content in this shared environment.

Flexibility. This model provides maximum flexibility for applications, which can draw arbitrarily to provide desirable AR functionality unencumbered by the assumptions embedded in the traditional windowing model. With free-drawing (and sufficient information about the user's world), applications can track and augment any objects in the user's view and visually interact with content from other applications.

Control. Unfortunately, the flexibility this model provides to honest applications also enables malicious or buggy applications to more easily display undesirable content. Without restrictions, an app could display content that disorients or visually impairs the user, or that endangers the user by occluding or modifying possibly safety-critical real-world objects. Consider the car case study. With no output restrictions, a malicious app could occlude pedestrians or road signs, potentially causing physical harm to users and bystanders. Furthermore, with multiple applications free-drawing simultaneously, applications may directly interfere with or extract information from each other's visual content. These risks are greater than with isolated windows, even if those windows are partially transparent or overlapping. Thus, while allowing applications to free-draw supports flexible rendering needs, it also provides the OS with no capability to prevent or constrain malicious behavior, nor to isolate content from different applications.

3.3 Model 3: Fine-Grained AR Objects

The above models require trading off flexibility for coarse-grained output control. We desire a visual output model that provides *both* flexibility for honest applications as well as more fine-grained output control for the system. To achieve this goal, we introduce a new AR output model based on fine-grained objects. *The key idea is to manage visual content at the granularity of* ***AR objects*** *rather than windows.* Objects are abstract representations of AR content that applications wish to place into the user's view of the world, such as 3D virtual pets or 2D informational overlays. In the windowing and free-drawing models, applications must manage these objects internally. Our proposed model elevates these objects to first-class OS primitives.

Flexibility and Control. Object-level granularity provides key flexibility and control benefits. Applications can create and draw objects throughout the user's view of the world; however, by making these objects first-class OS primitives, the system can enforce rich object-based constraints and dynamically manage *where* and *when* objects can be drawn, as well as how objects from different applications can interact. For example, an AR car system could prevent applications from drawing over critical objects such as road signs while still allowing applications such as navigation to display content (e.g., direction arrows) dynamically throughout the world. In some cases, the system may even be able to regulate *what* objects an application draws, to the extent that the semantics of certain objects are known to the OS. For example, the system might only allow objects that follow a certain template to overlay on top of road signs.

Concrete Instantiation. There are many possible implementation strategies — for example, one might start with a windowing model and modify the definition of a window to bound arbitrary 3D objects, or one might start from scratch. Our goal is not to propose or evaluate specific implementations or APIs, but rather to explore AR objects as first-class OS primitives. Nevertheless, we present a strawman output module design, which allows us to illustrate this idea more concretely and raise important design questions:

1. AR objects consist of visual descriptors (e.g., 3D meshes and textures) and optional "physical properties" (e.g., how the objects should respond to collisions).
2. To display content, applications request the OS to draw particular objects with certain requirements. For example, apps might request specific display coordinates or locations relative to other application or real-world objects (e.g., to display labels on real-world faces).
3. The OS processes object placement requests, along with *constraints* from a variety of parties (e.g., applications, the user, or the system itself) and contextual information about the current world state (e.g., the current speed of the car). The OS then decides which requests are permissible (or modifies requests) and renders the appropriate visual content.

This model enables several desirable properties:

OS Support for Dynamic Updates. The OS can dynamically update and redraw objects in response to user actions (e.g., head movement) or changes in the real world, rather than requiring applications to handle these changes manually.

Shared World. The OS can position application content in a shared world and manage physics-based (and other) interactions between application and real-world objects. For example, applications might register for system-based events, like collisions between objects, and define how to respond to these events. We further discuss the system's possible roles in managing these interactions in Section 4.

Subsuming Previous Models. The ability to handle a rich set of possible constraints allows an object-based model to *subsume* both the windowing model and the free-drawing model. Constraints based on logical windows, or the lack of any constraints, would emulate these models, respectively.

