A HoloLens Application to Aid People who are Visually Impaired in Navigation Tasks

Jonathan Huang

Advisors:
Prof. Emily Cooper
Prof. Wojciech Jarosz
A HoloLens Application to Aid People who are Visually Impaired in Navigation Tasks

Jonathan Huang

Abstract

People with impaired vision have trouble with day-to-day activities such as navigation and reading. Reading text on signs may be particularly difficult for the visually impaired because signs are typically in dynamic environments. Indoors, signage may include office, classroom, restroom, and fire evacuation signs. Outdoors, they may include street signs, bus numbers, and store signs. Depending on the level of visual impairment, just identifying where signs exist can be a challenge. Using Microsoft’s HoloLens, an augmented reality device, I designed and implemented the TextSpotting application that helps those with low vision identify and read indoor signs so that they can navigate text-heavy environments. The application can provide both visual information and auditory information. In addition to developing the application, I conducted a user study to test its effectiveness. Participants were asked to find a room in an unfamiliar hallway. Those that used the TextSpotting application completed the task less quickly yet reported higher levels of ease, comfort, and confidence, indicating the application’s limitations and potential in providing an effective means to navigate unknown environments via signage.

1 Introduction

According to the World Health Organization (WHO), in 2014 there were approximately 285 million visually impaired people in the world [51], and this number is increasing. In fact, in the United States, researchers expect that the number of visually impaired people will double to more than 8 million by 2050 [48]. However, there are many different levels of visual impairment, ranging from moderate (low vision) to severe (blindness). Of the estimated 285 million people with impaired vision in the world, 246 million people have low vision and 39 million people are legally blind [51]. While low vision refers to a condition where the best-corrected vision is impaired enough to interfere with day-to-day activities, blindness refers to either visual acuity that is worse than 20/200 or a visual field of 20 degrees or less [32]. Although many people with low vision maintain some visual function, they each have trouble with day-to-day activities, sometimes even risking injury because of their visual impairments [22].

Independent mobility and reading are two of the most important activities for the visually impaired because they are both necessary for an individual to fully participate in society. In fact, Elliott et al. [11] analyzed 4,744 low vision examinations and found that the primary objectives of elderly patients with low vision are reading and moving around, while common secondary objectives include watching television. Unfortunately, reading and navigation are also two difficult activities for the visually impaired. In a survey of 445 low-vision patients who were asked to rate the difficulty of 24 tasks (e.g. cooking, playing sports), recreational reading was the most difficult activity for low-vision patients [28]. Additionally, the repercussions from these impairments go far beyond just the inability to independently navigate and read. For instance, reduced mobility may result in depression due to social isolation [33, 38].

In the next sections, I review prior literature on devices that aid the visually impaired in exclusively reading or navigation. However, it is important to note that in addition to being independently difficult tasks, reading and navigation often go hand in hand: identifying and reading informative signage is often an important component of navigation.
1.1 Readings Aids

Many assistive devices have been invented to aid reading. Although the magnifying glass continues to be one of the most recognizable devices, it is limited in its effectiveness because users must position themselves at an appropriate distance to read the text. Thus, they must first locate text without assistance (a difficult task in itself), and they must be close enough to the text (a condition not always possible when navigating through an unknown environment). As a result, other assistive devices have been developed, such as the closed-circuit television (CCTV) and head-mounted video magnifiers.

![A photograph of a user using a head-mounted magnifier.](image)

As shown in Figure 1, the CCTV consists of a large monitor and a camera above a platform where the text is placed. The camera, which has a zoom lens, captures the text and displays the magnified image onto the screen. To examine the text, the reader simply moves the paper on the platform. Although the CCTV makes it easier for users to zoom in and out of text, it is not portable. To solve this issue, researchers created the head-mounted video magnifier (head-mounted CCTVs) that users can wear around. After comparing the effectiveness of head-mounted video magnifiers, CCTVs, and large print, Ortiz et al. [34] concluded that head-mounted video magnifiers aid low-vision reading performance. However, the American Foundation for the Blind does not recommend head-mounted video magnifiers for mobility because of the reduced field of vision caused by magnifying the user’s vision [49].

Despite these improvements, some assistive devices continue to be designed for stationary settings where the user can sit down and read a piece of text at his or her fingertips. Reading signs and other text that are not printed on paper remain a difficult task. While traditional readings aids are limited by the fact that they are stationary, augmented reality systems, because of their portability, has the potential to help users read signs while moving around.

1.2 Mobility Aids

Researchers often separate mobility into two types: obstacle avoidance and spatial navigation (or “wayfinding”). While obstacle avoidance deals with walking without hitting any objects, wayfinding refers to the ability to follow routes or navigate to a desired location.

1.2.1 Obstacle Avoidance

Many people with low vision effectively avoid obstacles using a white cane or a dog guide. In addition to these tools, some researchers have also created specialized head-mounted devices
such as stereo vision aids that substitute visual information with auditory information. Systems like Auditory Augmented Reality [41], Electronic Travel Aid [25], CASBLIP [13], and Stereo Vision based Electronic Travel Aid (SVETA) [7] use stereo cameras on a headgear to capture images and convert pixel information into sound information. While the Auditory Augmented Reality device leverages 3D sound source localization, the Electronic Aid uses a USB webcam and face detection to send beep sounds, and the CASBLIP detects objects through sensors and orients the user using global positioning system (GPS). The SVETA, as shown in Figure 2, calculates distance to an object using a stereo algorithm and communicates distance to the user using sonification with high frequency tones signaling the top portion of the image and low frequency tones signaling the lower portion of the image. Despite these advancements, these devices have a few issues. First, after repeated training on the SVETA, users were only able to recognize simple shapes such as triangles and squares when they might want to identify shapes that are more complex. Second, many of these devices are rather bulky. They all require large helmets, and the SVETA requires the user to wear an additional pouch. Lastly, the inherent difficulty with translating visual information to auditory information might limit the adoption of any stereo vision aid. Although obstacle avoidance is extremely important, the goal of the TextSpotting application is to help people who are visually impaired navigate an environment.

![Figure 2: An example of a sonification assistive device to avoid obstacles. Reproduced from Balakrishnan et al. [7]. (a) SVETA Prototype System, which uses stereo cameras on a headgear to capture images and convert pixel information into sound information. (b) a photograph of a blind user wearing the SVETA.](image)

Rather than using sensory substitution, others have developed navigation aids to take advantage of a user’s residual vision. Hicks et al. [18] developed a real-time head-mounted visual display with a depth camera to detect distance to nearby objects as shown in Figure 3, and Van Rheede et al. [47] iterated on this design by creating residual vision glasses (RVGs) that use an infrared depth camera mounted on a headset. The RVGs consist of organic light emitting diode (OLED) panels that convey distance by changing the brightness of the panels. Much like the SVETA, however, the RVG must be carried around with a computing device (a Thinkpad X220 laptop that weighs 4 pounds). Additionally the displays on the glasses have a maximum resolution of 160 x 128, and the displays are opaque. As a result, users lose any information from their residual vision as the device completely covers a user’s visual field with OLEDs that only convey distance.
1.2.2 Wayfinding

Although considerable work has been done to aid the visually impaired in obstacle avoidance, less is known about wayfinding. Nevertheless, all wayfinding systems must have some form of localization to determine the user’s position and/or orientation. Fallah et al. [14] grouped localization methods into four general categories: (1) dead reckoning, (2) direct sensing, (3) triangulation, and (4) pattern recognition. The following sections discuss these techniques (see also Table 1 for a summary).

