Interactive Learning in Mixed Reality (May 2018)

Alekhya Lingutla
Computer Science and Engineering
Chaitanya Bharathi Institute of Technology

Mohammed Aijaaz
Computer Science and Engineering
Chaitanya Bharathi Institute of Technology

Isha Padhy, Assistant Professor
Computer Science and Engineering
Chaitanya Bharathi Institute of Technology

***Abstract*—Mixed reality is the result of blending the physical world with the digital world. It is the next evolution in human, computer, and environmental interaction and unlocks possibilities that before now were restricted to our imaginations. It is made possible by advancements in computer vision, graphical processing power, display technology, and input systems. Microsoft HoloLens is the first self-contained, holographic computer, enabling to engage with the digital content and interact with holograms in the world. Specialized components—like multiple sensors, advanced optics, and a custom holographic processing unit—enable to go beyond the screen. We present a method of utilizing the HoloLens for advanced learning in Mixed Reality. One methodology of achieving this goal is, when a user is reading a book while wearing the HoloLens, if the user comes across a word which he/she has no idea about, then the user taps on the word using a tapping gesture, this will be recognized by the HoloLens. It then uses Optical Character Recognition (OCR) tools to recognize the word. The next step is to use the word and look up using Google search APIs to get relevant results. The most relevant result will be chosen and is prepared to be rendered as a hologram. The user can interact with the hologram to understand more about it, to have a look from all angles etc.**

***Index Terms*—Optical Character Recognition, Mixed Reality, HoloLens, User interfaces, Annotations**

I. Introduction

The purpose of this project is to understand and make use of the HoloLens' sensual and natural interface commands, interface with them and connect them to make learning interactive. The entire system is categorized under Mixed Reality [1], implying that the application is superimposed onto the real world. The underlying idea behind the project is to interface with a standalone wearable system, and to develop an application for the HoloLens, using Unity and Visual Studio. The main contribution would be to detail how an application for the HoloLens can be built with the use of available resources.

A. Problem Definition

The problem definition is as follows, Interactive Learning in Mixed Reality is to make learning interactive by leveraging the Microsoft HoloLens[2]. Interactivity is achieved by letting the user of the application understand about what he/she is seeing in real time. The audio/video descriptions and 3D models of a particular phrase are rendered on the HoloLens, with which the user can understand about the phrase. As the results are rendered right in front of the user superimposed in real 3D space [3], the user need not shift his/her attention towards another source which would waste time.

B. Existing System

The traditional way of learning is very slow. The environment limits the students to learn faster and more efficiently. Currently, if a student wants to understand a particular word, the word can be name of a process, or any entity, the student would use any of the following resources to look up for the word,

- Encyclopedia
- Dictionary
- Internet
- Subject books
- Journals/Papers

Not all resources may be available to the student. If the student has access to a good library with encyclopedias, dictionaries, books etc., then the student may use them to look up for the required word/phrase and try to understand. The problem with this approach is that, the time taken to reach the library, find the right book, right section in the book, learn it repeatedly and to understand is very large. During the process the student might also loose interest. On the other hand, if the user has access to the Internet, then the user can use a search engine to make the task faster. Though this will significantly reduce the time taken to lookup and understand, there are some shortcomings to this approach. The main problem here is that, if the student is not near a computer then it will add the overhead of physically going to a computer, turning it on and then looking up. If there were a handheld device with internet connectivity, then the time would even be reduced but there is still distraction invovled between the student and the content. All these methods discussed here reduce the distraction more than the previous methods, but none of these methods completely remove distractions [6].

C. Proposed System

The proposed system leverages the head mounted HoloLens with Mixed Reality technology. This system makes the line of distraction between the user and the content finer than the existing system. Here, the user, wearing a head mounted HoloLens will see a word in his/her surroundings, then essentially Airtaps the word that the user wants know more about. As soon as the word is tapped, information on the word

will be shown right next to it in real 3D space [2]. The information can be in textual, audio, video, 3D formats. Most of the phrases have an image and a text to help the user understand. While Airtapping – gesture recognition is one way of interacting with the application, another way is to use voice to give input to the application. If the word/phrase that the user wants to know more about is not in sight then the user can use the second method of input that is through voice the user can let the application know about the word he/she is looking for [3].

