First Experiences with GOOGLE GLASS in Mobile Research

There has been a long line of wearable research in the mobile computing community. However, a new, easily accessible platform often functions as a research enabler and leads to a burst of research activity in the field. Will Google Glass and similar offerings from other vendors have this catalyst effect on mobile and wearable research?

Photography, (this page) bigstockphoto.com

Viet Nguyen and Marco Gruteser *WINLAB/Rutgers University*

Editors: Marco Gruteser, Sharad Agarwal

Google Glass, the epitome of wearable displays, seems poised to become the most widely available wearable interaction device for the mass consumer market. Thanks to its compact design, rich sensor equipment, and growing API support, Glass also represents an exciting platform for researchers in the mobile field. While wearable computing research has a long tradition [1, 2], such research was conducted with custom hardware arrangements. The availability of a convenient, easy to use hardware platform often leads to heightened research productivity.

There are also other devices of head-mounted wearable devices available. For example, the Epson Moverio BT-200 [3] provides a full Android experience, with a transparent display that hovers approximately four feet in front of users. While Google Glass primarily aims to be a notification device with the screen appearing in the corner of the right eye, Epson glasses can create a 3D display. The Recon Jet device [4] offers navigation, weather, social media, SMS, call info, web connectivity, and more. With GPS functionality and onboard sensors that measure speed, distance and elevation gain, this device targets athletes. Epiphany Eyewear [5] provides only one function: record video. GlassUp [6] is able to read texts and emails, tweets, Facebook updates and other social networks. These devices also deserve full consideration but we will focus our following discussions on Google Glass, which arguably offers one of the richest APIs for development.

In this article we report on our first experiences with Google Glass from a mobile systems researcher's perspective. It does not intend to report on any specific research activity, but simply aims to provide an overview of the capabilities and limitations that researchers are likely to encounter. We begin with an overview of the hardware and discuss what APIs Google Glass offers developers, before we report on performance characteristics and experiences.

A BASIC OVERVIEW OF THE HARDWARE

Currently Google Glass is distributed to developers in the form of an Explorer version. The detailed custom user interface and hardware specifications are listed in Table 1 and are further described below.

User Interface

The biggest differences to smartphones are arguably the custom designed user interfaces for Glass. In addition to audio output, Glass provides its well-known display inside the person's field-of-view as well as touch, voice, and gesture input.

• **Display:** The main display of Google Glass has a resolution of 640x360, equivalent to a 25-inch high definition screen from eight feet away. It is a Liquid Crystal on Silicon (LCoS) field sequential color, LED-illuminated display. As Google Glass is designed to be a no-distracting notification device, its screen is not on all the time. Instead, it is only held on for a few seconds before it automatically turned off until users reactivate it.

• **Touchpad:** The capacitive touchpad is located on the right side of Google Glass. Users can tap on the touchpad to wake Glass up. Users can also control the device by swiping the touchpad to navigate a timeline-like interface on the screen, as well as choose options on each timeline card.

• **Voice Input:** This is the second, hands-free, method of input. User can trigger predefined action through keywords. Users can also dictate emails or messages. Note that voice dictation function is performed on remote server, so this function is not working off-line.

• **Head Gestures:** Glass can also be activated by tilting the head backward ('looking up'). It can also detect when the user puts on the device.

Sensors

• **Ambient light sensor:** This sensor serves a similar purpose as on smartphones; it enables automatic control of display backlight from a dark environment to direct sunlight. Its data (in SI lux units) can be read through Android API.