<table>
<thead>
<tr>
<th>Technique</th>
<th>Methods</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dead reckoning</td>
<td>Fischer and Muthukrishnan [16], Koide and Kato [23]</td>
<td></td>
</tr>
<tr>
<td>Direct sensing</td>
<td>RFID</td>
<td>Sanchez et al. [43], Murad et al. [31], Yelemarthi et al. [53], Kulyukin and Nicholson [24]</td>
</tr>
<tr>
<td></td>
<td>IR</td>
<td>Jain [20], Baus et al. [8]</td>
</tr>
<tr>
<td></td>
<td>Ultrasound</td>
<td>Lee et al. [26], Ran et al., [40]</td>
</tr>
<tr>
<td></td>
<td>Barcode</td>
<td>Legge et al. [27], Kalia [21], Al-Khalifa [5], Coughlan et al. [10]</td>
</tr>
<tr>
<td>Triangulation</td>
<td>GPS</td>
<td>Lee et al. [26], Yelemarthi et al. [53], Roentgen et al. [42]</td>
</tr>
<tr>
<td></td>
<td>Wi-Fi</td>
<td>Paisios et al. [35], Rajamaki and Viinikainen [39]</td>
</tr>
<tr>
<td>Pattern recognition</td>
<td>Computer vision</td>
<td>Silapachote et al. [45], Hub et al. [19], Ran et al. [40]</td>
</tr>
</tbody>
</table>

Table 1: Overview of localization techniques used in different wayfinding systems. The left-hand column lists techniques used in wayfinding, the middle column lists implementations of those techniques, and the right-hand column shows papers that are representative of each implementation. Abbreviations include: radio-frequency identifier description (RFID), infrared (IR), global positioning system (GPS).

**Dead reckoning** Dead reckoning systems estimate a user’s position based on previously determined positions. These systems typically use various sensors like accelerometers, gyroscopes, and compasses to incrementally calculate a user’s position and/orientation. Using simulation results, Fischer and Muthukrishnan [16] tests a navigational aid that uses foot-mounted inertial sensors while Koide and Kato [23] proposes an aid based on accelerometers and gyroscopes. Because location is estimated incrementally, however, devices that rely on dead reckoning may lead to errors that magnify over time. Additionally, getting the initial position requires other localization techniques such direct sensing, which is described in the next section.

**Direct sensing** Direct sensing methods determine location by sensing tags that have been previously installed in the environment. These tags can contain location data internally, or sensors can retrieve data from an external database after recognizing a unique tag ID.
Generally, four types of tags have been developed: (1) radio-frequency identifier description (RFID), (2) infrared (IR), (3) ultrasound identification, and (4) barcodes.

RFID tags are one of the most commonly used types of tags [24, 31, 43, 53]. Typically, systems that use RFID tags must install tags into the building before the user navigates the environment. Users hold an RFID reader, and when the reader senses an RFID tag, the system knows the approximate position of the user and relays that information to the user. Sanchez developed a cane that detects RFID tags and uses these tags as checkpoints to a destination. Once the reader detects an ID, the microcontroller on the cane sends the ID to a laptop, and the laptop relays an audio output to the user’s headset, directing the user on where to go next. Users, however, must first enter their destination on the laptop before they use the cane. In general, systems using RFID tags have a few disadvantages. RFID tags must first be installed in a building before the system can be used, and this installation can be costly in large environments. Once installed, the RFID tags can be difficult to update if the environment changes as maintenance and update costs may be high. Additionally, humans can block radio-frequency signals, making it difficult for sensors to detect RFID tags [6].

Similarly, infrared (IR) localization relies on installed IR transmitters that broadcast a unique ID [8, 20]. Once an IR receiver detects an IR transmitter, the receiver relays the data to the user. Jain [20] developed a system that uses a mobile phone to detect IR signals coming from previously installed IR wall units in the building. In addition to the IR data, an accelerometer captures the distance traveled by the user, and the mobile phone conveys this data to the user by displaying large text on the cell phone and by using a Text-to-Speech (TTS) engine. Much like RFID systems, however, IR tags must be previously installed into the environment, and users must interact with an interface to select their destination. IR systems have the additional disadvantage of being difficult to detect. Infrared requires a line of sight, and light can interfere with infrared waves [30].

Ultrasound identification, on the other hand, uses ultrasound emitters to emit short wavelengths that receivers can detect [26, 40]. One disadvantage of ultrasound sensors, however, is that walls may reflect or block ultrasound signals, leading to less accurate predictions of location.

Lastly, barcodes can act as identifiers to localize users [5, 10, 21, 27]. Like all other direct sensing techniques, infrastructure must already be in place for the user to use the application. In this case, users typically carry a barcode reader and manually scan the barcodes along their path. Legge et al. [27] developed the Digital Sign System, a hand-held reader that detects 2-D matrix barcoded signs posted at room entrances using an infrared camera. With synthetic speech, the mobile device communicates information to the pedestrian. If the floorplan is loaded into the system, the Building Navigator (the software used by the Digital Sign System) can verbally describe the surrounding area audibly to the user [21, 27]. Barcode systems, however, require others to design and pre-install significant infrastructure into the environment. Additionally, users must either find each barcode or be close enough to a barcode for the scanner to read the tags.

Assistive aids, however, are not restricted to one direct sensing technique. SmartVision, for instance, takes advantage of GPS, Wi-Fi, and radio-frequency identification (RFID) information to detect specific landmarks in an environment and provide navigation instructions based on the angle between the user’s current direction and the desired landmark [15]. However, Fernandes and Muthukrishnan [15] has not tested this device.

**Triangulation** While direct sensing localization techniques usually position a user after sensing one tag, triangulation techniques typically use multiple identifiers to triangulate a user’s location. Global positioning system (GPS) and wireless local area networks (WLAN) triangulation are two such examples.

In GPS localization devices, satellites send a periodic signal, and receivers on the device use these signals to triangulate a user’s position. Many have developed navigational aids that use GPS [26, 53], and Roentgen et al. [42] evaluated four such devices, concluding that each is useful in its own way. The main disadvantage with GPS, however, is that it can only be used outdoors, where GPS signals are strong enough. Once the user is inside or between tall
buildings, GPS signals become unreliable.

WLAN localization systems, on the other hand, use data from wireless base stations to triangulate the location of a user. The LaureaPOP Indoor Navigation Service, for instance, uses WLAN technology on a mobile phone to pinpoint a user’s position [39]. In general, however, WLAN techniques have a lower precision than GPS because of reflection issues, and WLAN technologies rely on strong signals for the technology to be accurate.

**Pattern recognition** Pattern recognition techniques typically use data from sensors and compare that data with known location in an environment. In computer vision based techniques [19,40] of pattern recognition, users navigate through an environment with a camera. The camera captures images of the environment and compares the images to a database of known images and locations. For instance, the Visual Integration and Dissemination of Information (VIDI) system detects signs in the environment using support vector machine classifiers on color and texture features [45]. After detecting a possible sign, the VIDI tries to match the sign with a database of known signs and if matched, voice synthesizes the sign to the user. One disadvantage of this technique is that it requires significant computing power to run computer vision methods and match them with known images.