II. SYSTEM DESIGN

The project focuses on the implementation of the algorithm that can take inputs from the user and outputs the results. The inputs to the algorithm are from two events. They are as follows:

- AirTap Gestures
- Voice Input

Interactive Learning:

Event: On Air tap

Take screenshot s
Invoke Google Vision API
Get the response from the Google Vision API
Parse the response
Draw annotation frames about the word
Invoke Google Custom Search API
Parse the response
Display the result

End

When the user makes an AirTap gesture [1], a screenshot of the current frame is taken. This frame is sent to the Google Vision API for Document Text Detection. The response is then parsed to get individual words from the frame. For each word extracted from the response, a frame is to be rendered over the word as an annotation 21 of the corresponding word. The user is then presented with these annotations which can be tapped on. When the user taps on an annotation that he/she wants to know about, a request to Google Custom Search API [4] is made with the word as the query parameter. The API then responds with a JSON containing the search results obtained from the preconfigured search engine. The response is then parsed to extract a textual description of the query and a thumbnail image. This is displayed on a panel where the user can easily read and understand. The other input method – voice input is activated to take the input from the user's speech. Once the word is extracted from the user's speech the next steps of requesting the search API to get the results is the same as for the gesture input.

III. IMPLEMENTATION

The implementation of the proposed system is briefly discussed below, along with the environmental details.

A. Unity 3D

Implementing the Project on a Game Engine is the first Module that is to be completed. In Unity 5.5, the entire Scene, including all its menu and User Interface is designed. GameObjects[2] constitute the fundamental units of the Game Engine. Primarily, Unity works with Scenes. A Scene is, in the most basic terms, a representation of the state of the system. In a Scene, multiple GameObjects may be placed, and their components changed. This is the fundamental idea behind designing a app in HoloLens. Unity provides a user with a Camera and a Directional Light, both of which are crucial to ensuring that all elements in a Project are captured. Any user can only view objects from the standpoint of the Camera.

In this project, the frame for annotations is designed in Blender and is imported in the Assets folder in Unity. Now, Menu needs to be designed in order to create a User Interface for the user. Menu can be created using canvas which is present under GameObject > UI > Canvas. The buttons for taking a screenshot, voice input and closing the panel are added in the menu under the hierarchy in Unity [3]. A cursor is imported into the Unity, so that it can be used as a mouse cursor which is visible in the HoloLens device. A small option of Mapping is put in the main menu in order to scan the room and place the furniture items according to the user. All the scripts for selection of options in menu are programmed in C#. An overview of the scene can be played in Unity by clicking on the play option. The menu and the items can be changed anytime, according to the implementation of the user. The changes made in Unity will reflect to the app after it is deployed

B. Visual Studio

Microsoft Visual Studio is an IDE that houses the code required to run an Application. Visual Studio debugs and compiles the code that is responsible for the App to run. It must be understood that Unity only helps achieve the design of the app i.e the outer shell. The code required to ensure that the menu as well as the furniture items in the app perform their functions [5], and other necessary actions must be placed in Visual Studio, and then transferred to Unity. After the scripts are written in Visual Studio, they need to be imported into the hierarchy and to the inspector panel of Unity according to the functionality of the script.

- Change the platform to Windows Store.
- Change the SDK from Windows 8.1 to Universal 10.
- Change the UWP Build Type to D3D.
- Check the Unity C# Projects.
- Enable the Spatial Perception and Microphone in the player settings
- Select on the Player Settings button and from the right-side inspector section, select "Virtual Reality Supported" from Other Settings.

Once done, Click on **Build** button and select a folder (Create it for first time) where the app's code will be saved. As in Figure 3.1 Once Selection done, Unity will start building the solution automatically and will create necessary player and **Visual Studio Solution.**

The entire application after viewing in HoloLens emulator can then be put into the HoloLens device by changing the run option from **HoloLens Emulator** to **Local Machine.** The device needs to be connected to the PC through USB cable and then the run option needs to be clicked. After clicking the run option, the app is installed in the device and it can be run as per the user.

