Conor Brandon and Anthony Scarlatos
May 20, 2022

Virtual Visit: A Web-based Assistive Interface for Touring Cultural Spaces Remotely

1. Introduction

Historically, navigating user interfaces for people with motor disabilities has been difficult. With Virtual Visit, we aim to allow such users to use gaze to navigate interactive 360° panoramas with embedded informative hotspots. Dominant Human Computer Interaction (HCI) models today are built with able-bodied people in mind, leaving disabled users with few options (SU). With spinal cord injuries specifically, disabled users may still retain control of their eyes and head (Al-Rahayfeh), and in other cases, only their eyes, leaving eye tracking as the only method by which they could feasibly use a computer (SU). Estimates for the size of this population range from 250,000 to 450,000 people living with spinal cord injuries in the US, with 17,000 new injuries a year (Spinal Cord Injury).

Modern eye tracking solutions most often use a combination of video cameras and infrared LEDs. While these systems have come down dramatically in price (Zhang), they still require extra and unwieldy hardware (Hachman) and can suffer from steep learning curves (Olivieri). Early eye tracking attempts also discovered the Midas Touch problem that is still trying to be solved today. The problem refers to the tradeoff between passive interaction and active selection. It is difficult to know when a user is simply looking at something, versus when they want to take action on it. If the system is too eager, the user can find themselves stuck in a feedback loop of constant selection (R. G. Lupu).

Over the years, the most common method to make intentional selections has been the Dwell Method. The user fixes their gaze in a target area for a few seconds, after which the selection is registered on the target (Zhang). It's a general method that works across many different types of interfaces, but, as we discovered, suffers from user fatigue and accidental selections and cancellations. Virtual Visit successfully solves these problems by being a fully self contained, web browser based application with a new Bayesian approach for selection built off the work of BayesGaze (Li).

Our focus is on showcasing exhibits from the Paul W. Zuccaire Gallery on the campus of Stony Brook University, although the interface we developed can showcase any location. Another motivation is to enable anyone, regardless of their ability to use computer input devices, to view museum spaces or other cultural locations virtually. The COVID-19 pandemic caused many places to temporarily close and restricted in-person travel, so we hope to convey the experience of moving through and exploring a gallery through the screen in a natural way using gaze.

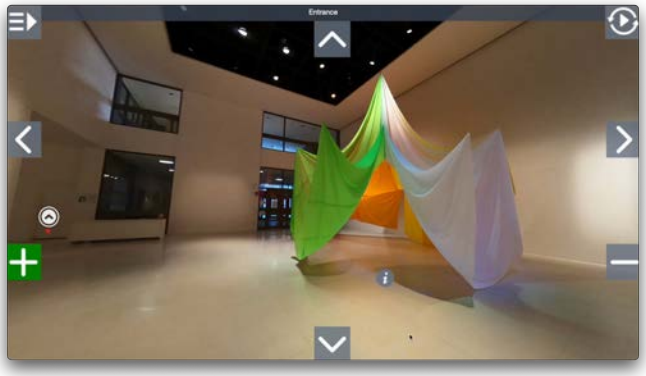
2. The Virtual Visit Application

2.1. Creating a self-contained browser based application

A main focus of Virtual Visit was to create an easy-to-use, functional, and engaging interface all from within the browser. This starts with interactive panoramas. We use the open source library Marzipano to navigate the panoramas and embed informative hotspots in them. Marzipano is completely customizable. We extensively leveraged the ability to edit the HTML, CSS, and Javascript to tweak it for our purposes. We also use an eye tracking library called WebGazer, developed at Brown University (Papoutsaki). The browser based nature of Webgazer means the app doesn't have to send gaze data back to a server for processing, which would have introduced latency and other challenges. Later, we also introduced an open source web framework for creating behavioral experiments, JSPsych (de Leeuw), to help with the calibration phase. Finally, we also had to introduce speech when we found navigating with gaze alone was insufficient. The Web Speech API is a recently new, native web API for fully browser-contained speech recognition. It is currently only fully supported in the Google Chrome browser. With these technologies, Virtual Visit operates entirely within the browser and does not require any special hardware, only a standard webcam and microphone.