All four of these wayfinding techniques rely on preprocessing the environment so that sensors can match data with known locations. Although dead reckoning techniques estimate location based on changes in localized sensor data, they still rely on knowing the initial position of the user. Normally sighted individuals, however, do not always navigate in the same way. When navigating, they often find their destination by incrementally reading signs in the environment without comparing those signs with a map. Navigation applications that can read signs in real time without the need to preprocess the environment can, thus, help bring the visually impaired one step closer to navigating environments in the same way normally sighted individuals do. Because they process the environment in real time and do not rely on special infrastructure, augmented reality systems have a distinct advantage compared to many of the systems described above.

### 1.2.3 Other Assistive Devices

In addition to devices aimed to help the visually impaired in wayfinding, multipurpose commercial solutions have also been developed to help people understand their environment and perform typical tasks. eSight is a $15,000 pair of smart glasses that captures video and displays a processed version to the user on an OLED screen in real-time [1]. A controller allows the user to adjust zoom, contrast, and brightness of the video. Horus is a wearable assistant that recognizes objects and describes those objects to the user using a pocket unit and an earphone [2]. It can also perform text-to-speech; however, users must first detect the presence of text before the system can recognize the text. OrCam, like eSight, uses a smart camera [3]. Unlike eSight, however, OrCam detects objects and translates all visual data to audio data for the user. Lastly, Visionize is a headset that uses a Samsung Galaxy S6 to magnify parts of images in a scene [4]. When wearing the headset, a magnification bubble appears in the center of the user’s vision, and the user can zoom in or magnify the image in the bubble.

Others have also developed devices to recognize objects or translate signs in specific environments. Taking advantage of the planarity of grocery objects and the typically static placement of groceries, the ShelfScanner detects and recognizes groceries by using a hand-held camera to capture video and a multiclass naïve-Bayes classifier to detect grocery items [50]. Like many other aids, however, the camera must be connected to a powerful laptop that must be carried on the user’s back. Yang [52], on the other hand, developed a head-mounted system that translates signs from Chinese to English. Although these devices are not specifically used to help the visually impaired navigate certain environments, they do provide insight into object and text detection, which is necessary for navigation aids.
1.2.4 Augmented Reality

While researchers and companies have created specialized devices for the visually impaired, the recent development of consumer-grade augmented reality has opened new doors to potential solutions. In contrast to virtual reality devices where the user’s entire environment is computer-generated, augmented reality devices overlay holograms or computer-generated objects onto the user’s view of reality in real time. Holograms are responsive to the world around them, and users can interact with holograms by using simple gestures or voice commands.

Microsoft’s HoloLens, more specifically, is an untethered, consumer-grade augmented reality device (see Figure 4) that, although currently only available to developers, is intended to be available to all consumers. The HoloLens includes speakers, a 2.4 megapixel (MP) camera, and a depth camera. These two cameras allow the HoloLens to construct a representation of the environment by detecting real-world surfaces, and this representation is called a spatial map [29]. In addition to its hardware components, the HoloLens has a custom 32-bit Microsoft Holographic Processing Unit (HPU), 2 gigabytes (GB) of RAM, and 64 GB of storage, and 2-3 hours of active battery life, allowing the HoloLens to run as a standalone computer. It does not need an additional computing machine to run HoloLens applications. To interact with these applications, the HoloLens has gesture recognition, voice control, Wi-Fi capabilities, and Bluetooth capabilities.

Because the HoloLens is a general platform for everyone, the HoloLens provides a promising solution to those with low vision who seek to not only have many assistive applications in one place but also wear a device not specifically associated with low vision. However, it is important to note that the current design of the HoloLens is slightly heavy at 1.2 pounds and fairly conspicuous although this could change in the future.

Thus, many solutions have been developed to help people with impaired vision exclusively read or wayfind. The HoloLens, however, is a unique platform because it has the potential to help the visually impaired with both reading and wayfinding using augmented reality. In the next sections, I will describe the TextSpotting HoloLens application developed to help the visually impaired navigate text-heavy environments and discuss a user study that tests its effectiveness.
2 Application Design

2.1 Overview

The TextSpotting HoloLens application is designed to aid the visually impaired when they navigate an environment where they must read signs. The application will detect text located in front of the user and place spherical icons where text exists. These icons are anchored to the location of the sign so that when users move around, the icons stay in the same location as the original sign. Additionally, they are designed to be easily detected by visually impaired users and color-coded to reflect how confident the application is in recognizing the text at the icon’s location (green for confident, orange for semi-confident, and red for doubtful). Users can select icons, and when a user selects an icon, the application will read and display the sign that the icon is associated with (see Figure 5). Although the application is targeted towards visually impaired users who still have residual vision, blind users can use the audio-only mode as discussed in Section 2.4.6.

![TextSpotting application screenshots taken on the HoloLens.](a) the user is standing in front of a door sign that says “Rooms 327-330.” (b) the user uses a voice command or the clicker to signal to the HoloLens to detect text. A flashing icon appears where the sign is. (c) the user selects the icon using the clicker or voice command, showing the enhanced visual display of the sign. The application also reads the text aloud.

2.2 Requirements

Because the TextSpotting application uses the Google Vision API to detect text, the application must be used in a Wi-Fi enabled environment. Additionally, users must be able to avoid obstacles either independently or by using another aid such as a cane.

2.3 Application Flow

When the user first launches the application, a hologram cursor will appear and follow the user’s gaze by accounting for his or her head orientation (not eye tracking). The cursor, along with certain gestures and voice commands, allows the user to switch between two states: the interactive and text detection states (see Figure 6a). In the interactive state, the user can adjust the settings of the application, select icons, and tell the application to detect text. Once the user asks the TextSpotting application to detect text, the application enters the text detection state, where it locates any text in front of the user and places an icon at these locations. Technically, the application can be in both states at once (i.e. the user can still select an icon while the HoloLens is detecting text), but this framework simplifies the explanation of the application.
Figure 6: **TextSpotting application flow.** (a) application states and transitions, showing how users can interact with the application. In the interactive state, users can adjust settings and interact with icons. In the text detection state, the application detects text and places icons in the scene. (b) text detection pipeline, showing how the application detects text.

### 2.3.1 Interactive State

In the interactive state, users can switch modes, interact with icons placed in the environment, and trigger the text detection state. The TextSpotting application runs in three modes (manual, automatic, hybrid), and depending on the mode, users can interact with application differently (see Table 2 for a list of voice commands associated with each mode).

In manual mode, users can tell the application when to detect text. Once users decide that they want to find text in the scene, they can either click with the clicker or ask "what’s here?" and the HoloLens will enter the text detection state where it will detect text using optical character recognition (OCR). The application then places icons, which are associated with detected signs in the environment, wherever text is present in the scene. The next section describes the text detection process in more detail. Once icons appear in the scene, users can click on any icon by pointing the cursor to the icon and using the tap gesture, clicker, or voice command ("show me"). Afterwards, green Arial font text (1 cm tall, 0.75 degrees in visual angle) will appear, and the icon will disappear. As users’ head orientations and positions change, the text will follow users so that they can see the text at all times. They can also click outside an icon or say, “hide words,” to dismiss text. The text will disappear, and the icon will reappear in its original location. As long as the application is running, the user can detect text as many times as he or she would like.