Figure 3.1. *Build Settings in Unity*

C. User Interface

When the user first launches the application, he/she is then greeted with a welcome message. The menu has the following buttons as shown in Figure 3.2:

- "+" button to take a new screenshot
- "X" button to close the panel
- "Mic" button to start taking the voice input

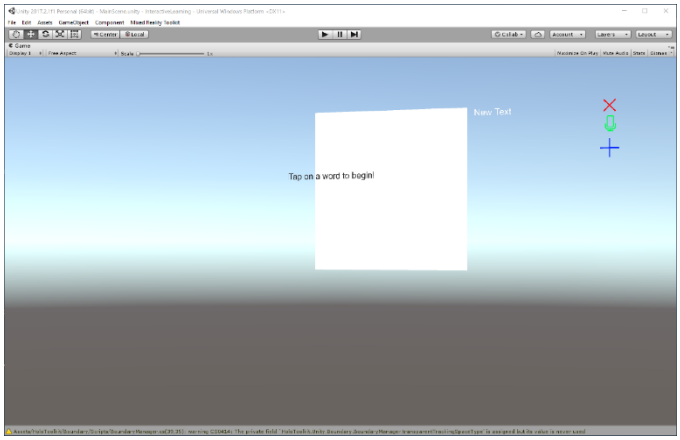

Figure 3.2. *The MainScene – showing the buttons of the menu*

D. User Interface

The user will tap on the '+' button to take a screenshot of the current frame. The snippet below creates, starts and takes a photo. First the PhotoMode is created by invoking the createAsync fuction. Once the system is done creating the photo mode, it will invoke the onPhotoCaptureCreated method. In this method the resolution, height, width, pixel format, opacity etc., camera parameters are set for configuring the main camera. After configuring the camera, photo mode is started. The system will then invoke the method onPhotoModeStarted. The camera is now ready to take a photo, using the TakePhotoAsync method, a new photo is taken. This will delegate to an anonymous method which gets two parameters – PhotoCaptureResult and PhotoCaptureFrame [8]. The second parameter has the raw bytes of the photo taken and also the information on the position, orientation etc., of the camera. This information is then extracted using the TryGetCameraToWorldMatrix method. After taking the screenshot, the photo mode must be stopped to release the resources, this is done by invoking the StopPhotoModeAsync method.

E. Google Vision API

The raw bytes of the images are then converted into base64 format. The request object is then formed as shown in Figure 3.3.

```
{
  "requests":[
    {
      "image":{
        "content": base64String
      },
      "features":[
        {
          "type":"DOCUMENT_TEXT_DETECTION",
          "maxResults":1
        }
      ]
    }
  ]
}
```

Figure 3.3. *Sample request JSON of the Google Vision API*

F. Google Vision API

When a photo is taken, the depth information is lost because photo is in 2D. But to render the annotation frames on top of the corresponding word in 3D space, the depth information is needed Figure 3.4. For this purpose, a ray is casted, the point where the ray cast intersects is taken as the point in 3d space [3].

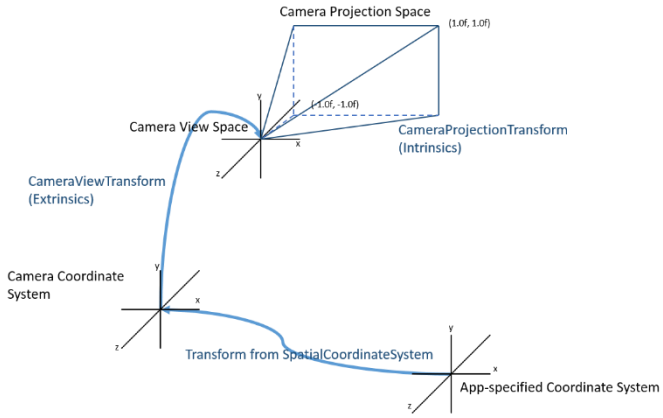

Figure 3.4.. *The Coordinate systems of the HoloLens*

To render the annotations, create four vectors from the four coordinates of a bounding poly and scale them accordingly. The center of these vectors is then calculated to perform a ray cast in the direction given by the center vector. Before performing the ray cast it is important that the camera is in the same position as it was while taking the photo. But this is impossible because in the time taken between requesting, getting response, parsing etc., the user will move and thereby changing the position of the camera. This is why, the camera position information was stored while taking the photo in the cameraWorldMatrix [2]. A secondary camera is then positioned in a way similar to the main camera and this secondary camera is used to perform the raycast. The point where the raycast hits is the centerpoint, this is where the annotation frame will be rendered in 3d space.