2.2. Original Interface and Dwell Method

The original Marzipano interface provides all the functionality needed to move horizontally, vertically, zoom in/out, expand the info hotspots, and switch scenes. With the Dwell Method, the user would attempt to fix their gaze in a bounding box of 200 by 200 pixels for three seconds. At *time=0*, the closest target would be found using the distance from the gaze dot in pixels, and over the course of three seconds, the closest target would change from gray to red to yellow and finally flash green when it was selected. If the user broke outside of the bounds of the box, even for an instant, the entire selection would be reset and the algorithm had to restart from scratch.

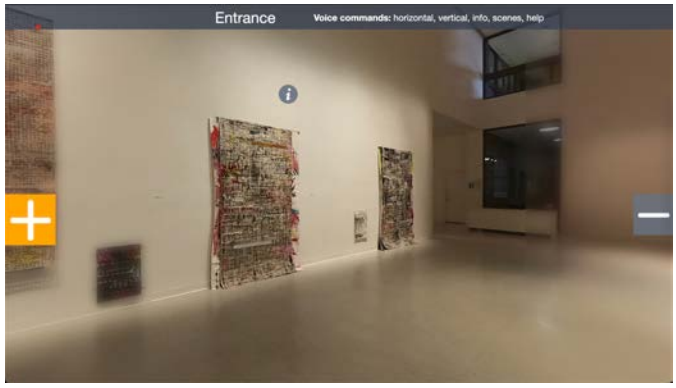


2.3. Redesigning the original interface for speech

It was immediately apparent that there were too many targets on screen at a time for the eye tracking to be accurate. WebGazer was the perfect library for our needs, but the accuracy of the tracking was not good enough, especially in challenging situations like low light. We realized not all modalities had to be present on screen at a time, so we introduced the open source Web Speech API.

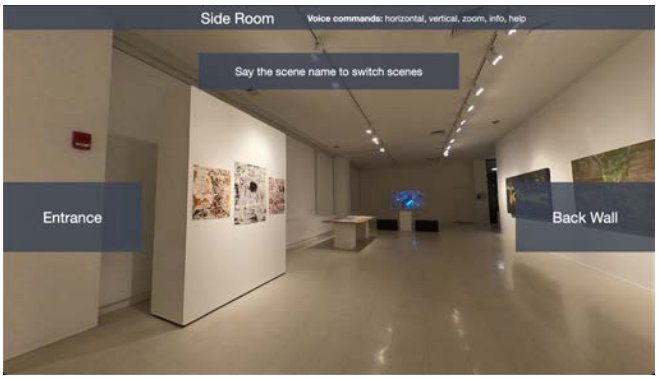
The redesigned interface moved the navigation buttons to the edges of the screen, centered vertically, and uses speech to switch between modalities. The different modalities are: “horizontal”, “vertical”, “zoom”, “info”, “scenes”, and “help”. Simply say one of these commands to switch to that modality, and the commands are displayed along the top of the interface for reference.

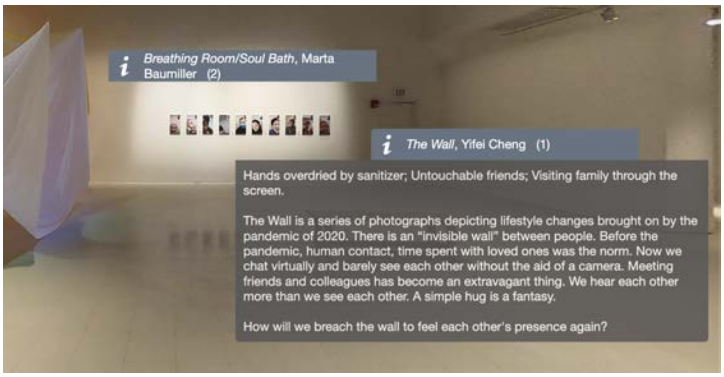
In horizontal, vertical, and zoom modes, two buttons are displayed on the edges of the screen (i.e. zoom in, zoom out).