In automatic mode, the application will continuously look for text in the scene every three seconds. Although the user cannot manually tell the application to detect text, all other interactions are identical.

Hybrid mode, on the other hand, combines both manual and automatic mode. The application continuously looks for text in the scene every three seconds, but the user can choose to manually tell the application to detect text as well.
Table 2: List of voice commands for the TextSpotting application. The left-hand column shows the available voice commands, the middle column shows which mode(s) the user can use the voice command in, and the right-hand column gives a description of the voice command. In addition to being responsive to voice commands, the application responds to clicker clicks and gestures. In all modes, the clicker and tap gestures can select and deselect icons. In the manual mode, the clicker can also direct the application to detect text.

2.3.2 Text Detection State

When detecting text, the HoloLens will go through the following pipeline seen in Figure 6b:

1. Take a picture of the scene,
2. Send the picture via the Google Vision API to the Google OCR service,
3. Receive the response from the Google Vision API and parse the JavaScript Object Notation (JSON) response to determine what text exists in the scene and where it exists, and
4. Place icons containing text in the 3D world.

The TextSpotting application uses the PhotoCapture API in Unity to take a picture on the HoloLens. After taking a picture, the application constructs a JSON request and sends this request to the Google Vision API. After receiving the JSON response (see Appendix A for a sample JSON response) and parsing the text description and location (represented as a bounding box in image pixel coordinates), the application converts image coordinates to 3D world coordinates. To do this, the application first creates, positions, and rotates a new camera to the same location the picture was originally taken from. It then transforms the text locations of each result (represented as image coordinates) into vectors, averages those vectors to find the center of each result, and uses the new camera to cast a ray in the averaged vector direction towards the spatial map. Once the ray hits the spatial map, the application places an icon at the location of the hit.

2.4 Thresholds, Parameters, and Other Design Decisions

2.4.1 Google Vision API

The application uses the Google Vision API because out of the three OCR approaches that were tested (Google’s Vision API, Microsoft’s Cognitive Services, and the open-source library called Tesseract), the Google Vision API is the most accurate in this use case. Because all images taken by the user will be real-world images, the OCR service must be able to analyze natural-scene images. Natural-scene images present a particularly challenging problem for OCRs because of complex backgrounds and variations of text patterns (e.g. font, size, color) that occur in the real world. When comparing the three OCR techniques, Tesseract had the most trouble detecting text in natural-scene images while Microsoft Cognitive Services could detect and recognize some text. As shown in Figure 7, the Google Vision API was the most accurate
as measured by region and word accuracy (described below). It is important to note, however, that the Google Vision API is not perfect. It, along with Tesseract and Microsoft Cognitive Services, could not detect vertical text and exit signs printed on transparent backgrounds.

To determine which OCR method is most accurate, I measured the region, word, and detected-words accuracy for each OCR. Region accuracy refers to the percentage of regions with text an OCR detects and is calculated by:

\[
\text{Region Accuracy} = \frac{\text{# text regions detected by OCR}}{\text{# text regions in the scene}}
\]

Word accuracy measures the percentage of fully correct words detected by the OCR out of all the words in the scene. It is calculated by:

\[
\text{Word Accuracy} = \frac{\text{# correct words detected by OCR}}{\text{# words in the scene}}
\]

Lastly, detected-words accuracy measures the percentage of fully correct words detected by the OCR out all the words the OCR detects. Detected-words accuracy is calculated by:

\[
\text{Detected-Words Accuracy} = \frac{\text{# correct words detected by OCR}}{\text{# words detected by the OCR}}
\]

To gather data, I took 14 pictures around Dartmouth’s Moore Hall using the HoloLens (see Appendix B for the 14 pictures). For each picture, I determined by hand the regions of text a user might expect an OCR to detect. These 14 images had a total of 41 regions and 172 words. After running the Google Vision OCR, Microsoft Cognitive Service, and Tesseract on the images, I calculated the region, word, and detected-words accuracy for each image and OCR.

Figure 7: Comparison of OCR APIs by accuracy type. The bar plots show how three OCR APIs perform on three different measures of accuracy. Region accuracy refers to the percentage of text regions an OCR can detect. Word accuracy measures the percentage of fully correct words an OCR detects out of all words in the scene. Detected-words accuracy measures the percentage of fully correct words an OCR detects out of all the words it is able to detect.

As shown in Figure 7, the Google Vision OCR is the most accurate OCR out of the three chosen in region and word accuracy. While the Google Vision OCR detected 93% of text regions, Microsoft Cognitive Services detected 59% of text regions, and Tesseract detected 20% of
text regions. Analyzing the word accuracy of OCRs leads to similar results. The Google Vision API recognizes 85% of words correctly while Microsoft Cognitive Services recognizes 40% of words correctly, and Tesseract recognizes 5% of words correctly.

Microsoft Cognitive Services, however, was the most accurate in terms of detected-words accuracy, correctly identifying 78% of the words it detects. The Google Vision OCR followed with a detected-words accuracy of 71%, and Tesseract had a detected-words accuracy of 33%. These results suggest that Microsoft Cognitive Services detects less text but is more correct with its detected text. Google Vision API, on the other hand, had a lower detected-words accuracy than word accuracy, suggesting that the Google Vision API tends to over-detect text. It lets the user know that text exists without being as certain with what the text says.

Figure 8 below shows a specific example of the accuracy of these three OCRs. The Google Vision OCR is able to detect the text highlighted by the green boxes. It is able to recognize the room number “ROOMS 323-326” and text on the poster. Using the same image, Microsoft Cognitive Services could only detect “ROOMS 323-326” while Tesseract could not detect any text in the scene. The Google Vision OCR did have issues, however. The text recognition for the poster is worse because of the smaller font. While the true text says “Department of Psychology Colloquium,” the Google Vision OCR responds with “Department of Psychology Colloouum,” showing how the Google Vision API can have a lower detected-words accuracy than Microsoft Cognitive Services. Despite this error, the Google OCR is more accurate than Microsoft Cognitive Services and Tesseract in region and word accuracy.

Only three OCR services were considered because many algorithms [9, 12, 45] and methods for text detection and text recognition, although developed, are not available to the public. It is also uncertain how well they would work with natural scene images. While Google has not explained how their OCR service works, Panigrahi [36] speculates that it uses Tesseract in some way. Tesseract uses adaptive thresholding, connected component analysis, and adaptive classifiers to recognize text [46].
2.4.2 Combining Text into One Icon

When the Google Vision API sends a response back to the application, the response separates every word into its own entry. If each entry corresponded to a different icon, this would create too many icons in the scene. Instead, certain pieces of text should be combined into one icon (e.g. “Room” and “323” should be in the same icon and not two separate icons). To tackle this issue, I determined when two entries should be in one icon by calculating the distance in pixels between the two pieces of text in the entries. If two pieces of text are 30 pixels apart or closer, they should be in the same icon. Otherwise, they should be in separate icons.

I empirically arrived at this threshold by taking pictures, running the Google Vision API on the pictures, and determining by hand when the text should be combined. More specifically, I took 14 pictures around Dartmouth’s Moore Hall using the HoloLens and ran the Google Vision API on them (see Appendix B for the 14 pictures). After looking at each JSON response, I hardcoded the result of whether two entries should or should not be combined into one icon. I followed a couple of guidelines when determining whether text should be combined:

1. Two pieces of text should generally be combined if they are a part of the same sign,
2. However, even if two pieces of text are part of the same sign, if they refer to two different sections of the sign, then they should be separate (e.g. the title of a poster is different from the description of the poster). Sections of a poster are identified by font sizes.

