BonelessVR: A lightweight, high-productivity desktop environment for Google Cardboard

John Dean
deanjl@stanford.edu

Joan Creus-Costa*
jcreus@stanford.edu

Abstract

We developed a lightweight, smartphone-based, VR *environment for maximizing productivity and portability, which we call BonelessVR. The interface runs in the browser on a smartphone and is designed to be used with a bluetooth keyboard. In contrast with existing* VR *remote desktop environments that simply try to mimic the real world in the most realistic way possible, BonelessVR is includes a number of features that can't be attained with a physical monitor. These include high sensitivity, orientation deadzone, flat text rendering, and an integrated camera overlay. We show how these qualities offer improvement int text rendering and usability, and hope that these ideas are incorporated into the future of* VR *productivity.*

1. Introduction

Head-mounted displays (HMDs) have become a mainstream technology in the modern era. From the release of the Oculus Rift and HTC Vive in 2016, to the budget Google Cardboard, there have been a variety of affordable options for consumer HMDs for some time now. While they have made an impact on the gaming community, with notable triple-A titles that support VR, they have not made as much as an impact for uses that do not involve entertainment.

HMDs offer a unique opportunity for mobile work. In most office environments, people tend to prefer to have multiple, relatively large monitors for displaying large amounts of information at a given time while working. In contrast, laptop displays are typically small, and limited to as single display when used on the go. A user setting up at a coffee shop, or on a train, bus, or airplane, is typically limited to a single small display. HMDs offer a similar number of pixels as modern day laptop screens. However, this screen moves with the users head, and is able to keep track of orientation. This allows for the creation of a nearly unlimited amount of virtual screens, that the user can navigate in a natural way—by turning ones head. This allows for the possibility of creating the experience of multiple large monitors, while simply using the user's phone—housed in the HMD, rather than in the user's hands. For this project, created a solution that harnesses the hardware that the average person carries around everyday—their smartphone—to create a highly portable and productive environment, using Google Cardboard.

* This author contributed to the implementation of the project, but the report was written entirely by the other author

Figure 1: BonelessVR in use, showcasing out vision of the future of portable productivity.

There are many issues posed by current hardware limitations for an idea like this. The biggest limitation is that the angular pixel size is much larger to the user on current HMDs than a laptop screen. While there may be a similar number of pixels on modern day HMD as a laptop, these are evenly distributed over a larger field of view, and extend far beyond the fovea of the eye, causing a much larger angular pixel size. This can make displaying text clearly more difficult, creating a significant hamper to productivity. Another limitation is that virtual displays typically suffer from extra aliasing. This is because the pixel is rendered once on a virtual screen in a 3D environment, and then again when the camera in the 3D environment rasters the virtual screen to the HMD screen. After anti-aliasing is applied, this makes text blurry and difficult to read, and even with higher resolutions text would not look as crisp as it does on a native display.

The focus of this project is twofold: portability and productivity. Our setup is *portable* by nature of using a smartphone. Another feature we add ed is the addition of the phone camera as a background in the virtual display. They idea here is not that of AR—we aren't attemping to merge the real and virtual world. Instead, we want the user to be be using the virtual environment for work, while being aware of their surroundings. Consider a user walking on the sidewalk while using their smartphone. The digital worlds is distinct from the outside world–it is within the bezels of the display, yet they are able perceive both that the same time using peripheral vision. We've created a similar experience, just with more

screen space available to the user via virtualization of the display space within the HMD. Figure 1 showcases our project in use in this way.

With regards to *productivity*, a fundamental principle guiding this project is that no head-mounted display will replace a proper multi-monitor desktop in the near-term. As a result, unlike previous approaches, our goal is *not* to build a full desktop environment; instead, we want to reduce the gap between mobile and desktop productivity, and take advantage of what VR can do that reality can not. So, we created an interface that can be used to read and write with a minimalist interface. For further productivity, included support for an SSH terminal into a remote desktop.

While our end result may not look as impressive as previous work, it is in many ways better for productivity for the power-user, much like the Vim editor which is still widely used. The interface is not at a level that can compete with a laptop for general use, but rather can be optimized for productivity for certain tasks, and can be more effective that using just a smart phone natively.