The left and right edge placement of the buttons is very intentional. We found eye tracking to be more accurate on the horizontal axis than the vertical axis, due to more of the whites of your eyes being visible when you look horizontally rather than vertically. We added slight blurring of the left and right edges to mimic peripheral vision and to indicate to the user where their gaze is actively being tracked. The red gaze dot—visible in the upper left corner of the left screenshot above—is hidden in the center of the screen and appears on the edges only when the predictions are actually being used in the selection algorithm. This has the advantage of the center being a “safe zone” where the user can inspect the scene without also having to avoid selecting a target.

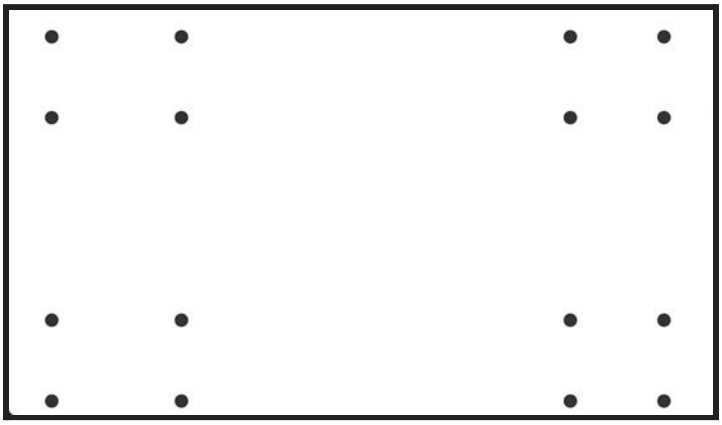
We also use speech to select scenes and open info hotspots. Initially, scenes were displayed in each corner of the screen. But, this allowed for a maximum of five scenes (as the current scene wasn’t displayed). Scene names are now displayed horizontally across the screen, allowing for a flexible number of scenes, and can be selected by saying the scene name. Info hotspots can be placed wherever there is more to learn about an object, and can be opened by saying “show info <number>”, where the number is displayed in the info header, and can be hidden with “hide info”.





The advantage of using speech in these modes is the user doesn't have to be concerned with determining which scene they want to select and either trying to select it, or *avoid* it getting selected while they browse their options. This problem is even worse in info mode.

By specifically targeting the edges of the screen, we were also able to reduce the complexity of the calibration process. We switched from random points to vertical lines on the left and right edges of the screen. These points are placed in the areas where the tracking must be good.



2.4. Switching to a Bayesian approach

The redesigned interface was a big improvement over the original incarnation of the app. While the problem of choosing the target to perform the selection on was alleviated through the aforementioned interface changes, the process of selection with the Dwell Method was flawed. WebGazer's accuracy limitations would cause the gaze dot to jump around. This skittering behavior was further worsened by saccades, which are small, imperceptible, and unintentional rapid eye movements between fixation points. Additionally, user fatigue was high with the Dwell Method because users had to lock their gaze in the bounding box. If a detection accuracy issue, saccade, or user fatigue caused the gaze dot to move outside the bounding box, the entire selection would be canceled and the user would have to start from scratch.

Our breakthrough was coming across a new approach from Zhi Li, et al., also developed at Stony Brook University. BayesGaze builds on the Center of Gravity Mapping Method from Bernhard, et al. In BayesGaze, a target "accumulates interest" based on the distance the gaze position is from the target. Once the interest *I* surpasses a threshold $\boldsymbol{\theta}$, the target is selected and the interest for all targets is reset to zero:

1. If $I_i(t_j) > \theta$, then select the target. Reset interest to 0 for all targets.

Interest accumulates by adding the previous interest to the current calculated interest. The current interest is calculated by multiplying the "sampling interval", $\Delta\boldsymbol{\tau}$, by the posterior probability from Bayes Theorem, $P(t|s_i)$:

2. $$I_i(t) = I_{i-1}(t) + \Delta\tau \cdot P(t|s_i)$$

The posterior probability, $P(t|s_i)$, is the application of Bayes Theorem in this context:

3. $$P(t|s_i) = \frac{P(s_i|t)P(t)}{P(s_i)} = \frac{P(s_i|t)P(t)}{\sum_{j=1}^{N} P(s_i|t_j)P(t_j)}$$

Interestingly, we can ignore two terms from this equation. The prior, $P(t)$, is the probability of looking at any given *t*arget. For our purposes, this is uniformly distributed and can be ignored because we only have two targets on-screen at any given time. We can also ignore the evidence, $P(s_i)$. This is the probability of looking at any point on the *s*creen at any given time, and the user is equally likely to be looking anywhere. That leaves the likelihood term, $P(s_i|t)$.