After determining whether text should be combined, I calculated the distance between two pieces of text using the following formula and plotted the results for when entries should be together and when they should not be together:

\[ \text{Distance} = \min(\text{minimum width distance}, \text{minimum height distance}) \]

Figure 9 displays the results. The 97.5th percentile for being together is 31 pixels, and the threshold set on the application is 30 pixels.

Figure 9: Analysis to find the pixel threshold for combining icons. The box plots show the distribution of distances in pixels for icons that should be kept separate (left) and icons that should be combined together into one icon (right). 25% and 75% quartile limits (boxes), median lines (bolded lines), means (X’s), and outliers (dots) are shown for both groups.

The application currently uses pixel distance to set the threshold for whether or not two results should be combined, but distance in pixels differs with viewing distance. Thus, a more
accurate method would be to set this threshold based on visual degrees rather than pixel distance. Additionally, the application can obtain more accurate results by taking into account the relative size of individual letters because larger letters will be farther apart from each other than smaller letters.

2.4.3 Coloring Icons Indicate Confidence

The accuracy of the OCR depends on the resolution with which the text is sampled. To communicate the accuracy of the OCR, the TextSpotting application shows a green icon if the OCR is confident with its results, an orange icon if it is not as confident, and a red icon if it is doubtful (see Figure 10). The icon is green if the text in an icon is larger than 21 pixels; the icon is orange if the text is between 11 and 21 pixels, and the icon is red if the icon is less than 11 pixels. I derived the confidence level using a calibration procedure based on the height of a text box returned by the Google Vision OCR.

Figure 10: TextSpotting screenshot with confidence levels. The green icon shows that the application is confident that text it detected at that location is accurate. The orange icons show that the application is semi-confident that the text it detected at that location is accurate.

For this calibration procedure, I took 14 pictures at Dartmouth’s Moore Hall using the HoloLens and ran the Google Vision API on the pictures (see Appendix C for the 14 images). I measured how accurate each result was by calculating the edit distance between the expected and actual result (e.g. if the word is supposed to be “apple” and it got “apole,” then the accuracy is 80%). Then, I created bins for a range of pixel heights determined from the bounding box and calculated the average accuracy for those bins. The range of pixel heights corresponding to each confidence level was hand-selected to maximize the difference between the confidence levels. However, because these images are natural-scene images, letters in one word are not all the same height, and the bounding box can be any quadrilateral. Thus, the pixel height for a given word can be calculated using the maximum, minimum, or average height of the bounding box coordinates. I used all three methods and found that using maximum height resulted in the largest difference between confidence levels.

As shown in Figure 11, for any method of calculating height, accuracy increases as you move from the 0-10-pixel bin to the 21-or-more-pixel bin (this was not the case for other bin sizes). The maximum height method of calculating size, however, gave the clearest separation in accuracy. When the text is 0-10 pixels in height, the application is certain that the recognized text is not accurate. When the text is 11-20 pixels in height, the application is 72% certain that the recognized text is accurate, and when the text is 21 or more pixels in height, the application is 83% certain. Because a greater standard deviation in accuracy levels gives more meaning to
the distinction between green and orange icons, I chose the maximum height method as the method to calculate size of an icon.

![Image](image.jpg)

**Figure 11:** Analysis to find the confidence of a Google Vision result given the pixel height of the result. The bar plots use three different methods of calculating height to separate Google Vision API results into three groups: 0-10, 11-20, and 21+ pixels. For each group and method of calculating height, the corresponding accuracy of the result is calculated. The ideal method is the method that creates the largest separation between the three groups.

### 2.4.4 Icon Visibility

In addition to changing the icon color to indicate confidence, the TextSpotting application displays the icons such that people who are visually impaired can easily find the icons. First, the TextSpotting application adjusts the size of the icon so that it matches the text in the real world. This allows the user to identify what the text might be by the size of the icon. To calculate the size of the text, I use the bounding boxes Google Vision API provides with each text it detects (see Appendix A for a sample JSON response). Using these bounding boxes, the HoloLens projects three rays to the top left, top right, and bottom left corner of the box. Using those rays, the application finds the width and height of the object, where:

\[
width = top\ right\ x\ coordinate - top\ left\ x\ coordinate
\]

\[
height = top\ left\ y\ coordinate - bottom\ left\ y\ coordinate
\]

The diameter of the icon is equal to the maximum of the pixel width and the height calculated above. Originally, the application placed rectangular icons at locations where text exists, but the spatial map provided by the HoloLens was not detailed enough to create precise rectangular icons in the world. Thus, the rectangular icons were always shifted slightly off center from where the text truly existed in the real world. Consequently, I decided to have the application use spherical icons to compensate for this slight error.

Second, icons flash at a frequency of 3 flashes per second, which is below the frequency that is likely to trigger photosensitive epilepsy (5-30 flashes per second) [44]. Flashing icons help visually impaired users notice the icons in the scene because of the continually changing contrasts. When users set the cursor on an icon, that icon stops flashing and turns magenta so that users know which icon they can select at that moment (see Figure 12).
Figure 12: TextSpotting screenshot with user gazing at an icon. When the user gazes at an icon, it turns magenta and stops flashing so that low vision users can clearly identify which icon they are currently looking at. Making a selection command by using the clicker, making a tap gesture, or using a voice command will select the magenta icon.

2.4.5 Icon Tracking

During the automatic and hybrid modes, the application continuously detects text and places icons in the scene. To avoid placing duplicate icons, I empirically found a threshold distance between two icons such that if two icons are closer than that threshold distance, then the application confidently labels the icons as duplicates. I calculated the threshold distance by first taking four different pictures of the same scene. Each picture had a different viewing angle and distance to account for the different orientations the Google Vision API might encounter (see Appendix D for the four images). After running the Google Vision API on each image, I took two icons and determined whether or not the icons were duplicates of each other. After analyzing 31 pairs of icons, I found that two icons that are 0.1313 Unity units apart are duplicates 95% of the time (see Figure 13) and thus set the threshold distance to 0.135 Unity units (i.e. if the application finds two icons that are less than 0.135 apart, the application deems the icon to be a duplicate).

Initially, whenever the application is about to place an icon in the scene, it simply calculates the distance between that icon and every other icon in the scene. However, the time complexity of this process is quadratic, and even with just 15 icons in the scene, latency issues already began to rise. Thus, I implemented a technique called locality hashing. First, the application creates a dictionary where icons with similar text are all hashed to the same key. Then, instead of comparing every potential icon to every icon already in the scene, the application only compares a potential icon to icons already in the scene with similar text. To figure out if two icons have similar text, the application calculates the Levenshtein distance between the text and a constant text. The Levenshtein distance calculates how many edits must be made for one string to match another. The constant text is used to create hash keys for the dictionary and bucket similar pieces of text together. This technique greatly reduced the time needed to calculate duplicate icons, thus, increasing the speed of the application.
Figure 13: **Analysis to determine if an icon is a duplicate of another.** The box plots show the distribution of distances in Unity units for icons that are duplicates (left) and icons are not duplicates (right). 25% and 75% quartile limits (boxes), median lines (bolded lines), means (X’s), and outliers (dots) are shown for both groups.

