Enabling Multiple Applications to Simultaneously Augment Reality: Challenges and Directions

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

ABSTRACT

Augmented reality (AR) platforms are evolving to support immersive 3D experiences. Most modern AR platforms support only a single immersive app at a time, but users may also benefit from the ability to engage with multiple apps at once. The ability of different apps to simultaneously augment a user's world raises critical questions: how might apps visually conflict with each other, and how can we design AR platforms to support rich behaviors while mediating conflicts? In this work, we pose and explore these questions, identifying means of visual conflict as well as platform design strategies to mediate conflicts. We then analyze state-of-the-art AR platforms (HoloLens, Magic Leap One, and Meta 2) to understand their trade-offs and identify unexplored gaps in the broader design space. Our exploration reveals key guidelines and lessons to inform future multi-app AR efforts.

ACM Reference Format:
Kiron Lebeck, Tadayoshi Kohno, Franziska Roesner. 2019. Enabling Multiple Applications to Simultaneously Augment Reality: Challenges and Directions. In *The 20th International Workshop on Mobile Computing Systems and Applications (HotMobile '19), February 27–28, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3301293.3302362

1 INTRODUCTION

Augmented reality (AR) is ushering in a new era of immersive computing, with devices that can understand a user's physical world and blend 3D content into the user's view of the world. However, most modern AR platforms do not allow users to engage with multiple immersive apps simultaneously, and those that provide multi-app support still have significant limitations. Consider a user who wishes to engage with multiple apps while walking in a city, such as an AR navigation app [5], an AR game [11], and social apps that augment nearby people, e.g., by displaying their names above their heads or 3D masks over their faces. On a single-app platform, the user can only view and interact with one app at a time. By contrast, a multi-app platform could allow the user to shift their attention between apps — for example, periodically glancing at directions without closing their game, while still seeing social overlays on nearby people.

HotMobile '19, February 27–28, 2019, Santa Cruz, CA, USA

ACM ISBN 978-1-4503-6273-3/19/02…$15.00
https://doi.org/10.1145/3301293.3302362

Realizing the vision of multi-app AR will require identifying and overcoming new challenges that stem from the unique capabilities of AR platforms. In particular, rather than sharing the blank canvas of a traditional computer screen and displaying content within isolated windows, the output of immersive AR apps will exist atop the backdrop of the user's ever-changing world. These apps may need to dynamically update their outputs in response to changes in the user's physical environment while simultaneously displaying content alongside each other, raising fundamental questions: how might immersive AR apps visually conflict with each other, and how can multi-app AR platforms allow different apps to simultaneously augment their shared world while mediating conflicts?

Prior AR-related efforts [1, 8, 9, 13] primarily focused on *individual* apps negatively influencing users' perceptions of the real world, rather than on visual conflicts between multiple apps. We currently lack a foundation for reasoning about these conflicts or understanding the design challenges involved with supporting multiple immersive apps. In this work, we provide such a foundation by conducting an intellectual investigation into the multi-app AR design space, deferring implementation and experimental evaluations to future work. Specifically, we contribute the following:

1. *Problem Identification*: We identify the need to view the design space of multi-app AR platforms with a critical eye towards visual conflicts that may occur between the output of different apps.
2. *Design Space Exploration*: We introduce a broad categorization of approaches for multi-app AR platforms to handle conflicts, and we uncover key trade-offs presented by different design strategies.
3. *AR Platform Analysis*: We analyze the multi-app capabilities of modern AR platforms to understand how they fit into the broader design space.
4. *Future Directions*: Through our exploration and analysis, we identify promising directions for future work. For example, we encourage future work to implement and evaluate key concepts set forth in this paper.

2 MOTIVATION

We begin with case study scenarios that highlight the possibilities of multi-app AR, including risks that users may face from visual interactions between apps.

Tourism. Alice uses Tour Guide while on vacation, which displays floating icons above landmarks that she can select to read more information. Restaurant Assistant displays food safety and customer ratings above nearby restaurants, which Alice can select to read detailed reviews and menu options. Navigation guides Alice as she walks to a new destination by displaying directional arrows

on the ground, and for entertainment, an immersive Pokémon game blends interactive 3D characters into Alice's physical environment.