Contextual Tuning. Object-level granularity allows the OS

to contextually tune how strictly it applies controls, since permissible behavior may vary based on user preferences, application needs, and/or real-world context. For example, content that can safely be displayed on an AR windshield while a car is parked differs significantly from content that can safely be displayed while the car is in motion. With object-level control, and some notion of object semantics, the system can contextually tune which application outputs are controlled and how. This type of control is not achievable with a coarser-grained windowing model, which can control only where and whether an application can display.

Summary. Our key insight is that, by managing visual output at the granularity of objects rather than in windows or the full display, the OS can enable flexible application behaviors while *simultaneously* preventing them from displaying undesirable content. Instantiating such a model, however, requires rethinking rendering abstractions and interactions between applications. Future AR systems must consider these issues if they are to provide both rich experiences as well as output safety and security for users.

4. CHALLENGES AND OPPORTUNITIES

We now dive more deeply into the design and research questions that our object-based output model raises, as well as the novel opportunities it enables.

4.1 Key Design Challenges

We first consider a set of important design questions that must be addressed to realize an object-based output model.

Defining Objects and Rendering APIs. Two key considerations in supporting fine-grained, object-level augmentation control are — how should the OS define objects, and what kind of APIs should it expose to applications for displaying content? We presented a strawman API in Section 3.3, but carefully designing these APIs to minimize the burden on application developers will be critical.

Constraint Specification and Evaluation. Important object management questions we have not yet addressed are: according to what *constraints or policies* should the OS do this management, who specifies these constraints, how are they expressed, and how are they evaluated? For example, reasonable constraints in an automotive AR environment might be that applications cannot display ads or overlay objects on real-world traffic signs while the car is in motion. However, these constraints may change contextually (e.g., the car may permit ads to be shown when the vehicle is stopped). Furthermore, different parties — applications, the system, external regulatory agencies, or the user — may have conflicting preferences. Key research questions are how the OS should allow multiple parties to specify policies or constraints, how to evaluate potentially conflicting constraints in real time, and how to manage conflicts when they arise.

Managing Objects in a Shared Environment. Our object-based model allows the OS to mediate interactions between augmentations from different applications within a shared environment, though exactly what this role should be raises open questions. How much information about the real world do applications need to intelligently position (or specify constraints for the positions of) their objects? What kind of feedback should the OS provide to applications when their objects interact with each other or the physical world (e.g., a virtual ball bouncing on the real-world floor) given that such feedback might leak private information about one application to another? How should the OS handle possibly malicious object interactions, such as one app that tries to draw over another app's object and occlude it from view? These questions highlight some of the unique challenges that immersive multi-application AR platforms may encounter.

4.2 New Capabilities

We next explore how our object-based model allows the OS to take on new roles in supporting AR applications.

OS Support for Object Semantics. Our model allows the OS to dynamically manage the interactions of application objects with each other and the real world, significantly reducing the burden on application developers to reposition objects as the user's view changes. Going one step further, the OS could *natively support* certain AR objects (e.g., people) in addition to abstract objects, by allowing applications to register objects under pre-defined classes. This design allows the OS to have a semantic understanding of certain objects and to more intelligently manage their interactions. For example, it would enable an output policy such as "applications may draw only pre-approved types of objects on top of streets while the user is driving" — a significantly more flexible policy than disallowing any drawing at all, while still maintaining safety properties. However, applications may wish to display diverse types of objects, and natively supporting all possible classes of objects is infeasible. A key question for AR system designers is thus what role, if any, the OS should take in providing native object support.

Supporting Novel Inter-App Interactions. While some amount of inter-application feedback is necessary to maintain a consistent world state within a shared environment (e.g., so applications know when their objects collide), an open question is — to what extent should applications be aware of each other? Object-level granularity raises interesting new possibilities for how applications can interact. For example, could an application take as input not only real-world objects, like faces, detected by system recognizers (as proposed in prior work [9]), but also virtual objects created by other applications? A plausible use case would be a translation application such as WordLens that takes as input not only real-world text, but also text generated by other (single-language) applications. This degree of interaction could enable powerful new AR scenarios, but it raises new challenges as well: for example, could applications manipulate their virtual objects to attack other applications that take these objects as input? Whether and how best to support such interactions must be carefully considered.