### 2.4.6 Audio Only Version

The audio only version of the application, like before, detects and recognizes text around the world in manual, automatic, or hybrid mode. Instead of placing icons in the scene, however, the application reads the results from the Google Vision API directly to the user. Although this version has not been tested in a user study, this version aims to aid either those who are completely blind or those who cannot recognize icons even with their residual vision.

### 2.5 Issues

#### 2.5.1 JSON Parsing and NuGet

To construct the JSON request and parse the JSON response from the Google Vision API, I originally used the Newtonsoft JSON library, which I have since changed to the SimpleJSON library. Using these libraries presented a challenge because traditionally, packages are imported into Visual Studio using the NuGet package manager. However, because Unity builds the Visual Studio solution, packages must be imported via Unity, not Visual Studio. This was solved by creating a “Plugins” folder in Unity and directly importing the Newtonsoft JSON libraries into the folder. However, because Unity only supports version 3.5 of the ASP.NET framework (current version is 4.5), a previous version of Newtonsoft JSON must be used in development whereas the most recent version is used after deploying to the HoloLens.

#### 2.5.2 Latency

Originally, the application took 19 seconds end-to-end for it to complete the text detection process (take a picture, send the picture to the Google Vision API, receive the JSON response, and place the icons in the scene). The two biggest contributors to this delay are (1) taking the picture, which constituted 7.1 seconds or 37% of the delay, and (2) waiting for the Google Vision API to respond, which constituted 10.2 seconds or 53% of the delay.

I eventually reduced this 19 second end-to-end delay down to a 3.7 second delay where taking the picture constituted 1.2 seconds or 32% of the delay, and waiting for the Google Vision API took 2.2 seconds or 59% of the delay. To reduce the delay, I first sent smaller
resolution (1280x720 instead of 2048x1152) pictures to the Google Vision API so that it could respond faster. Second, I reduced the amount of time it took to take a picture by changing how the picture is stored in memory. Originally, I saved the picture to the hard drive of the HoloLens. After I changed the application to store pictures in RAM and construct the JSON request from these pictures, the delay to take and store a picture reduced from 7.1 seconds to 1.2 seconds.

I used the latency of the application to determine the frequency with which the automatic and hybrid modes continuously call the Google Vision API. Although the latency is 3.7 seconds, the automatic and hybrid modes call the Google Vision API every 3 seconds. As long as the application knows the previous call’s results, the application can make the next API call because it can check where duplicate icons might exist. Since the Google Vision API responds in 2.2 seconds, the application still runs smoothly even when it continuously calls the API every 3 seconds.

In its current state, the latency of the application is due mostly in part to waiting on the Google Vision API to respond with the results as 59% of the delay is due to this bottleneck. In future versions of the application, this delay can be reduced by using a locally run OCR (e.g. Tesseract) or by improving the Wi-Fi connectivity of the HoloLens.

2.5.3 Icon Replacement

Originally, many icons were generally positioned in the correct location in the scene, but they were shifted towards the center of the user’s gaze. This issue arose because of the different field of views the HoloLens supports for videos and images. While the HoloLens takes video at a fixed resolution of 1280x720, the HoloLens can take pictures at multiple resolutions. By default, the application takes pictures at a resolution of 2048x1152. With these two resolutions, pictures taken by the HoloLens have a wider field of view compared to videos taken by the HoloLens. However, raycasts from the HoloLens camera must be within the video field of view (1280x720). Thus, it was not feasible to reach some areas of the 2048x1152 image even after scaling down the image. To solve this issue, I changed the application to take pictures in the 1280x720 resolution. Because this resolution is smaller, users might see text in front of them that is out of range from the video resolution. In this case, users must recognize the application’s limited field of view and reorient themselves so that the desired text is directly in front of them (see Figure 12).

2.5.4 Spatial Mapping Latency

Because the HoloLens takes time to update the spatial map, rays cast from the camera that are used to place icons in the scene may never hit the spatial map. In these cases, the application does not place icons in the scene. To solve this issue, the application places a plane at a fixed distance of 2 meters in front of the user at all times, where 2 meters represents the typical viewing distance for posters and signs [37]. If the ray does not hit the spatial mapping, then the application will place the icon where the ray hits the plane instead.

3 User Study

In addition to the TextSpotting application, I designed and ran a user study to test how well the TextSpotting application would work. Because success with an assistive device is determined by how well the device performs and how satisfied the user is with using it, the study can effectively evaluate the strengths and weaknesses of the application (Jutai, Strong, & Russell-minda, 2009). The goals for this study are as follows:

- To test the reliability and functionality of the TextSpotting application in a real-world context,
- To assess how helpful the TextSpotting application is in navigation tasks in terms of speed, ease, comfort, and confidence.
3.1 General Methods

To achieve these goals, I conducted a user study with human subjects, who wore modified goggles to simulate low vision (see Figure 14). Their task was to locate a room in an unknown hallway, and they completed the task twice in two different hallways: once with only simulated impaired vision (“low vision only” condition) and once with impaired vision and the TextSpotting application (“low vision + app” condition). Participants were randomly assigned the condition they would first complete the task in: in the low vision only condition or the low vision + app condition. Although I developed automatic and hybrid modes for the application, participants only used the application in manual mode.

3.1.1 Participants

24 participants took part in the user study. Participants were recruited from the Dartmouth undergraduate student population. All participants gave written informed consent, and the Dartmouth College Institutional Review Board approved the study procedures. Volunteers were excluded if they ever visited a professor’s office on the second or third floor of Dartmouth’s Moore Hall, which is where the study was held. 12 participants completed the navigation task in the low vision only condition before completing the task in the low vision + app condition (5 male and 7 female, mean age 19.2 years, mean normal visual acuity -0.08 LogMAR, mean impaired visual acuity 1.27 LogMAR). 12 participants completed the navigation task in the low vision + app condition before completing the task in the low vision only condition (4 male and 8 female, mean age 19.8 years, mean normal visual acuity -0.07 LogMAR, mean impaired visual acuity 1.29 LogMAR).

3.1.2 Procedure

In the study, participants wore a pair of modified SPEEDO Vanquiser 2.0 goggles to simulate low vision. These goggles had two Bangerter occlusion foils attached: the inside of the goggles had a layer of LT 0.1 occlusion foil attached and the outside of the goggles had a layer of 0.1 occlusion foil attached. Before participants started any task, I trained them on how to wear the HoloLens, and before they began a task in the low vision + app condition, I trained them on how to use the TextSpotting application in manual mode. During training, participants used the application in a separate room away from the navigation task location and practiced on printed signs that simulated signage in the real world. While they were in the separate room, they could use the application for as long as they desired. After training, participants moved to a fixed start location at the end of a hallway to begin their assigned navigation task, where they searched for a professor’s room in the hallway.

Regardless of the condition, participants were asked to find Professor Manning’s room in the first task (see Figure 15a). In the second task, they were moved to a different hallway with an identical layout (see Figure 15b) and asked to find Professor Smith’s office, located on the opposite side of the hall compared to Professor Manning’s office.
office is 17.6 meters from the starting position while Professor Manning’s office is 16.7 meters from the starting position. After completing the first navigation task, participants completed a questionnaire that asked how easy, comfortable, and confident they thought the task would be in a typical situation on a five-point Likert scale. After completing the second navigation task, participants completed the same questionnaire and an additional questionnaire in which they compared the two conditions.