• **Inertial and compass sensor:** According to a Google Glass teardown [7], it has an InvenSense MPU-9150 sensor [8], which includes a 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer (digital compass). The gyroscope has a user-programmable full-scale range of ±250, ±500, ±1000 and ±2000o /sec. The accelerometer has a programmable full-scale range of ±2g, ±4g, ±8g and ±16g. The magnetometer has output data resolution of 13 bit (0.3µT per LSB) and the full-scale measurement range is ±1200µT. The sensor is used for providing built-in functions such as the compass and detection of the 'Lookup' head gesture that activates the display. It is also available for Glass app use and could therefore support many more functions. A notable difference to smartphone sensors is that this sensor moves together with a person's head. It could therefore track information about head movement and in

TABLE 1: Hardware specifications

Camera	5 MP for photos and 720p for videos
Audio	Bone Conduction Transducer
Processor	Texas Instruments OMAP 4430 SoC 1.2Ghz Dual (ARMv7)
Connectivity	802.11 b/g and Bluetooth 4.0 (with BLE support). Able to connect to Internet by Wi-Fi, or by tethering through smartphone.
Storage	12 GB of usable memory, synced with Google cloud storage. 16GB Flash total
Battery	Single-cell Lithium Polymer battery, roughly 570 mAh. [7]

which direction a person is looking.

• **Proximity sensor:** This is the sensor used to measure the distance to user face and eye (in centimeters). It can detect when a user puts Google Glass on his head, and when a user "winks".

PROTOTYPE DEVELOPMENT WITH GLASS

At the time of writing, Google Glass is running on Android 4.4 (API 19). Therefore, researchers with experience in prototyping Android smartphone apps would have an easy transition to Google Glass prototyping. There are two API options for developers to develop Glassware. The first one, Mirror API, allows you to build web-based services that interact with Google Glass. These services can be programmed in Go, Java, PHP, .NET, Python or Ruby. Google Glass is synced with Mirror API, and the web service can send notifications or receive user options through Mirror API. For example, a New York Times Glassware would periodically send brief news to Mirror API, and Mirror API will automatically deliver these content to user's Google Glass. When the user wants to read news in detail, his option will be recognized by Mirror API, and it in turns sends a request to the web service for the full content. This is the advantage of a web service using Mirror API: it only needs to deliver content (in JSON format), then leaves all Google Glass built-in functionalities to Mirror API.

The second API, Glass Development Kit (GDK), enables richer Glassware complete with interactive features and access to some of the hardware features. It is an Android SDK add-on that contains APIs for Glass-specific features: voice control, gesture detector (such as head on, "wink" detection, etc.), or timeline cards. Also, Google designed the Glass platform to make the existing Android SDK work immediately on Glass. This lets developers code in a familiar environment, but for a uniquely novel device. GDK is used when you need real-time user interaction, offline functionality, and access to hardware.

Although experience researchers would have no difficulty in getting used to the Google Glass development environment, they should take into account the unique user interface of Google Glass by adopting these new design pattern building blocks:

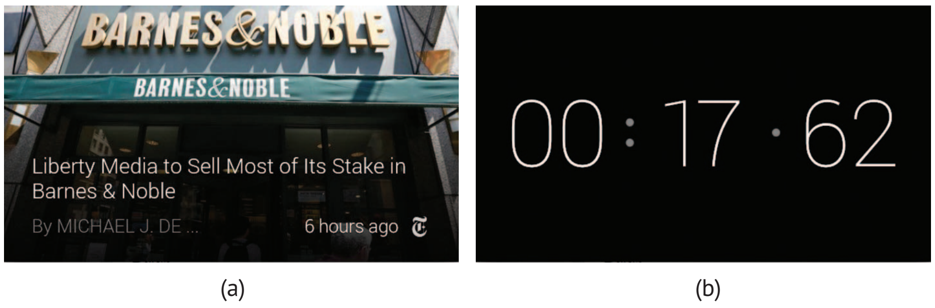

FIGURE 1. (a) Static card: New York Times. (b) Live card: Stop watch

• **Timeline:** The timeline is the main user interface that is exposed to users. It presents live and static cards, performs voice-commands, which is a common way to launch Glassware. Timeline is arranged into three parts: at the center there is Glass clock, then on the right there are static cards delivered by Mirror API, while on the left there are currently running live cards.