Unfortunately, multiple Pokémon characters inadvertently stand atop Alice's Navigation arrows on the ground and prevent Alice from seeing her directions. At the same time, Tour Guide has an endorsement contract with a local café, and to discourage Alice from eating elsewhere, it displays fake negative ratings above other eateries that block the true ratings of Restaurant Assistant.

Social Gatherings. Bob is attending a festival with friends and wishes to connect with other attendees. Social Media AR recognizes nearby people in Bob's extended network and displays their names, mutual friends, and common interests above their heads. Since Bob is interested in romantic connections, he also uses AR Dating, which computes compatibility scores of other users, highlights them, and displays the scores above their heads. Finally, Bob and his friends use Immersive Snapchat Filters to modify each other's appearances in fun ways, such as overlaying humorous costumes.

Bob notices that a friend-of-a-friend is also identified as a potential romantic connection, with Social Media AR and AR Dating both displaying information above their head. However, content from both apps appears jumbled atop each other, and Bob cannot disambiguate content from either app. AR Dating also identifies other potential partners near Bob, but since Snapchat has already displayed full-body filters over them, AR Dating cannot highlight them.

The Workplace. Carol and her colleagues use AR to improve productivity at work, with Collaborative Workspace allowing them to interact with shared 3D models and virtual whiteboards both in the office and remotely. Colleague Assistant displays helpful reminders that float next to Carol's coworkers (such as upcoming meetings or recent emails), and AR Chat allows Carol to stay connected with her team by displaying real-time messages that float next to her. Finally, AR Art lets Carol easily personalize her workspace with virtual paintings, sculptures, and other artwork.

Carol finds Colleague Assistant helpful, but the app is compromised and intentionally positions its reminders to obscure AR Chat messages. While AR Art improves Carol's workplace ambiance, the app is buggy and creates 3D objects that interfere with Collaborative Workspace. Since there is no indication that AR Art created these objects, Carol believes Collaborative Workspace to be malfunctioning and disables it. Additionally, when an AR Chat message moves atop an AR Art piece on Carol's desk, the art is "knocked" to the ground.

A New Output Paradigm. The above scenarios raise a fundamental question: can a multi-app AR platform support the diverse needs of immersive apps while also mitigating negative interactions between them? As with apps on other computing platforms, immersive AR apps may compete for resources such as memory, CPU cycles, and network bandwidth. What sets these apps apart are their output needs.

Consider a traditional desktop app, such as a video game, text editor, or web browser. The outputs of these apps exist within independent windows, and the behavior of these apps does not depend upon the precise placement of their windows on the computer screen (i.e., the user could reposition any of the windows and the apps would behave the same). However, in AR, the behavior of an app may depend directly on how its outputs are positioned in the context of the user's world. For example, the efficacy of Alice's Navigation app depends upon the app's ability to precisely position directional arrows on the ground, and Bob's AR Dating and Social Media AR apps must be able to place overlays above specific people's heads. Furthermore, on a traditional desktop display, all content shown on screen is controlled directly by either apps or the OS. By contrast, users will view AR apps atop the backdrop of the physical world rather than a blank screen. This external environment may change unpredictably, introducing variability that AR apps may need to contend with. For example, apps may need to dynamically update their outputs in response to changes in the user's world itself (e.g., AR Dating must update the locations of its overlays as people move throughout Bob's field of view), as well as changes in the user's own position within the world (e.g., Navigation must appropriately place new arrows on the ground as Alice walks around and changes directions). AR presents a new output paradigm from traditional displays, creating new challenges that will require novel solutions.

Threat Model. In this work, we focus on the conflicts that stem from visual interactions between immersive AR apps, leaving a discussion of additional output modalities (e.g., audio) for Section 5. Furthermore, we focus on users' *perceptions* of AR content rather than their *interactions* with apps. Output conflicts may lead to harmful user interactions (e.g., AR "clickjacking"), but such issues depend on the specific input capabilities provided by an AR platform, which we consider out of scope.