2. Related Work

There have been a number of reports from independent sources that claim daily VR use will become commonplace in the future. A 2018 paper [10] discusses the office of the future, and the role of VR in that, and the authors envision a future office worker being able to work productively anywhere with head-mounted displays. However, a specific issue that it raises is that "[most] current HMDs are tethered and use external sensors/beacons for tracking, limiting the user to a small volume of operation." The use of higher quality VR headsets is appealing for immersion, but lacks practicality for everyday use on-the-go. One IEEE Consumer Electronics article [13] from 2016 claims that there will be an adoption curve similar to that of smartphones in the next seven to ten years. However, a 2017 study [11] demonstrated that while certain educational content was more enjoyable in with a Google Cardboard, it did not perform as well for readability and understandability when compared to typical media.

There has been much work into the area of visualized desktop environments for HMDs. There are a number of commercially available products available for Windows platform, such as [1] for the Oculus or [5] available for SteamVR. These apps are launched much like a video game, and are meant to be used with a gaming VR setup. There exists a similar tool for Linux [3]. The commonality between these such application is that they work on non-portable VR setups, and create fully emulated desktop environments. Each has a flashy product video, but none are built for actual functionality. While tech reveiwers typically give solid reviews to these services [8], it's a telling sign that such reviews do not have seemed to actually switch to using such services.

There has also been a fair amount of work into using the Google Cardboard for more niche applications, similar to what this project seeks to accomplish. One report [7] looked into creating an interface for immerse analytics, utilizing WebVR. Another [12] looked into the most usable navigation methods for VR with Google Cardboard, in which a Bluetooth device was the clear winner. A third paper [9] looked into exploring history utilizing the Google Cardboard. Our project looks to be similar to these—although with hopefully larger potential impact, as ours is a tool for productivity in a wide variety of tasks.

3. System

The system is made to work with a smartphone in a Google Cardboard and a Bluetooth keyboard. In Figure

Figure 2: A screenshot from BonelessVR

In this section, we detail how the system works, starting with how the viewport shows content, then describing the orientation tracking and how it translated the screen with a deadzone. We then discuss what types of content can be used with our setup, and what options the user can configure.

3.1. Viewport

BonelessVR operates with using a simplistic viewport that displays two layers of information: floating windows for displaying interactive content in the foreground, and a live video feed from the phone's camera in the background. As the users head turns, the viewport translates over the grid of nine content windows. The camera image is always fixed relative to the viewport. Figure 3 shows this layout for a single eye.

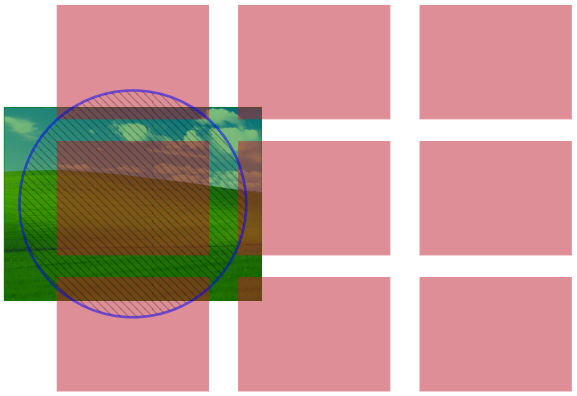


Figure 3: Layout for a single eye, while a user is viewing the middle-left window (zoomed out). The blue hatched circle shows the visible region of the screen to the eye. The green rectangle with the picture represents where the camera output is displayed. The red rectangles represent the 9 windows for content.

For displaying content to both eyes, the screen is split into

two regions, and and the viewport is shown to both eyes, with a spacing of d_{ipd}, the interpupillary distance of the user. However, there is an additional shift to the content windows, they are brought towards each other by a distance d_{verg}, to simulate the effect of looking at a monitor that is near the user's face. The camera image is at a distance d_{ipd} such the it appears to be far away from the user. The two eye view with this effect is detailed in Figure 4, and it creates a "2.5D" effect, in which flat content is displayed at different depths. The reticle and cursor are rendered on the same layer as the windows.