• **Static cards** (Figure 1a): These are the main components for displaying texts, images, and video content. They are produced by Mirror API and added to the timeline. Their main usage is to provide periodic notifications to users as a specific event happens, such as when users arrive at a specific location.

• **Live cards** (Figure 1b): Different from static cards, live cards can update frequently to provide real-time information to users. Example applications include timer, compass, etc. Live cards also have access to low-level sensors, such as the accelerometer, gyroscope, and magnetometer. In addition, they run inside the timeline, so users can navigate to other cards when the live card is running.

• **Menu options:** These options can be called from each card. They carry out actions associated with the card, such as sharing a video, replying to an SMS, deleting an image, etc. Google Glass is still in an experimental phase, and APIs have been gradually updated for other functionalities. At the time of writing this paper, the APIs offer ways to control camera, get voice input, access GPS data and inertial sensors, including gyroscope, accelerometer, and magnetometer. In addition, it is also possible to communicate with another Android smartphone through a Bluetooth connection. One missing feature in the API is an accessing proximity sensor, which directly controls the "wink" gesture. Google Glass has an experimental feature that allows users to use the "wink" gesture to quickly capture a photo, but Google has not offered an API for developers to use the "wink" gesture as another input method.

PERFORMANCE

Here we provide a basic performance characterization in terms of battery lifetime, computational power, and network throughput and compare it to Android phones. Note that these stress tests are clearly not representative of typical or recommended use of Glass. Since researchers are always pushing the boundaries, we hope, however, that they provide insight on what research projects Glass can support.

• **Battery lifetime:** We run Google Glass with continuous video recording, continuous display use, and continuous sensor use and measure how long its battery lasts in each test. The results are shown in Table 2. In the new API version XE16, sensor data logging is optimized, therefore the time duration is better than the previous version XE12. The number in parentheses shows how the new battery lifetime compares to the previous API version.

• **Computational limits:** We use the LinPack benchmark for comparing performance of Google Glass and several other Android devices, based on MFLOPS (Mega floating-point operations per second). For each device, we run the experiment 20 times for single-thread and multi-thread case, and record the average value and deviation. The result is shown in Figure 2 (the higher the average value is, the better performance the device has). As can be seen, Google Glass' computation performance isn't as high as that of normal smartphones, which is expected for a notification device rather than a computing platform. Therefore, the device is not designed to process heavy load applications, such as image processing

TABLE 2: Battery lifetime test

Scenario	Duration (in mins)
Continuous Camera Use	60
Continuous Display Use	65
Continuous Sensor Use	240 (310)

TABLE 3: Network throughput test

Device	TCP bandwidth (in Mbps)	UDP packet loss rate
Google Glass	24.9	0.07%
Nexus 5	45.1	1.58%

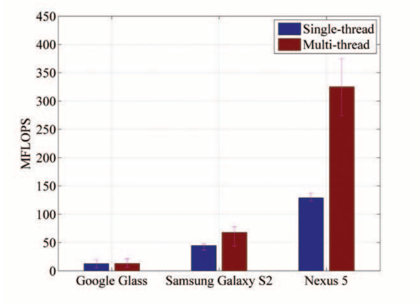

FIGURE 2. Computation benchmark

or computer vision. The recommended way to perform these tasks is uploading images or videos to some cloudlet or cloud, having some dedicated computers or servers running the algorithms and then returning the results to Google Glass.

• **Network throughput:** We use the iPerf for Android network benchmark tool [9] for comparing Wi-Fi network throughput of Google Glass and Nexus 5. In these tests, the Android devices act as the server and a laptop acts as the client. The Wi-Fi version of Google Glass is 802.11 b/g, while the Nexus 5 supports Wi-Fi 802.11 a/b/g/n/ac. Table 3 shows the results for two tests: TCP bandwidth and UDP packet loss ratio. These results are likely due to the more advanced Wi-Fi versions in the Nexus 5.