Our threat model encompasses both apps that are malicious, as well as apps that are honest-but-buggy and do not intentionally seek conflict. We begin by considering a broad space of visual conflicts that may arise, including the following:

- *Occlusion.* The output of one app might block the user from seeing that of another. For example, Alice, Bob, and Carol all encounter occlusion above. We exclude situations where the user intentionally positions one app's content to occlude other apps, focusing on occlusion events that arise in the absence of user intent.
- *Placement Denial.* By occupying a particular space, one app might prevent another from generating content. For example, Bob's Snapchat app prevents AR Dating from highlighting certain individuals, by occupying the space around them with full-body filters.
- *Eviction.* By moving content into a space occupied by another app, an offending app might cause the victim's content to be removed or displaced, as Carol experiences when AR Chat knocks an AR Art object to the ground.
- *Masquerading.* One app might generate content that is mistaken for that of another. For example, Carol mistakenly perceives buggy output from AR Art as output from Collaborative Workspace.
- *Content Modification.* As we will see in Section 3.2, certain conflict mediation mechanisms may modify the visual properties of app outputs, e.g., by adjusting transparency. Such approaches raise an additional threat: one app may be able

to induce changes in the visual properties of another app's content.

3 DESIGN SPACE EXPLORATION

We now turn to our design exploration of multi-app AR platforms, asking: how can these platforms mediate visual conflicts between apps, and what are the trade-offs associated with different design alternatives? We consider the ability of an AR platform to meet the following criteria while remaining resilient to the above-mentioned conflicts:

- *Support for Multiple Applications.* Does the platform allow multiple apps to run simultaneously?
- *Full Output Autonomy.* Does the platform give apps full control over the placement of their outputs in 3D space?
- *Some Output Autonomy.* Does the platform give apps at least *some* positional control over their outputs?
- *Limited User Burden.* Does the platform require limited or no user involvement in managing output?
- *Limited Developer Burden.* Does the platform limit the need for app developers to handle unexpected interactions with other apps?

Figure 1 summarizes key trade-offs that characterize the design paths we discuss throughout this section.

3.1 Display Abstractions

The interface that an AR platform provides to apps for displaying content determines the space of available output behaviors. Consider the following:

Single-App. Inter-app conflicts cannot occur if only one app can display content at a time. While this approach is at odds with our goal of supporting multiple apps, it is the only design in Figure 1 to meet every other goal and may suffice for individual apps requiring the user's undivided attention.

Windows. One method for preventing output conflicts is to confine apps to separate regions of space — a 3D analogue of the window abstraction used by desktop PCs. We consider a model where windows are controlled by the user and cannot be created or repositioned autonomously by apps. These properties allow windows to visually isolate apps from each other, but in doing so, they trade-off the ability for apps to dynamically generate content throughout the user's world. While our prior work argued for the insufficiency of windows in AR due to such flexibility limitations [8], we find their viability actually depends upon the needs of specific apps. For example, Carol's AR Chat, AR Art, and other apps naturally fit within bounded spaces, but Alice's Pokémon and Navigation apps require more dynamic output capabilities.

Shared World. The final model we consider is a shared world that allows multiple apps to simultaneously display content throughout the user's environment. This approach stands in contrast to windows, sacrificing visual isolation to give apps the flexibility to place AR content wherever they wish. As a result, one app may draw in the same space as another app or otherwise occlude that app's output. We explore strategies for addressing such conflicts below.

3.2 Managing Output in a Shared World

When considering how to manage output conflicts in a shared world, we must first determine *who* should shoulder this burden. Thus, we explore opportunities for the OS, apps themselves, or the user to take on this responsibility. While we present these design paths individually, we note that they may be combined to manage output in different ways.

3.2.1 OS-Enforced Conflict Mediation

As discussed above, giving apps the freedom to place content wherever they wish may lead to occlusion conflicts. We thus begin with two complementary design paths that enable the OS to prevent occlusion. These designs leverage the AR object abstraction proposed in our prior work [8]. AR objects are OS-managed primitives that encapsulate AR output — for example, a single Pokémon character would be one AR object. The OS can modify the visual properties of AR objects (e.g., position or transparency) to prevent occlusion. Specifically, we introduce the following approaches:

1. *Runtime Policies.* The OS prevents occlusion by observing visual interactions between AR objects at runtime and enforcing policies that modify them in response. For example, the OS could observe when one of Alice's Pokémon objects occludes a Navigation arrow and turn the Pokémon object partially or fully transparent to ensure that Navigation's arrow remains visible.
2. *Declarative Output.* The OS provides apps with a language to abstractly indicate their output needs, but it controls *how* these needs are met to prevent occlusion. For example, Bob's AR Dating and Social Media apps could request to display content above someone's head, and the OS would determine an appropriate layout. Similarly, Alice's Restaurant Assistant app could place virtual signs in front of restaurants without controlling the precise 3D coordinates of these objects.