5. RELATED WORK

Researchers have explored applications and technical challenges surrounding AR for decades (e.g., [1,2,5,13,20,24,27]). However, only recently have these technologies become commercially available, and only recently have researchers begun seriously considering the security, privacy, and safety concerns associated with AR (e.g., [6,16]). Most technical work to date has focused on *input* privacy (e.g., [9,10,14,17,22]); in this paper we consider the challenge of *output* safety and security. Surround Web [25] considers both AR input and output issues, in that it limits the amount of real-world input applications need to decide where and what to display;

its techniques are complementary to ours. Some of the issues that we raise — e.g., the need to consider AR output security at all — have also been identified previously [6, 16], but we expand on these directions here and make significant progress toward developing a visual output model for AR that balances OS control and application flexibility.

Related earlier efforts in the AR space have studied multi-application settings [18] and world-aware virtual objects [4]. These works do not consider security explicitly, but our object-based model could incorporate some of these earlier ideas. These works both also consider multi-user settings, which we do not consider — though we observe that they will raise interesting additional security and privacy questions.

More generally, prior work has considered display security in non-AR contexts, including secure window systems (e.g., [7, 19, 26]), shared cross-application UIs (e.g., iframes on the web or research proposals for smartphones [15]), and non-AR automotive displays (e.g., [8]). Though some shared issues arise in these other contexts, the complexity of AR scenarios (with virtual 3D objects and real-world interactions) raises significant new challenges.

6. CONCLUSION

Immersive AR platforms, such as Microsoft's HoloLens, are quickly becoming a reality. However, fundamental security and privacy challenges exist. Past works largely consider the privacy implications of visual *inputs* available to AR applications. We initiate a complementary investigation into the security risks of visual AR *outputs*. For example, multi-application AR systems need mitigations to prevent malicious or buggy applications from displaying content that blocks the outputs of other applications, or that occludes critical real-world objects (e.g., traffic signals or stairs). Our key technical proposal is to enable operating system control over visual output at the granularity of individual *AR objects* that are owned by applications and correspond to items with which the applications wish to augment the user's view. We provide an initial design and reflect upon how it can enable safer, diverse multi-application AR environments.

7. ACKNOWLEDGEMENTS

We thank Linda Ng Boyle and David Molnar for valuable discussions, as well as Antoine Bosselut and Suman Nath for helpful feedback on earlier drafts. This work was supported in part by NSF grants CNS-0846065, CNS-0963695, and CNS-1513584 and the Short-Dooley Professorship. This material is based on research sponsored by DARPA under agreement number FA8750-12-2-0107. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

8. REFERENCES

[1] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.

[2] R. T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6:355–385, 1997.

[3] R. Baldwin. Mini's weird-looking AR goggles are actually useful, Apr. 2015. http://www.engadget.com/2015/04/22/bmw-mini-qualcomm-ar/.

[4] A. Butz, T. Höllerer, S. Feiner, B. MacIntyre, and C. Beshers. Enveloping users and computers in a collaborative 3D augmented reality. In *IEEE/ACM International Workshop on Augmented Reality*, 1999.

[5] E. Costanza, A. Kunz, and M. Fjeld. Human machine interaction. chapter Mixed Reality: A Survey, pages 47–68. Springer-Verlag, 2009.

[6] L. D'Antoni, A. Dunn, S. Jana, T. Kohno, B. Livshits, D. Molnar, A. Moshchuk, E. Ofek, F. Roesner, S. Saponas, et al. Operating system support for augmented reality applications. *Hot Topics in Operating Systems (HotOS)*, 2013.

[7] J. Epstein, J. McHugh, and R. Pascale. Evolution of a trusted B3 window system prototype. In *IEEE Symposium on Security and Privacy*, 1992.