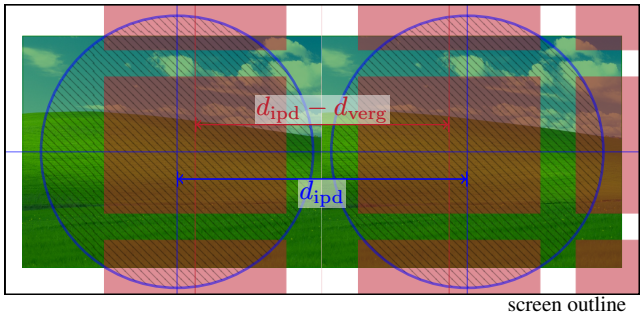
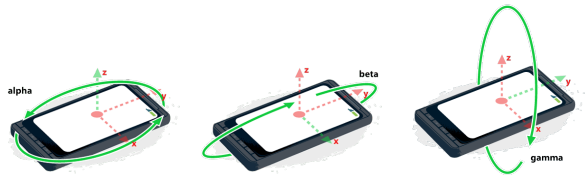
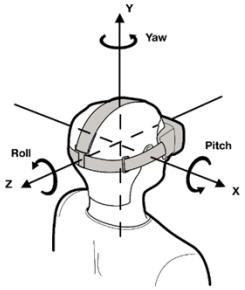


Figure 4: Screen rendering for both eyes. The color scheme is the same as Figure 3. Here, the difference in spacing of the windows compared to the camera background can be seen. The effect is exaggerated for this diagram.

3.2. Orientation



(a) Axes for Euler angles reported to the browser from the smartphone. The rotations are applied in alpha-beta-gamma order.



(b) Desired axes for Euler angles for the HMD tracking.

Figure 5: Differences between reported and desired coordinate frames for orenation.

As demonstrated by Figure 5, smartphones report orientation to the browser differently than is desired for HMD applications. As such, these angles need to be transformed into new angles that are consistent with standard HMD coordinates. The transformation from α, β, γ angles (from the phone) to ϕ, θ, ψ (yaw, pitch, roll for the HMD) is the following:

$$\phi \to -\beta \qquad \theta \to \gamma \qquad \psi \to \alpha$$

with the angles applied in ψ-θ-ϕ order after the transformation. We use this to convert to a quaternion to give the angle of the HMD, which we denote q_{phone}.

There are two more considerations to take into account. First, γ from the phone is zero when the device is flat horizontally, and for the HMD we want it to be on its side. If we were to simply offset this angle when transforming axes, this leads to sign problems when the view changes quadrants. Instead, we pre-multiply our orientation by a quaternion $q_{\text{quad}} = q(-90°, (1, 0, 0))$, a $-90°$ rotation about the x axis. Next, after a page refresh we want the content always centered to the current yaw location. As such, we use ϕ_0, the first yaw value recorded, to construct $q_{\text{center}} = q(\phi_0, (0, 1, 0))$, and pre-multiply the orientation by that, to give our final orientation, which we define q. In summary, we have

$$q = q_{\text{center}} \times q_{\text{quad}} \times q_{\text{phone}}$$

For bonelessVR, we don't rotate the display with the roll of the HMD, as we want to keep text aligned with the eyes. As such, from the q we compute the pitch and yaw angles only, which we will then use for computing pixel translations. We denote the final pitch and yaw output from the orientation tracking with $\phi_{\text{c}}, \theta_{\text{c}}$.

3.3. Viewport Translation and Deadzone

3.3.1 Computation

There is a user-configurable parameter, s, that gives the sensitivity of the orentation tracking of the display. Specifically, it gives the conversion between rotation angle of the HMD to pixel translations for the BonelessVR display. We define $p_{\text{new}} \in \mathbf{R}^2$ as the as the new pixel translation for each time that the orientation of the phone is updated, with

$$p_{\text{new}} = (x_{\text{px}}, y_{\text{px}}) = (\phi_{\text{c}}, \theta_{\text{c}})\, s.$$

We do not translate the viewport each time that we compute a new p_{new}, as we have a user-configurable deadzone with a parameter τ that gives the size of the deadzone (in pixels). We define p_{vp} as the actual translation of the viewport. When the page is first loaded, we set $p_{\text{vp}} = p_{\text{new}}$. To compute p_{vp} on subsequent updates, we first define $\Delta p = p_{\text{new}} - p_{\text{vp}}$, and then use the following equation to update the viewport translation at the n^{th} update, which we denote $p_{\text{vp}}^{(n)}$

$$p_{\text{vp}}^{(n)} = p_{\text{vp}}^{(n-1)} + \Delta p^{(n-1)} \left(\max\{0, \|\Delta p^{(n-1)}\| - \tau\} \right)$$

And so we can see that if $\|\Delta p^{(n-1)}\| - \tau \leq 0$, then $p_{\text{vp}}^{(n)} = p_{\text{vp}}^{(n-1)}$, and the viewport does not move.