4. $$P(s_i|t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{||s_i - c_t||^2}{2\sigma^2}\right)$$

$\boldsymbol{\sigma}$ is a parameter that controls the rate at which interest accumulates based on the distance from the target the user is looking. Since it is in the denominator, if $\boldsymbol{\sigma}$ is too high, then it will be hard to accumulate interest at larger distances from the target. In the numerator of the exponent, the actual distance the user's gaze is from the target is calculated: $|| s_i - c_t ||^2$. Since we only end up using the likelihood term, this approach ends up essentially becoming the Center of Gravity Mapping Method (Bernhard).

This was not a plug-and-play solution initially. BayesGaze was a more theoretical approach in a controlled environment. Distance was measured in centimeters and the sampling interval was more consistent because they used higher-end hardware. In our Javascript code, the only measurement units we have are pixels, so we had to scale the parameters to match. We

settled on $\Theta = 2$, $\sigma = 400$ pixels, and $\Delta\tau = 13$ or 15 units. In order to make the tail end of the selection process quicker, $\Delta\tau$ is increased by 2 units once the interest has reached $\Theta / 2$. When interest is low at the beginning, it makes sense to be less aggressive at first to make sure the user isn't accidentally accumulating interest. Our thinking is once interest is greater than $\Theta / 2$, it is likely that you really do intend to select that target. As interest accumulates, it is visible in the increased size of the target and changing the color from gray to orange to a flash of green upon selection.

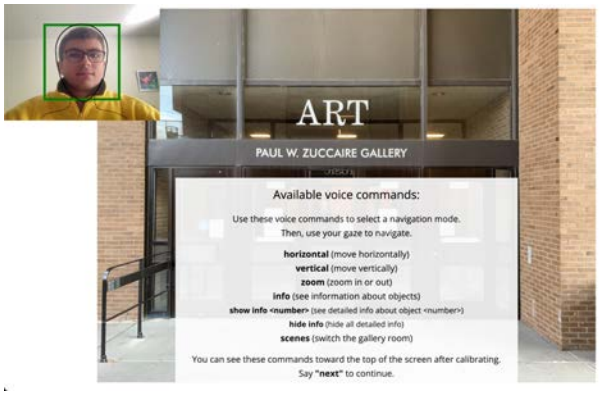
In addition, gaze position is only used in the calculation on the blurred edges of the screen. If the gaze dot is hidden, the user doesn't know exactly where they're looking and will get confused if targets start growing or changing color while they are trying to inspect the scene, reference voice commands, or read text in info mode.

This yielded immediate improvements. While the Dwell Method was frustrating and prone to errors, interest accumulation is resilient to accidental intrusions into the gaze area on the edges and doesn't immediately cancel a selection when you move out of the gaze area. Interest remains accumulated and only increases over time until a successful selection is made. We were also able to reduce calibration time further because this approach is simply better, and it is more resilient to inaccurate tracking.

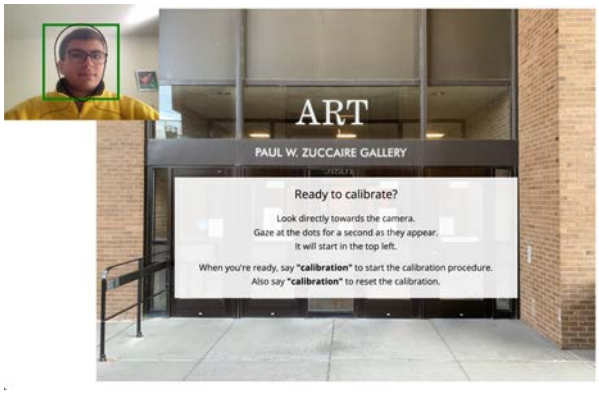
2.5. Creating a great user experience

In order to make this application something people will actually want to use, it was imperative that the user knew exactly how to use the app and could perform the setup and calibration using speech. JSPsych made it easy to create a great start-up experience by using the library's built-in calibration plugin and other helper functions to control WebGazer's behavior. The following screenshots show the instruction and calibration screens. Screen 2 is duplicated for the help screen, accessible to the user mid-use, if they want to review the commands again or recalibrate.