Figure 15: **Floorplans for the tasks in the user study.** (a) floorplan for task 1. Participants began at the starting location and were asked to find professor Manning’s office. (b) floorplan for task 2. Participants began at the starting location and were asked to find professor Smith’s office.
4 Results

All participants completed the tasks successfully. Overall, the results show that participants completed the navigation task faster in the low vision only condition. However, they reported higher levels of ease, comfort, and confidence in the low vision + app condition. These results hold separated by tasks, and a two-way ANOVA shows significant main effects of condition on task time, reported ease, reported comfort, and reported confidence (see Table 3 for a summary of the results).

Data across both tasks confirm the overall results (see Figure 16 for boxplots of the results). Participants on average completed the navigation task in 60.72 seconds when in the low vision only condition and 174.62 seconds when in the low vision + app condition. Despite being slower in the low vision + app condition, participants reported higher ratings on the ease, comfort, and confidence of the task in the same condition. While participants in the low vision + app condition reported average scores of 3.17 on ease, 3.38 on comfort, and 3.96 on confidence, participants reported average scores of 1.54 on ease, 2.00 on comfort, and 2.17 on confidence when they were in the low vision only condition. A one-way ANOVA revealed significant effects ($p < 0.001$) of condition on task time, reported ease, reported comfort, and reported confidence. However, because participants completed the same task twice, these results potentially have strong ordering effects.

Between subject differences for the first and second tasks are shown in Figure 17, and the data reveal similar results. Looking at the data for only the first task, I find that participants who completed this task in the low vision only condition (58.59 seconds) were on average faster than participants who completed the first task in the low vision + app condition (187.86 seconds). However, participants who completed the first task in the low vision + app condition reported higher ratings of ease, comfort, and confidence (average scores of 3.42, 3.83, and 4.08, respectively). Participants who completed the first task in the low vision only condition reported ratings of 1.33 for ease, 1.83 for comfort, and 2.17 for confidence. Thus, ordering effects to not seem to play a large role in the effect differences between the low vision only condition and the low vision + app condition.

Additionally, data from how participants performed in their second task reflect similar results (see Figure 17 for boxplots of the results). Participants who completed the second task in the low vision only condition (62.85 seconds) were on average faster than participants who completed the task in the low vision + app condition (161.37 seconds). Like previous data, however, participants who completed the second task in the low vision + app condition reported higher levels of ease, comfort and confidence. Participants in the low vision + app condition reported average scores of 2.92 for ease, 2.92 comfort, and 3.83 for confidence while participants in the impaired only condition reported average scores of 1.75, 2.17, and 2.17, respectively. Like before, the reported differences are statistically significant. A two-way ANOVA shows significant main effects ($p < 0.001$) of condition on task time, reported ease, reported comfort, and reported confidence. After completing both tasks, participants also directly compared the two conditions on levels of ease, comfort, and confidence. While 96% of participants preferred the low vision + app condition in terms of ease, 83% thought the low vision + app condition was more comfortable, and 92% felt more confident in the same condition.
Table 3: **Summary of results.** The “Both Tasks” pane shows results across both tasks. The “Task 1” pane only shows results for the first task participants completed where they found Professor Manning’s office. The “Task 2” pane only shows results for the second task participants completed where they found Professor Smith’s office. A one-way ANOVA test shows that the differences in time, ease, comfort, and confidence are significant across both tests. A two-way ANOVA test shows significant main effects of condition on task time, reported ease, reported comfort, and reported confidence. Significant code: “***” is $p < 0.001$.

![Table 3: Summary of results](image)

Figure 16: **Overall performance and reported scores.** Results of both conditions are compared using data from both tasks. (a) distribution of time taken to complete the navigation task. (b) distribution of reported ease for the navigation task in a typical environment. (c) distribution of reported comfort for the navigation task in a typical environment. (d) distribution of reported confidence for the navigation task in a typical environment. For all boxplots, 25% and 75% quartile limits (boxes), median lines (bolded lines), means (X’s), and outliers (dots) are shown.
Figure 17: **Performance and reported scores separated by tasks.** In each panel, the task duration and scores are shown separately for task 1 and task 2. In task 1, users are instructed to find professor Manning’s office, and in task 2, users are instructed to find professor Smith’s office. (a) distribution of time taken to complete the navigation task. (b) distribution of reported ease for the navigation task in a typical environment. (c) distribution of reported comfort for the navigation task in a typical environment. (d) distribution of reported confidence for the navigation task in a typical environment. For all boxplots, 25% and 75% quartile limits (boxes), median lines (bolded lines), means (X’s), and outliers (dots) are shown.

5 Discussion and Next Steps

5.1 Summary

These results, at the most general level, show promise that systems designed for consumer-grade augmented reality may help people who are visually impaired. Although participants completed the task more slowly while in the low vision + app condition, it is important to note that they had minimal experience using the application even with the training, and they were asked to complete the task at their own pace. Thus, many participants in the low vision + app condition chose slower, more comfortable strategies such as using the "read all here" voice command to read text as opposed to clicking the icons with the clicker.

Despite being slower at completing the navigation task, participants still found value in using the application. They reported higher scores on ease, comfort, and confidence, and when they directly compared the two conditions, the majority of them preferred using the application. Participants may have reported higher scores for a number of possible factors that are worth exploring individually in future studies. The application allows people to interact with the world in intuitive ways that mirror how typically sighted individuals interact with the world around them. Additionally, the application allows them to stand a fair distance away from signage, thus, reducing the awkwardness of walking all the way up to a sign. However, a few potential reasons may have also reduced these reported scores. Many participants not only
remarked on the bulkiness and conspicuousness of the HoloLens but also commented on the noticeable latency of the application and the need to click icons. Nevertheless, participants still reported higher levels of ease, comfort, and confidence, suggesting that the application has benefits that outweigh the costs. Slimmer hardware designs and the automatic or hybrid mode could make the application faster and more comfortable to use. These preliminary results, thus, show that augmented reality and more specifically the HoloLens is a promising platform to help the visually impaired in daily tasks.

5.2 Limitations

The findings reported above, however, must be understood in the context of the user study. First, participants all wore goggles to simulate low vision and, thus, never experienced low vision outside of the experiment. People who are visually impaired may respond differently to the application. Second, all participants wore the same pair of goggles, resulting in a mean visually impaired acuity of 1.28 LogMAR. Because the range of impaired visual acuity was from 1.1 to 1.54 LogMAR, the results of this user study are only applicable for people with visual acuities within this range. People with better visual acuities might have no need for the application when performing a navigation task while people with worse visual acuities might not find the application as helpful. However, the application runs in other modes (e.g. audio-only mode, automatic mode, hybrid mode) as well that may help those with worse visual acuities. These modes, however, were not tested in the user study.