There is one final step before translating the viewport, which is that we round p_{vp} to the nearest integer, so that the viewport only shifts a full pixel at a time, which is critical to prevent aliasing and ensure readable text.

3.3.2 Interface

In order to display the deadzone intuitively to the user, a reticle with two elements is used, detailed in the Figure 6 below. As the user turns their head, the cursor will move around in inside the deadzone circle. When cursor reaches the edge, it will pull the circle with it, and translate the viewport.

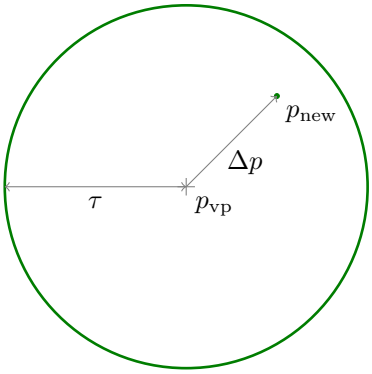


Figure 6: Reticle for displaying the deadzone to the user. Only the green is actually drawn to the screen. The points are labeled in the coordinates of the viewport. As the viewport is translated by $-p_{\text{vp}}$, the outer center of the outer circle remains on a fixed location relative to the screen, at eye coordinate $(0, 0)$. The cursor dot, is displayed at p_{new} in the viewport, and Δp on the screen

3.4. Window Types

To demonstrate the usability of BonelessVR, we implemented a few different type of windows to showcase how the system can be used, listed below.

1. Plain HTML: The simplest window type cnosidered supports plain simple HTML input. These are useful for showcasing just reading text, but nothing more....

2. Website `iframe`: We added support for websites to be displayed in HTML `iframes`. This allows for browsing simple websites, such as news articles, the course website, or software documentation pages. However, more complex websites are not supported, as there are many limitations imposed on `iframes` by web browsers for security reasons.

3. SSH terminal: We implemented a fully functional SSH terminal, so that the user can experience typing in VR, and use programs such as VIM.

4. Control Window: this is discussed in depth below.

3.5. Other Features

There are numerous other minor features and options available to the user the deserve brief mention.

Keyboard The primary source of user input for the system is a Bluetooth keyboard. By means of keyboard shortcuts, the user can click on windows, enter text to the SSH terminal, scroll with the arrow keys, as well as lock and reset the view orientation.

Window Focus Most windows support keyboard input of some form, and there needs to be a way to signify which window should respond to input. If the user clicks on a window, then it gets a red border and gains focus, so all keyboard input will be directed to the window.

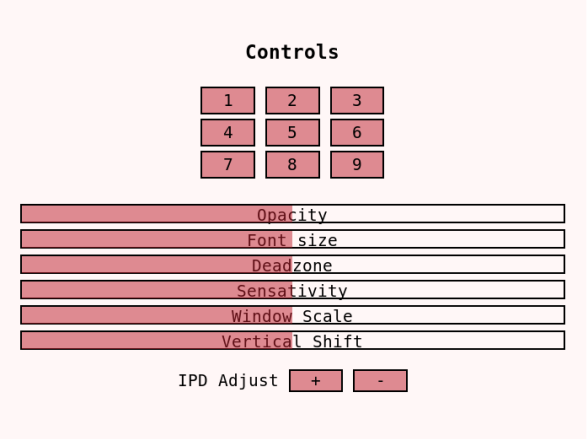


Figure 7: Control window, as seen by the user.

Control Window The control window, shown above in Figure 7. It's displayed on the bottom middle of the screen, and controls various aspects of the experience. The grid of numbered buttons can be used to toggle windows on and off. Below them are the following sliders.

- Opacity: This changes the background opacity of windows.
- Font Size: Scales the font of most text elements.
- Deadzone: Changes the size of the view deadzone. Under 10% will disable it.
- Sensitivity: Changes the how quickly the view pans with head rotation.
- Window Scale: Changes how big windows are.
- Vertical Shift: Shifts windows up and down vertically.

4. Implementation

4.1. General

The system was implemented entirely in Javascript and HTML. The first thing implemented was the dual viewport rendering which we build from scratch. The inability to duplicate elements in CSS was a significant hurdle, we have to duplicate elements by hand and keep them in sync. Any small discrepancies between eyes is very noticeable to the user, and this significantly limited the types of content we could display. By positioning the inner viewport absolutely, it is possible to shift all the windows by some number of pixels, as well as set them up in a grid. We shift this grid of windows around to create the moving viewport.