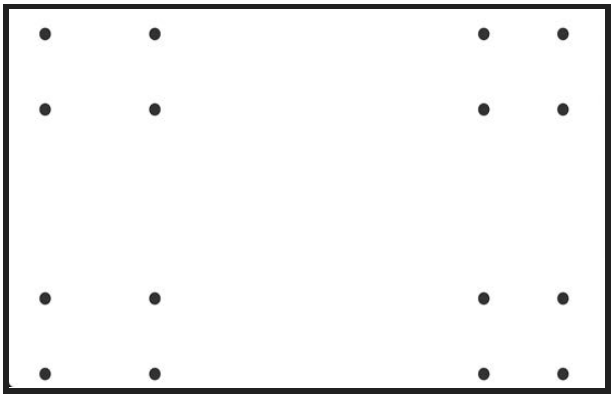
1



2



3

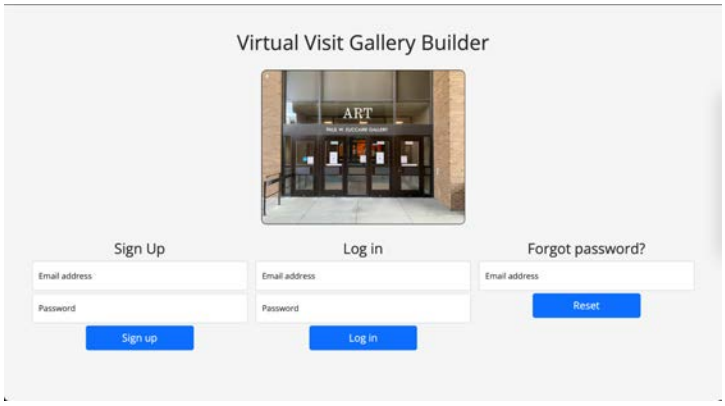


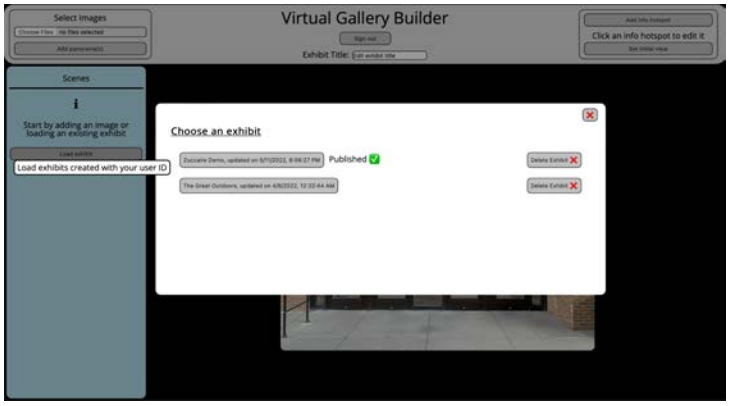
4

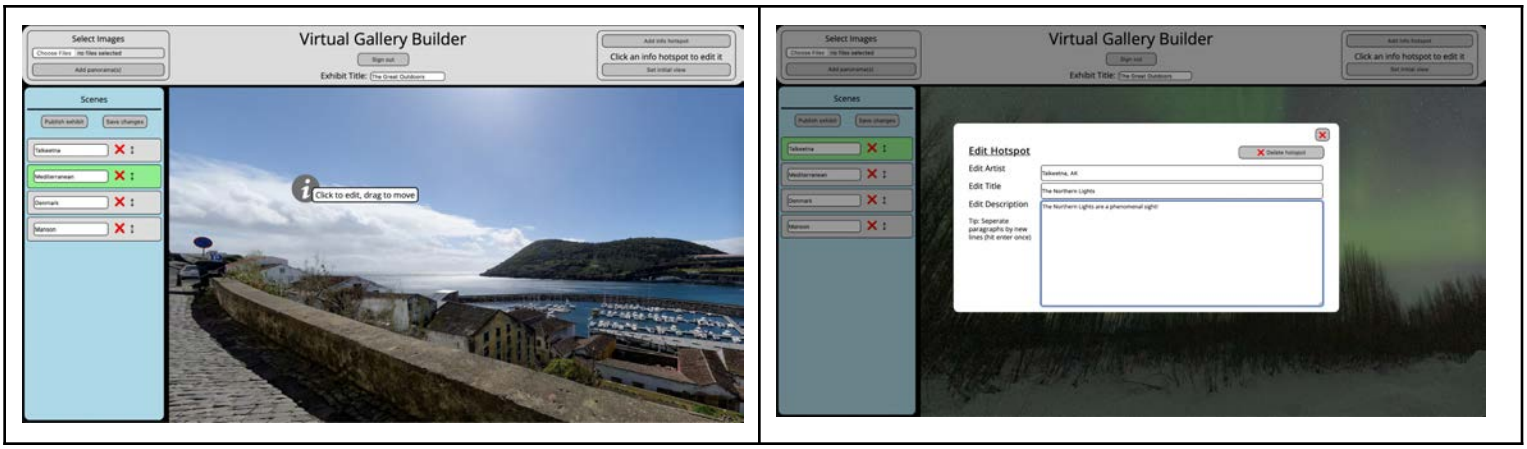
3. Virtual Visit Backend Manager

3.1. Backend Manager

One of the goals for this project was to create a way for a gallery director or other designer to create and publish future exhibits. The Virtual Visit Backend Manager allows managers to upload panoramic images, rename and reorder scenes, add/position/edit info hotspots, set an initial view for the scene, and save it for future editing or publish the exhibit—at which point it becomes the current exhibit on the frontend interface.







3.2. Backend Design

Virtual Visit uses Amazon Web Services (AWS) for all backend tasks: login and signup, database, API calls, and images. One of the other great features of Marzipano is its simple data structure. This is a representation of what is stored in our database, keeping expenses minimal.


```
{
  "scenes": [
    {
      "id": "0-entrance",
      "name": "Entrance",
      "initialViewParameters": {
        "yaw": 0,
        "pitch": 0,
        "fov": 1.5566593982312988
      },
      "infoHotspots": [
        {
          "yaw": 0.5701939069509017,
          "pitch": -0.0028290423685461974,
          "title": "<p><span class='italics'>The Wall</span>, Yifei Cheng</p>",
          "text": "<p>Hands overdried by sanitizer; Untouchable friends; Visiting family through the screen.</p>..."
        },
          ...
      ]
    ...
  ],
  "name": "Zuccaire Gallery",
  "settings": {
    "mouseViewMode": "drag",
    "autorotateEnabled": false,