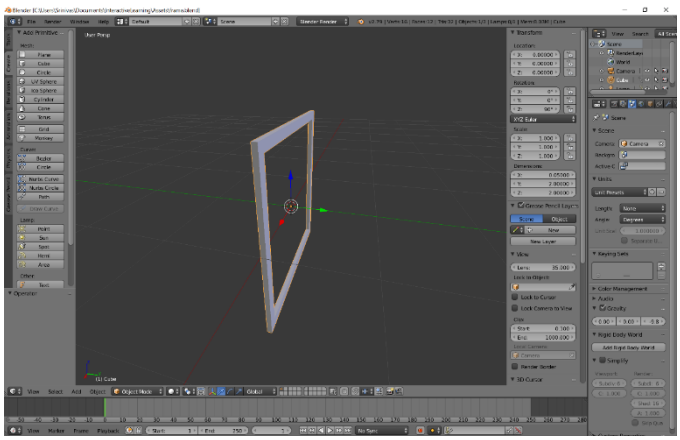

Figure 3.5. *Frame model designed in Blender*

The annotation frame Figure 3.5 now needs to be resized according to the corresponding word size. For this purpose, the top left and bottom left vectors are used to get the height of the annotation frame and the top left and top right vectors are used to get the width of the annotation frame. These calculations in Figure 3.6 are not accurate so the annotations frame may be smaller or bigger than the original word.

```
Vector3 raycastPoint = (topLeft + topRight + bottomRight + bottomLeft) / 4;
Ray ray = raycastCamera.ScreenPointToRay(raycastPoint);

RaycastHit centerHit;
if (Physics.Raycast(ray, out centerHit, 15.0f, m_raycastLayer))
{
    Ray topLeftRay = raycastCamera.ScreenPointToRay(topLeft);
    Ray topRightRay = raycastCamera.ScreenPointToRay(topRight);
    Ray bottomLeftRay = raycastCamera.ScreenPointToRay(bottomLeft);

    float distance = centerHit.distance;
    float goScaleX = Vector3.Distance(topLeftRay.GetPoint(distance), topRightRay.GetPoint(distance));
    float goScaleY = Vector3.Distance(topLeftRay.GetPoint(distance), bottomLeftRay.GetPoint(distance));

    GameObject go = Instantiate(m_annotationTemplate) as GameObject;
    go.transform.SetParent(m_annotationParent.transform);
    go.transform.rotation = Quaternion.LookRotation(Camera.main.transform.forward, Vector3.up);
    go.transform.position = centerHit.point;
    go.transform.localScale = new Vector3(0.1f, goScaleY, goScaleX); //change: inverted x and y

    go.GetComponent<FrameClickScript>().word = child["description"].ToString();

    frameGameObjects.Add(go);

    Debug.Log("Done added");
}
else
{
    Debug.Log("No hit :(");
}
```

Figure 3.6 *Implementation of the Frame annotation renderer*

G. *Google Custom Search API*

When the user taps on an annotation, the corresponding word needs to be looked up. For looking up, Google Custom Search Engine (CSE) Figure 3.7 is used. It is configured in such a way that every website that ends with .edu, .org, .com is parsed to get the results.

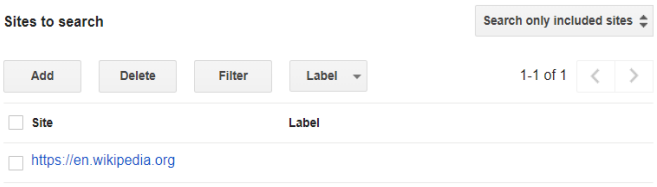

Figure 3.7. *Google Custom Search Engine configuration dashboard*

H. *System Requirements*

To design an application in HoloLens, following requirements are needed.

Hardware Requirements:

- Microsoft HoloLens
- Minimum of 8GB RAM PC or Laptop
- Intel Core i5/i7

Software Requirements:

- Windows 10 Professional
- Unity 5.4 or higher
- Microsoft Visual Studio 2017
- HoloLens Emulator (Hyper-V)

IV. RESULTS

The following are the screenshots of the application from the HoloLens. Here text hand written on a board is used to emulate the class room environment. The screenshots of the words from the monitor are as follows.

As shown in the Figure 4.1, the annotation of the word is displayed, this is rendered after the response from the Google Vision API is received [7]. The annotation frame are instantiated at a distance where the ray cast hits. That will give the 3D information that was lost when a screenshot was taken that is in 2D.

Figure 4.1. *Annotation render of the word*

As shown in the Figure 4.2, the output of the word "photosynthesis" is represented in the form of a textual description and image. The textual description is also spoken out to user using a text-to-speech module implemented in the application. This helps users whose vision is weak to see and read small text.