4.2. iframes

Due to security concerns, browsers severely restrict interaction between pages within different domains. This prevents us from doing things such as changing the font size of specific elements within iframes, changing backgrounds of iframes, or synchronizing text. So we were unable to use applications like slack. For future work, this could be fixed buy developing everything in custom app that would not have such restrictions.

4.3. SSH

We wanted to demo SSH, but javascript does not support arbitrary sockets and is therefore unable to directly support ssh. So, as a work around we use a gateway via a node server that uses socket.io [4] to relay the ssh protocol to the browser. socket.io uses WebSockets which provide realtime communications to the web. The code was adapted from an online example [2]. The front end uses the xterm.js [6] library to emulate terminals. In order to keep the terminals synchronized, the same callback from the socket writes to both terminals. Since in the SSH/Unix terminal pipeline characters are written via readback, this makes sure that they're kept in sync. Future work will try to use mosh rather than ssh, since its predictive system has been shown to be able to effectively mask latency in mobile environments. This is relevant because even with widespread LTE deployment phones are more prone to latency spikes, making interactive ssh programs more cumbersome to use remotely.

4.4. Sensors

To obtain orientation, we use the DeviceOrientation API to create a callback function whenever the orientation is updated to calculate new pixel offsets. No additional configuration is used, all orientation math is done on our side. To obtain a camera feed, we use the MediaDevices API. We configure it to use the back facing camera and target as high as a framerate as possible. The high framerate is necessary in order to provide a low-latency feed to the user and minimize disorientation or nausea while in motion.

5. Assessment

5.1. Usability

The creators found the BonelessVR interface to be quite comfortable and usable once complete. Text on the screen ended up remarkable readable, and reading websites was comfortable with the deadzone. The high sensitivity allows for easily switching back and fourth between pages without needing to turn the head far. The authors were able to effectively use SSH as if it were on a laptop.

5.2. Performance

The overall performance of the BonelessVR was quite good, it could run fine on a Moto x4, a mid-range smartphone from August 2017. However, there were some performance issues as we increased the complexity of and numbers of pages displayed in `iframes`. This is not a hard limitation, and could likely be mitigated with a more efficient software infrastructure, as modern smartphones are capable of having dozens of tabs open in memory.

The camera integration posed some issues, as having the camera active on a smartphone for long periods of time draws a considerable amount of battery. Combined with the fact that the screen is on and multiple webpages are being displayed, this ends up limiting the system to a battery life of 1–2 hours while in use. However, turning off the camera would considerably extend battery life, and the user should not need the camera for the entire time that they are using the system.

5.3. Portability Demo

To evaluate how usable the system was for use while not at a desk, the authors attempted using it while outside as a pedestrian. By attaching a keyboard to one's waist, we found that it was possible to interact with the system while standing. As a stress-test, we evaluated the setup while not only walking, but while on a skateboard and a bike. We found that after a few minutes of being acquainted with viewing the environment through the phone camera, we were able to comfortably use these transportation methods with experiencing the BonelessVR desktop. Photos from this test are shown below in Figure 8

Figure 8: Evaluation of the system while in transit.

5.4. Comparison

In order to assess our system, we created an separate 3D VR interface that is able to display floating panels of text, similar to how most existing desktop environments operate. This allowed for a direct comparison between our 2D text rendering and rendering onto a 3D canvas. We did not compare with existing products, as creating our own 3D rendered gave us the ability to fine tune the text rendering on the 3D system such that we could make a fair comparison. In both, we examined large blocks of the same size on screen, at a size that were near the readable limits of each system. The images can be found below in figure 9.

First, we can see that the BonelessVR interface is able to display more text. However, there is a very important additional effect here that is not captured in the images. While the pixels in BonelessVR for a given block of text only translate, and do not change relative to one another, the text is static and not a challenge to read. In the 3D environment, the text constantly "flickers" as the display angle changes slightly, making the text harder to read and causing discomfort for the user. We found it difficult to properly convey in the report, but even with a static image we still find the flat text to be an improvement.

We see that the 3D rendering is able to display 1113 words while remaining readable, while the BonelessVR is able to display 1271 words while remaining readable, a 14% improvement.