Trade-off: Intelligent Mediation vs. App Freedom. Runtime policies only allow the OS to identify occlusion after it has occurred, and they provide no contextual information about how the OS should respond to individual conflicts. By contrast, declarative output ensures that apps do not conflict in the first place, and by capturing the high-level needs of apps, it gives the OS the ability to intelligently respond to app requests. Consider AR Dating and Social Media from above. If the OS understands that both apps are attempting to augment the same person's head, it could (for example) arrange content so that both apps are visible above the person's head, rather than making one app's objects invisible.

In providing more effective mediation capabilities, declarative output trades off the ability to support fine-grained object placement for apps. Declarative output naturally caters to apps that can specify their output needs in terms of high-level visual relationships to physical-world objects, such as AR Dating. However, this approach does not lend itself to apps such as Alice's Pokémon game, which needs to create and move characters at precise 3D locations in Alice's world. For apps such as Pokémon that cannot operate under a declarative model, runtime policies provide the OS with a potential fallback mechanism for mediating conflicts.

			Output Conflicts					Functionality Goals				
			Occlusion	Placement Denial	Eviction	Masquer-ading	Content Modification	Multi-App Support	Full Output Autonomy	Some Output Autonomy	Limited User Burden	Limited Dev Burden
Display Abstractions	Single App		✔	✔	✔	✔	✔	N	Y	Y	Y	Y
	Windows		✔	✔	✔	✕	✔	Y	N	N	N	Y
	Shared World		✕	✔	✔	✕	✔	Y	Y	Y	Y	Y
Output Management in a Shared World	Runtime Policies	Ex1: Modify Occluder	✔	✔	✔	✕	✕	Y	N	Y	Y	N
		Ex2: LRU	★	✔	✕	✕	✔	Y	N	Y	Y	N
	Declarative Output	Ex1: Defined Layout	✔	✕	✔	✕	✔	Y	N	Y	Y	N
		Ex2: LRU	✔	✔	✕	✕	✔	Y	N	Y	Y	N
	Application Self-Management		★	✔	✔	✕	✔	Y	Y	Y	Y	N
	User-Managed Output		✔	✔	✔	✕	✔	Y	N	Y	N	N

Figure 1: Potential design paths for multi-app AR platforms. Check marks indicate that a design can prevent a conflict; stars indicate that the conflict is prevented when apps are trusted; and Xs indicate that a design cannot prevent the conflict.

Preventing Occlusion Can Enable New Conflicts. Preventing occlusion in a shared world fundamentally requires the OS to constrain the output behaviors of apps. In doing so, the OS may enable new forms of conflict. Recall the example runtime policy in which Pokémon's object is made transparent when it occludes Navigation's arrow. This policy allows Navigation to *induce* visual modifications in Pokémon's objects by placing arrows behind them. A declarative approach can also enable new conflicts — for example, the OS may deny an app's request to display content if it cannot determine an acceptable layout that would accommodate this request without causing occlusion.

As another cautionary example, consider a least-recently-used (LRU) mechanism that identifies overlapping objects and removes those that the user has interacted with least recently. When applied as a runtime policy or declarative output tool, an LRU mechanism enables even well-intentioned apps to inadvertently evict each other. Furthermore, a malicious app could leverage an LRU runtime policy to probe for the locations of other apps' objects by observing when its *own* objects are evicted, using this information to surround a victim app's objects and occlude them.

Limitation: Conflict Identification. A limitation of any OS-driven approach is that the OS may not be able to unilaterally decide which visual interactions are problematic. If the OS can determine a prioritization ordering for different apps, it can potentially decide which apps to act upon when mediating conflicts, whether it employs runtime policies, declarative output, or another strategy. However, the notion of what constitutes a conflict may not always be obvious, nor the decision of which app should receive priority. Note that we previously explored the idea of OS-enforced runtime policies in prior work [9]. However, that work focused primarily on visual conflicts between AR objects and real-world objects, where the real world was assumed to take priority, and it did not deeply consider the viability of runtime policies for resolving multi-app conflicts.