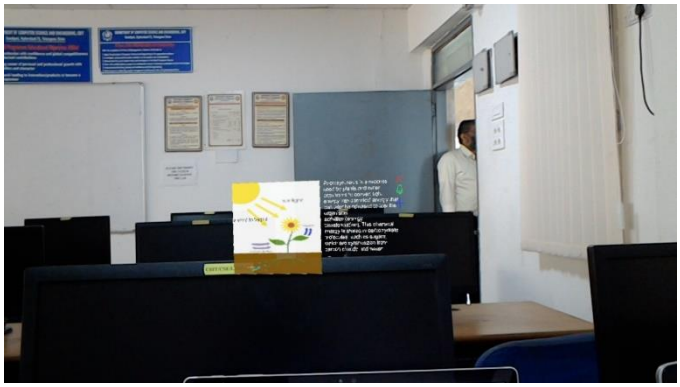

Figure 4.2. *Output of the word clicked by the user*

As shown in Figure 4.3, the output of the sample word "Earth" is depicted. As this word is a planet, the 3D model of the corresponding planet is rendered along with the usual

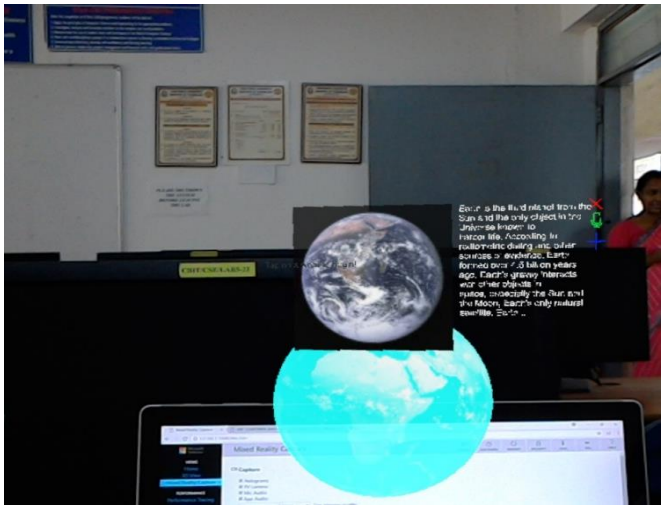

Figure 4.3. *Output with image, text, planet 3D model*

textual description and image thumbnail. The planet model will be rotating to add aesthetic value to the output.

V. Conclusion

It has been demonstrated effectively that an application can be built on the concepts of mixed reality, virtual interactions and reactions to various inputs. With respect to this application, the main motive of design is to understand and utilize various input mechanisms of the HoloLens and link them to make learning interactive. This approach leveraged the fundamental idea of interactions between the user of the HoloLens and the external world very intuitive hence raising interest in the users to learn more effectively in any given environment without any distractions seamlessly. There are some hardware limitations that this application suffers from. As the HoloLens is a wearable standalone device, the computing power is limited even though an Hyper Processing Unit it used. This is the reason why a frame-based approach was chosen over a video stream. In the latter case, the number of API calls was also very high.

Future scope of the project would be to find a way to implement the text annotations on a video stream in real time. This would reduce the number of taps to just one which will directly be on the annotation word and the output will be rendered right away. Another extension would be to add more and more 3D models and video explanations of the queries. The video explanations of the word will also be scraped in future. Named entities like person, organization etc., will be given a new layout with information like net worth, scope etc. With these extensions the application will make it more intuitive and easy to make learning interactive.

Appendix and the use of Supplemental files

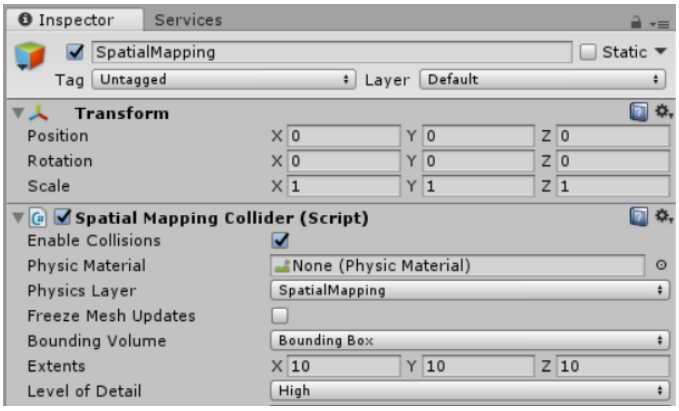

Figure A.1. *Spatial Mapping collider*

Gaze is a first form of input and is a primary form of targeting within mixed reality. Gaze tells where the user is looking in the world and lets us determine their intent. On HoloLens, interactions should generally derive their targeting from the user's gaze, rather than trying to render or interact at the hand's

location directly. Once an interaction has started, relative motions of the hand may be used to control the gesture, as with the manipulation or navigation gesture. With immersive headsets, you can target using either gaze or pointing-capable motion controllers.
Uses of Gaze

- An app can intersect gaze with the holograms in your scene to determine where the user's attention is.
- An app can target gestures and controller presses based on the user's gaze, letting the user select, activate, grab, scroll, or otherwise interact with their holograms.
- An app can let the user place holograms on real-world surfaces, by intersecting their gaze ray with the spatial mapping mesh.