Figure 9: Comparison of rendered text between 3D (top) and 2D (bottom)

6. Discussion

In summary, we created a demo that showcases the potential for a highly portable and productive desktop VR environment. We showed that a simplistic "2.5D" text rendering system is highly usable and efficient, outperforms the standard practice of displaying a 2D screen in a 3D virtual world. We build up enough interface tools to lay out our vision for the types of features that should exist in VR desktop environments moving forward. We demonstrated how such a system could be used on-the-go, to allow the user to never need to be disconnected from their work, even when walking down the street.

Our system and evaluation of it has a number of limitations. In its current form, it is not capable of being more than a proof of concept. This is primarily due to software integration issues when building a browser-based application, that make it difficult to implement full web browsing capabilities or support all keyboard keys without certain keystrokes being interrupted by the phone's operating system. Additionally, the power consumption of the smartphone camera is a big issue for battery life, and for long term use of the system, one can not have a full-resolution high frame-rate video feed unless they carry an external battery.

Due to the the complexity of building the system mostly from scratch, limited time was available to evaluate our system in depth, and is an area to be expanded on. Future work on such a system should involve a more in-depth user study to evaluate working in such an environment. For example, trials of how much reading or writing users can get done on our system in comparison to a desktop or laptop, as well as a survey about eye stain.

7. Conclusion

Our project serves as a demonstration for the direction that we believe productivity-oriented VR should follow. The underlying philosophies of our design were simple: we don't try to replicate the experience of sitting at a desk and looking at a monitor. Instead, we strive to make something different and better, leveraging the features that can't exist in the physical world. VR will always have drawbacks when compared to real life, so with mere imitation we will always create something that isn't as good as the real thing.

Furthermore, we focused on developing the core features necessary for productivity. Rather than attempt to come up with a flashy demonstration, and gimmicky animations, we focused on functionality and way to enhance the experience for the power-user. While flashy demonstrations may sell units to consumers, unless the core product is actually better than alternative ways of doing worth, VR will never actually catch on for productivity.

From this demonstration that that even relatively low resolution displays can be used for productive work, we hope that further progress is made in this space, and that these elements be incorporated a widely available applications. We hope that our demo will urge others to think outside the box and implement new and exciting interface that change how humans interact with computers for doing productive work.

References

[1] Bigscreen Beta on Steam. https://store.steampowered.com/app/457550/Bigscreen_Beta/.

[2] Connecting to remote SSH server (via Node.js/html5 console). https://stackoverflow.com/questions/38689707/connecting-to-remote-ssh-/server-via-node-js-html5-console.

[3] Linux Gets An Open-Source VR Desktop, Built Off OpenHMD. https://www.phoronix.com/scan.php?page=news_item&px=Safespaces-VR-Desktop.

[4] Socket.io. https://socket.io/.

[5] Virtual Desktop. https://www.vrdesktop.net/.

[6] xterm.js. https://xtermjs.org/.

[7] P. W. Butcher, J. C. Roberts, and P. D. Ritsos. Immersive analytics with webvr and google cardboard. *Posters of IEEE VIS*, 2016.

[8] Devindra. Virtual Desktop for VR is a glimpse at a future without monitors, Jul 2016. https://www.engadget.com/2016/04/18/virtual-desktop-vr-windows/.

[9] A. Fabola, A. Miller, and R. Fawcett. Exploring the past with google cardboard. In *2015 Digital Heritage*, volume 1, pages 277–284. IEEE, 2015.

[10] J. Grubert, E. Ofek, M. Pahud, and P. O. Kristensson. The office of the future: Virtual, portable, and global. *IEEE computer graphics and applications*, 38(6):125–133, 2018.

[11] S. H. Lee, K. Sergueeva, M. Catangui, and M. Kandaurova. Assessing google cardboard virtual reality as a content delivery system in business classrooms. *Journal of Education for Business*, 92(4):153–160, 2017.

[12] W. Powell, V. Powell, P. Brown, M. Cook, and J. Uddin. Getting around in google cardboard–exploring navigation preferences

with low-cost mobile vr. In *2016 IEEE 2nd Workshop on Everyday Virtual Reality (WEVR)*, pages 5–8. IEEE, 2016.

[13] P. Rosedale. Virtual reality: The next disruptor: A new kind of worldwide communication. *IEEE Consumer Electronics Magazine*, 6(1):48–50, 2016.