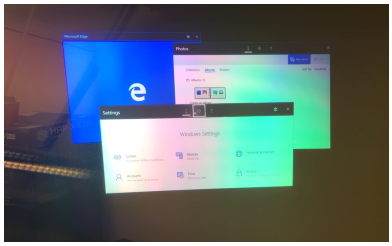
3.2.2 Application Self-Management

We next consider the potential for apps to collaborate in avoiding conflicts by sharing information with one another and reacting to each other's requests. For example, if Alice's Navigation app could provide the 3D locations of its directional arrows to Pokémon and request that Pokémon not occlude them, then Pokémon could adjust its behavior while still providing the user with the same overall experience.

Application self-management allows apps to retain control over their outputs and respond to conflicts in predictable ways, in contrast to OS-enforced policies that impose external modifications on app content. The consequence of giving apps this level of control is that self-management is only viable under a threat model where apps are trusted to avoid interfering with each other given the information to do so (e.g., on a closed platform running well-vetted apps that are designed to cooperate). A malicious app could leverage any additional information given to it about other apps to attack them — for example, if Pokémon was malicious and learned precisely where Navigation's arrows were, it could strategically generate objects that occlude those arrows.

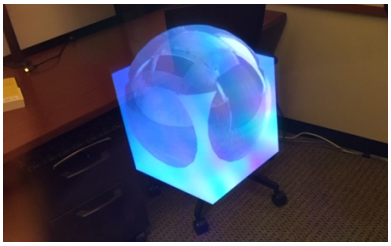
3.2.3 User-Managed Output

Ultimately, the user may be best positioned to determine which conflicts are detrimental to their own AR experience. Thus, the final design path we explore is one that leaves mediation to the user's discretion. An AR platform could provide the user with different tools for this task — for example, to demote problematic apps to more restrictive states (e.g., confining them to windows), to delete



(a) Microsoft HoloLens

(b) Meta 2



(c) Magic Leap One

Figure 2: Multi-app photos from three AR headsets, taken with an iPhone 6 through the lens of each device.

individual AR objects, or to provide apps with behavioral cues (e.g., to instruct an app to avoid displaying content in specific spaces).

The OS also has an opportunity to inform the user's actions by enabling the user to easily discern potential conflicts. Recall Carol's Collaborative Workspace app — Carol believed this app to be misbehaving, but the OS could inform her that the problematic object came from another app. Furthermore, the user may be unaware that certain conflicts have actually occurred. For example, unbeknownst to Alice, her Tour Guide app displayed fake restaurant ratings that hid the overlays of Restaurant Assistant. The OS could identify such visual interactions and provide Alice with this information so that she can act according to her wishes.

3.3 Summary

Identifying and mediating visual conflicts between AR apps is challenging, and different design strategies present varying trade-offs, as showcased in Figure 1. Our key insight is that any output mediation technique will infringe upon app functionality, and the precise nature of this infringement differs between design paths. Additionally, we observe that different techniques will be appropriate under different trust models, and our exploration highlights the potential for malicious apps to abuse well-intentioned capabilities.

4 AR PLATFORM ANALYSIS

In this section, we analyze the Microsoft HoloLens, Meta 2, and Magic Leap One AR headsets, asking: how do they fit into the broader design space above, and what unexplored directions may warrant further investigation? Each platform supports an immersive single-app mode that aligns with the first row of Figure 1, and we thus focus our analysis on the platforms' multi-app modes. Figure 2 depicts multi-app photos that we took through the lens of each device.

HoloLens. The HoloLens's multi-app mode supports Universal Windows Platform (UWP) apps, which run within 2D windows placed in 3D space by the user (Figure 2a). UWP apps run across different Microsoft platforms, providing a familiar interface for both users and developers. The window abstraction sacrifices support for immersive output to allow the HoloLens to enforce strong visual isolation between apps.

Meta 2. The Meta 2's multi-app mode is similar to that of the HoloLens, employing 2D windows placed in 3D space by the user (Figure 2b). The device tethers to a desktop PC and supports virtual "computer monitors" that enable the user to interact with their desktop's apps within AR windows.