Third, the results of the study are not applicable to many navigation tasks because the study tested a specific navigation task (finding a professor’s room in an unknown hallway) in a controlled environment. This task is very specific, and participants indicated that they do not perform this task often on a weekly basis (2.84 times per week on average). Additionally, this study was done in a very controlled environment, where the hallway was entirely empty, and there were no outside distractions. Typical environments, however, can be noisy, crowded, and filled with obstacles, all of which can reduce the effectiveness of the application. Fifth, placebo and ordering effects may have significantly affected the results. Because participants were not blind to the two conditions, the placebo effect could have caused participants to report higher ratings in the low vision + app condition. Each participant also completed the same task twice. Completing the first task may have familiarized people with the floorplan and may have caused some to approach the second task with a different strategy. The size of these effects are unknown and could have played a significant role in the results. Although I hypothesize participants to be faster on the second task while holding the condition constant, the data show the opposite for those in the low vision only condition. Participants who completed the second task in the low vision only condition completed the task slower than participants who completed the first task in the low vision only condition. This suggests that ordering effect may not have been a significant factor in the results, but it could also mean that those who completed the low vision only condition were accustomed to the application and found a harder time adjusting to using only their residual vision.

In addition to the study’s limitations, the application itself has limitations. The HoloLens is currently unavailable to the public and currently costs $3,000. Although this is considerably cheaper than some assistive devices, the cost of the HoloLens could still be a barrier to some. The HoloLens itself has a few limitations as well. In addition to the limited field of view, the displays can be difficult to see outdoors because of the increased brightness outdoors. Thus, the HoloLens is only effective indoors, making the application limited to indoor environments. Furthermore, the social consequences of wearing the HoloLens is currently unknown. Because the HoloLens is bulky and conspicuous, users might be socially deterred from wearing the HoloLens. However, the HoloLens contains many more applications than just the TextSpotting application. Thus, a user wearing the HoloLens does not self-identify as visually impaired, and this type of inconspicuousness may make the application more socially acceptable. Lastly, the HoloLens stores the spatial map of its surrounding areas, but storing large spaces in memory slows down the device. Thus, if users use the application in large spaces, the application could lag and create an uncomfortable experience for users. Aside from the inherent limitations of the platform, the TextSpotting application also has its own limitations. The application
currently requires Wi-Fi connectivity because it uses the Google Vision API to detect and recognize text. In manual mode, the user must also be able to detect where potential text exists so that they can tell the TextSpotting application to detect text. Some low vision users may not have this ability, and some signs might be difficult to detect. The automatic and hybrid modes, however, address this issue by continually telling the application to detect text so the user does not have to explicitly tell the application to detect text. However, these modes currently run slowly and give the user less control.

5.3 Next Steps

Although the application is mostly finished, there is much more room for improvement. First and as already noted, reducing latency is important. Currently, the delay in the application is mostly due to waiting for the Google Vision API to return the result. Thus, avoiding the sending of information across Wi-Fi could help the application run much more quickly. To do this, I could use another OCR that doesn’t require the cloud to eliminate the latency associated with sending data over Wi-Fi. Currently, I have implemented a proof-of-concept version with Vuforia, an augmented reality library that integrates with the Unity and the HoloLens. However, development is currently at early stages.

Second, there could be a better way to show users which icon(s) are important so that users don’t have to select every icon in the scene to find what they are looking for. The application can recognize potentially important pieces of text by analyzing the text and looking for keywords. For instance, icons that contain keywords such as “fire” or “exit” might be particularly important, and different colored icons can show up in these cases. These keywords can be either directly hardcoded into the application or discovered using machine learning.

Third, the Google Vision API currently does not detect symbols such as arrows. These universal symbols should be integrated into the application because they are inherently directional and used for navigation. Other APIs can be used in conjunction with the Google Vision API to recognize symbols.

Fourth, the user should be able to customize the text size and color when the application displays it on the screen. Currently, these settings cannot be changed by the user. This can simply be done by including a visual or auditory menu when the user first opens the application.

Fifth, when the Google Vision API fails to perfectly recognize text, it usually returns a misspelling of the text. This issue can be reconciled in two ways: a spelling filter can be applied to the results of the Google Vision API so that the results are more accurate or multiple OCRs can be used. The application can compare the results of the OCRs and either combine or use the best result.

6 Conclusion

Using the Microsoft HoloLens, I created an application that helps people who are visually impaired navigate text-heavy environments. The TextSpotting application detects and recognizes text around a user by placing flashing icons in the world. Users can select and deselect these icons, and the application will display and read the text aloud. In addition to developing the application, I ran a user study to test its effectiveness. Users were able to complete a navigation task more quickly while only visually impaired, but the application made the task easier and made users feel more comfortable and confident during the task. Although visual impairment remains an extremely complex issue, consumer-grade augmented reality opens new doors to a promising, new platform of assistive aids.

Acknowledgements

This work would not have been possible without many people who have supported me throughout this last year. First and foremost, I would like to thank my advisors Prof. Emily Cooper and Prof. Wojciech Jarosz for motivating and pushing me throughout this entire process. They have
not only taught me a great deal about scientific research but also discipline and patience. I would also like to thank Prof. Xing-Dong Yang and my fellow labmates (Max Kinateder, Justin Gualtieri, and Klara Barbarossa) for their constant feedback on the application, the user study, and the thesis. I am especially indebted to Max, who has continually supported and given up his time and effort for me this year.

I am also grateful to all of those who I have worked with on this project. The Dartmouth psychology professors willingly allowed participants to roam the hallways during the study, and the custodians of Dartmouth’s Moore Hall greatly accommodated as I ran my user studies. Last but certainly not least, I would like to thank my friends, family, and God for their everlasting love and guidance in all that I pursue. Their loving support has been and will continue to be extremely crucial to me, and the pursuit of this thesis has only confirmed that.
Appendices

A Sample JSON response from the Google Vision API

Given an image, the Google Vision API responds with a JSON response that lists the text found and the location of each text found in the picture.

Figure 18: Sample JSON response from the Google Vision API. (a) a user stands and takes an image of the door sign using the HoloLens. (b) the image is sent to the Google Vision API and the API detects text in the image as represented by the green boxes. Listing 1 below shows the resulting JSON response.
Listing 1: Sample JSON Response from the Google Vision API

```json
{
"textAnnotations": [
{
  "locale": "en",
  "description": "ROOMS\n327-330\n",
  "boundingPoly": {
    "vertices": [
      { "x": 955,
        "y": 577
      },
      { "x": 1344,
        "y": 577
      },
      { "x": 1344,
        "y": 754
      },
      { "x": 955,
        "y": 754
      }
    ]
  }
},
{
  "description": "ROOMS",
  "boundingPoly": {
    "vertices": [
      { "x": 955,
        "y": 599
      },
      { "x": 1293,
        "y": 577
      },
      { "x": 1297,
        "y": 634
      },
      { "x": 959,
        "y": 656
      }
    ]
  }
}
...
]}
```
B Comparing accuracy of OCR APIs and determining the threshold distance to combine icons

These images represent 14 pictures taken by the Microsoft HoloLens to analyze the accuracy of different OCR APIs (Google Vision, Microsoft Cognitive Services, and Tesseract) (see Section 2.4.1) and determine the pixel threshold for combining multiple text results into one icon (see Section 2.4.2).
Figure 19
C Determining the confidence scores of each icon

These images represent 14 pictures taken by the Microsoft HoloLens to convey how confident the application is with its results (see Section 2.4.3).
D Tracking and preventing duplicate icons

These images represent four pictures taken by the Microsoft HoloLens to determine the distance threshold for tracking and preventing duplicate icons (see Section 2.4.5).

Figure 21
References


Microsoft. Spatial mapping.


National Eye Institute. Blindness.