An app can know when the user is not looking in the direction of an important object, which can lead your app to give visual and audio cues to turn towards that object.

Spatial mapping provides a detailed representation of real-world surfaces in the environment around the HoloLens, allowing developers to create a convincing mixed reality experience. By merging the real world with the virtual world, an application can make holograms seem real. The two primary object types used for spatial mapping are the 'Spatial Surface Observer' and the 'Spatial Surface'. The application provides the Spatial Surface Observer with one or more bounding volumes, to define the regions of space in which the application wishes to receive spatial mapping data. For each of these volumes, spatial mapping will provide the application with a set of Spatial Surfaces.

These volumes may be stationary (in a fixed location with respect to the real world) or they may be attached to the HoloLens (they move, but do not rotate, with the HoloLens as it moves through the environment).

The starting point for spatial mapping is the surface observer. Program flow is as follows:

Create a surface observer object

- Provide one or more spatial volumes, to define the regions of interest in which the application wishes to receive spatial mapping data. A spatial volume is simply a shape defining a region of space, such as a sphere or a box.
- Use a spatial volume with a world-locked spatial coordinate system to identify a fixed region of the physical world.
- Use a spatial volume, updated each frame with a body-locked spatial coordinate system, to identify a region of space that moves (but does not rotate) with the user.
- These spatial volumes may be changed later at any time, as the status of the application or the user changes.

Use polling or notification to retrieve information about spatial surfaces

- You may 'poll' the surface observer for spatial surface status at any time. Alternatively, you may register for the surface observer's 'surfaces changed' event, which will notify the application when spatial surfaces have changed.
- For a dynamic spatial volume, such as the view frustum, or a body-locked volume, applications will need to poll for changes each frame by setting the region of interest and then obtaining the current set of spatial surfaces.
- For a static volume, such as a world-locked cube covering a single room, applications may register for the 'surfaces changed' event to be notified when spatial surfaces inside that volume may have changed.

Process the surface changes

- Iterate the provided set of spatial surfaces
- Classify spatial surfaces as added, changed or removed
- For each added or changed spatial surface, if appropriate submit an asynchronous request to receive updated mesh representing the surface's current state at the desired level of detail.

REFERENCES

[1] H. S. Al-khalifa. Utilizing QR Code and Mobile Phones for Blinds and Visually Impaired People. In 11th International Conference on Computers Helping People with Special Needs, pages 1065–1069, 2008

[2] Joonhyun Choi, Boram Yoon, Choongho Jung, "ARClassNote: Augmented Reality based Remote Education Solution with Tag Recognition and Shared Handwritten Note", 2017

[3] Jonathan Huang, "A HoloLens Application to Aid People who are Visually Impaired in Navigation Tasks", 2016

[4] T. Amemiya, J. Yamashita, K. Hirota, and M. Hirose. Virtual leading blocks for the deafblind: A real-time way-finder by verbal-nonverbal hybrid interface and high-density RFID tag space. In Proceedings of the Virtual Reality Annual International Symposium, pages 165–172, 2004

[5] G. Balakrishnan, G. Sainarayanan, R. Nagarajan, and S. Yaacob. Wearable Real-Time Stereo Vision for the Visually Impaired. Engineering Letters, 14(2):6–14, 2007.

[6] D. B. Elliott, M. Trukolo-Ilic, J. G. Strong, R. Pace, A. Plotkin, and P. Bevers. Demographic characteristics of the vision-disabled elderly. Investigative Ophthalmology and Visual Science, 38(12):2566–2575, 1997.

[7] H. Fernandes, P. Costa, V. Filipe, L. Hadjileontiadis, and J. Barroso. Stereo vision in blind navigation assistance. In World Automation Congress (WAC), 2010, pages 1–6, 2010.

[8] R. Smith. An overview of the tesseract OCR engine. In Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pages 629–633, 2007