Magic Leap One. By contrast, the Magic Leap One's multi-app mode supports multiple 3D apps at once. Apps may create "prisms" — bounded 3D regions in which they can display content. To probe the capabilities of prisms, we built multiple apps that display simple geometric shapes, and we ran two simultaneously. Figure 2c depicts two such apps: one displays a cube within a prism, and the other displays a sphere within a separate prism.

Prisms can be placed by the user, but we discovered that prisms from different apps are created atop each other by default. Apps can specify their prisms' sizes, but we could not determine if they can also control prism positions. If an app *can* control prism sizes and positions, then prisms act as a form of a shared world without conflict mediation mechanisms. As shown in Figure 2c, this design enables occlusion conflicts to occur. Furthermore, note that the cube and sphere are interleaved in 3D space, rather than one app receiving explicit rendering priority. Combining output from different apps in this way does not make intuitive sense from a user's perspective, suggesting that this occlusion is not intended behavior. We note that the Magic Leap developer guidelines suggest that prisms are *intended* to act as well-defined 3D windows, but this intention is not enforced by the platform.

5 DISCUSSION

Our design exploration and analysis establish a foundation for understanding and addressing key multi-app AR challenges. Here, we identify promising avenues for future work.

Output Management Techniques. Our analysis reveals a nascent multi-app landscape among today's AR platforms. Critically, no platform provides a shared world abstraction endowed with additional conflict mediation capabilities. Of the mediation strategies captured in Figure 1, we believe that declarative output is the most compelling path for further exploration. A declarative approach can prevent output conflicts even with malicious apps, and it strikes a balance between app flexibility and conflict mediation. The OS can handle app requests in a more predictable manner than runtime policies allow, and apps can exercise more immersive behaviors than a windowed display abstraction supports. Furthermore, this approach does not impose the burden of output management on users. Even though declarative output cannot support apps that require arbitrary 3D placement, it is well-suited for apps tasked with augmenting specific real-world objects (e.g., Tour Guide and AR Dating).

Going forward, we propose that future work should validate the conceptual directions laid out in this work, by investigating the

viability of declarative output (as well as the other above-mentioned output management techniques) in greater depth. One path would be to build a multi-application AR platform that supports different mediation strategies, and to evaluate these strategies along a number of axes — for example, the performance overheads that each technique imposes on applications; the ability of these techniques to effectively resolve output conflicts; the functionality limitations they place on application behaviors; and the burdens they place on both developers and users. Evaluating these criteria will better illuminate the trade-offs presented by different design paths, and will confirm (or contradict) our initial intuition regarding declarative output as the most promising path forward.

Non-Visual Output. While this work lays a foundation for addressing conflicts between AR applications in terms of visual output, AR platforms may provide additional output modalities as well, such as aural or haptic feedback. Future work should investigate conflicts that may arise between AR apps in terms of non-visual output, determine if and where design strategies for preventing visual conflicts can be adapted to non-visual settings, and identify areas where new approaches will be required. Additionally, future work should consider opportunities for AR platforms to leverage combinations of multiple output modalities to mediate conflicts (e.g., by incorporating both aural and visual cues to help users contend with visual conflicts between apps).

Understanding User Perceptions of Conflict. Determining the visual interactions that users find problematic can inform defensive efforts, particularly for conflicts that cannot be fully prevented. For example, as suggested in Figure 1, no design can truly prevent masquerading, which depends upon users' perceptions of AR content. An AR platform can *attempt* to prevent masquerading, just as early windowing systems employed labeling techniques to indicate the origins of different windows (e.g., [3]). However, a user may still incorrectly perceive the origin of AR content. Future work is thus needed to identify design strategies that effectively engage the user and minimize the impacts of such conflicts.

6 RELATED WORK

Our analysis reveals limited multi-app support on today's AR platforms. Researchers have previously proposed AR systems that support multiple apps, but they have not rigorously explored the design space or reasoned about conflicts that may arise. Argon [10] instantiates a shared world with overlapping full-screen, transparent windows for different apps, without conflict mediation. By contrast, Studierstube [15] confines app outputs to bounded 3D windows controlled by the user. Earlier non-AR efforts also considered secure windowing (e.g., [3]), but as discussed, AR raises new challenges.