[8] S. Gansel, S. Schnitzer, A. Gilbeau-Hammoud, V. Friesen, F. Dürr, K. Rothermel, and C. Maihöfer. An access control concept for novel automotive HMI systems. In *ACM Symposium on Access Control Models and Technologies*, 2014.

[9] S. Jana, D. Molnar, A. Moshchuk, A. M. Dunn, B. Livshits, H. J. Wang, and E. Ofek. Enabling fine-grained permissions for augmented reality applications with recognizers. In *USENIX Security*, 2013.

[10] S. Jana, A. Narayanan, and V. Shmatikov. A Scanner Darkly: Protecting user privacy from perceptual applications. In *IEEE Symposium on Security and Privacy*, 2013.

[11] L. Mathews. Microsoft's HoloLens demo steals the show at Build 2015, 2015. http://www.geek.com/microsoft/microsofts-hololens-demo-steals-the-show-at-build-2015-1621727/.

[12] M. May. Augmented reality in the car industry, Aug. 2015. https://www.linkedin.com/pulse/augmented-reality-car-industry-melanie-may.

[13] G. Papagiannakis, G. Singh, and N. Magnenat-Thalmann. A survey of mobile and wireless technologies for augmented reality systems. *Computer Animation and Virtual Worlds*, 19:3–22, 2008.

[14] N. Raval, A. Srivastava, K. Lebeck, L. Cox, and A. Machanavajjhala. Markit: privacy markers for protecting visual secrets. In *Workshop on Usable Privacy & Security for wearable and domestic ubIquitous DEvices (UPSIDE)*, 2014.

[15] F. Roesner and T. Kohno. Securing embedded user interfaces: Android and beyond. In *USENIX Security Symposium*, 2013.

[16] F. Roesner, T. Kohno, and D. Molnar. Security and privacy for augmented reality systems. *Communications of the ACM*, 57(4):88–96, 2014.

[17] F. Roesner, D. Molnar, A. Moshchuk, T. Kohno, and H. J. Wang. World-driven access control for continuous sensing. In *ACM Conf. on Computer & Communications Security*, 2014.

[18] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnaçao, M. Gervautz, and W. Purgathofer. The studierstube augmented reality project. *Presence: Teleoperators and Virtual Environments*, 11(1):33–54, 2002.

[19] J. S. Shapiro, J. Vanderburgh, E. Northup, and D. Chizmadia. Design of the EROS trusted window system. In *13th USENIX Security Symposium*, 2004.

[20] I. E. Sutherland. A head-mounted three-dimensional display. In *Fall Joint Computer Conference, American Federation of Information Processing Societies*, 1968.

[21] A. Tarantola. HoloLens 'Project XRay' lets you blast robot armies with a ray gun fist, Oct. 2015. http://www.engadget.com/2015/10/06/hololens-project-x-lets-you-blast-robot-armies-with-a-ray-gun/.

[22] R. Templeman, M. Korayem, D. Crandall, and A. Kapadia. PlaceAvoider: Steering first-person cameras away from sensitive spaces. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

[23] C. Tran, K. Bark, and V. Ng-Thow-Hing. A left-turn driving aid using projected oncoming vehicle paths with augmented reality. In *5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2013.

[24] D. van Krevelen and R. Poelman. A survey of augmented reality technologies, applications, and limitations. *The International Journal of Virtual Reality*, 9:1–20, 2010.

[25] J. Vilk, A. Moshchuk, D. Molnar, B. Livshits, E. Ofek, C. Rossbach, H. J. Wang, and R. Gal. SurroundWeb: Mitigating privacy concerns in a 3D web browser. In *IEEE Symposium on Security and Privacy*, 2015.

[26] J. P. L. Woodward. Security requirements for system high and compartmented mode workstations. Technical Report MTR 9992, Rev. 1 (also published by the Defense Intelligence Agency as DDS-2600-5502-87), MITRE Corporation, Nov. 1987.

[27] F. Zhou, H. B.-L. Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: a review of ten years of ISMAR. In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008.