FIRST EXPERIENCES AND PARTING THOUGHTS

Glass looks well-suited to study many challenges in Human Computer Interaction, Augmented Reality, and Positioning Systems among others. It provides an interface device that can be operated hands-free and adds sensors that look useful for various forms of head and gaze tracking.

We found transitioning from Android smartphone to Glass software development straightforward. One should keep in mind that battery and computational resources are more limited and many tasks such as image processing, speech recognition are best offloaded to the cloud. Second, for immediate interactivity, such as accessing and processing sensor data in real-time, the GDK should be used instead of the Mirror API. We also noticed that Google Glass could get quite warm under continuous load (such as video recording and sensor logging). We therefore found it useful to conduct much of our development and testing using screen casts and external input, without actually wearing the device.

While Google Glass is a relatively new HMD device, a few researchers have already conducted research with this platform. For example, in Pedersen and Trueman's article, "sergey brin is batman" [10], the authors argue that Google Glass has instigated the adoption of a new paradigm in Human and Computer Interaction. A 2013 article by Simoens, Verbelen, and Dhoedt [11] introduces Mercator, a distributed system that builds 3D maps of the world from crowd-sourcing data provided by depth-cameras mounted on HMDs like Google Glass. In the Augmented Reality domain, many applications, such as Word Lens [12] or Layar [13], are moving from smartphones to Google Glass, making them more useful and context-aware. Besides, Roesner, Kohno, and Molnar's recent ACM article [14] shows that although Augmented Reality work using Wearable devices like Google Glass is still young, it is the right time to carefully consider issues such as security and privacy. This article also proposes several novel applications, such as encrypting content in the real world or managing passwords.

Will Glass fuel wearable research just as smartphones have led to a wealth of mobile research? We suspect that this will largely depend on the commercial success of Glass. Glass does provide, however, an easily programmable platform for wearable research. ■

REFERENCES

[1] S. Mann, "Wearable computing: a first step toward personal imaging," *Computer*, vol. 30, pp. 25–32, Feb 1997.

[2] T. Starner, S. Mann, B. Rhodes, J. Levine, J. Healey, D. Kirsch, R. W. Picard, and A. Pentland, "Augmented reality through wearable computing," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 386–398, 1997.

[3] "Epson Moverio BT-200." http://www.epson.com/cgi- bin/Store/jsp/Landing/moverio- bt-200- smart- glasses.do.

[4] "Recon Jet." http://www.reconinstruments.com/products/jet/.

[5] "Epiphany Eyewear." http://www.epiphanyeyewear.com/.

[6] "GlassUp." http://www.glassup.net/.

[7] "Google Glass teardown." http://www.catwig.com/google- glass- teardown/.

[8] "Invensense MPU-9150 Product Specification." http://www.invensense.com/mems/gyro/documents/PS- MPU- 9150A- 00v4_3.pdf.

[9] "iPerf for Android." https://play.google.com/store/apps/details?id=com.magicandroidapps.iperf.

[10] I. Pedersen and D. Trueman, ""sergey brin is batman": Google's project glass and the instigation of computer adoption in popular culture," in *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, (New York, NY, USA), pp. 2089–2098, ACM, 2013.

[11] P. Simoens, T. Verbelen, and B. Dhoedt, "Vision: Mapping the world in 3d through first-person vision devices with mercator," in *Proceeding of the Fourth ACM Workshop on Mobile Cloud Computing and Services*, MCS '13, (New York, NY, USA), pp. 3–8, ACM, 2013.

[12] "Word Lens for Google Glass." http://questvisual.com/us/.

[13] "Layar for Google Glass." https://www.layar.com/glass/.

[14] F. Roesner, T. Kohno, and D. Molnar, "Security and privacy for augmented reality systems," *Commun. ACM*, vol. 57, pp. 88–96, Apr. 2014.