Researchers have also studied security and privacy for AR more generally [2, 13]. Prior works consider output security (e.g., [1, 2, 8, 9, 13]), focusing on ways that AR apps could negatively impact users' views of the world. Our work instead explores visual conflicts *between* apps. Other efforts address input privacy, or preventing apps from accessing sensitive information in a user's environment (e.g., [4, 6, 7, 12, 14, 16, 17]) — our work is complementary.

7 CONCLUSION

Immersive multi-application AR platforms can enable users to interact with apps that simultaneously blend digital content into the physical world. However, AR apps may visually conflict with each other as they navigate the dynamically-changing environment of the user's world. In this work, we identify the challenges of mediating visual conflicts between apps without unduly infringing on their intended behaviors. We explore the design space of multi-app AR platforms and uncover key trade-offs presented by different design alternatives. We then analyze the design choices of current AR platforms and identify promising opportunities for future work. Our lessons lay a foundation to guide multi-application AR efforts, and we encourage future work to implement and evaluate the directions set forth in this paper.

ACKNOWLEDGMENTS

We thank Niel Lebeck and Earlence Fernandes for many helpful discussions and feedback on earlier drafts. We also thank our anonymous reviewers and our shepherd, Mahadev Satyanarayanan, for their valuable guidance and feedback. This work was supported in part by the National Science Foundation under Award CNS-1651230.

REFERENCES

[1] S. Ahn, M. Gorlatova, P. Naghizadeh, M. Chiang, and P. Mittal. Adaptive fog-based output security for augmented reality. In *Morning Workshop on Virtual Reality and Augmented Reality Network*, 2018.
[2] L. D'Antoni, A. Dunn, S. Jana, T. Kohno, B. Livshits, D. Molnar, A. Moshchuk, E. Ofek, F. Roesner, S. Saponas, M. Veanes, and H. J. Wang. Operating system support for augmented reality applications. In *HotOS*, 2013.
[3] J. Epstein, J. McHugh, R. Pascale, C. Martin, D. Rothnie, H. Orman, A. Marmor-Squires, M. Branstad, and B. Danner. Evolution of a trusted b3 window system prototype. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.
[4] L. S. Figueiredo, B. Livshits, D. Molnar, and M. Veanes. Prepose: Privacy, security, and reliability for gesture-based programming. In *IEEE Symposium on Security and Privacy*, 2016.
[5] https://www.theverge.com/2018/5/8/17332480/google-maps-augmented-reality-directions-walking-ar-street-view-personalized-recommendations-voting.
[6] 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 Symposium*, 2013.
[7] S. Jana, A. Narayanan, and V. Shmatikov. A scanner darkly: Protecting user privacy from perceptual applications. In *IEEE Symposium on Security and Privacy*, 2013.
[8] K. Lebeck, T. Kohno, and F. Roesner. How to safely augment reality: Challenges and directions. In *HotMobile*, 2016.
[9] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner. Securing augmented reality output. In *IEEE Symposium on Security and Privacy*, 2017.
[10] B. MacIntyre, A. Hill, H. Rouzati, M. Gandy, and B. Davidson. The argon AR web browser and standards-based AR application environment. In *ISMAR*, 2011.
[11] https://www.pokemongo.com/.
[12] N. Raval, A. Srivastava, A. Razeen, K. Lebeck, A. Machanavajjhala, and L. P. Cox. What you mark is what apps see. In *MobiSys*, 2016.
[13] F. Roesner, T. Kohno, and D. Molnar. Security and privacy for augmented reality systems. *Communications of the ACM*, 57(4), 2014.
[14] F. Roesner, D. Molnar, A. Moshchuk, T. Kohno, and H. J. Wang. World-driven access control for continuous sensing. In *ACM Conference on Computer & Communications Security*, 2014.
[15] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnaçao, M. Gervautz, and W. Purgathofer. The studierstube augmented reality project. *Presence: Teleoperators & Virtual Environments*, 11(1), 2002.
[16] R. Templeman, M. Korayem, D. J. Crandall, and A. Kapadia. Placeavoider: Steering first-person cameras away from sensitive spaces. In *Network and Distributed System Security Symposium*, 2014.
[17] J. Vilk, D. Molnar, B. Livshits, E. Ofek, C. Rossbach, A. Moshchuk, H. J. Wang, and R. Gal. Surroundweb: Mitigating privacy concerns in a 3D web browser. In *IEEE Symposium on Security and Privacy*. IEEE, 2015.