    "fullscreenButton": false,
    "viewControlButtons": true
  }
}
```


Each exhibit consists of an array of scenes with initial view parameters. Each scene has an array of info hotspots and the scene name and ID. There are a few other configuration settings at the bottom as well. Marzipano can load images from the file system or from an HTTP request, making this incredibly dynamic for different exhibits.

4. Future work

In the future, we hope to field test the system with subjects from the target population (disabled users, specifically those with spinal cord injuries). With the subject's feedback, we hope to improve the system and tweak the parameters to increase usability. Their feedback would be collected based on the Technology Acceptance Model and System Usability Scale, two

standardized approaches for gathering user feedback on new technology and the effectiveness of our implementation (Zhang).

5. Conclusion

Virtual Visit successfully mimics the experience of visiting a gallery all from within the browser. It blends multiple open source technologies together in a way that is novel and highly functional: Marzipano for interactive panoramas, Webgazer for eye tracking, JSPsych for setup and calibration, and the Web Speech API for speech recognition. It demonstrates that the theoretical model of BayesGaze can work in a real world application and works better than the Dwell Method. In the future we hope to test the system with disabled users and that the Backend Manager can be used to create future exhibits of the Zuccaire Gallery so that anyone, regardless of ability, can enjoy the experience of visiting a gallery.

Works Cited

Al-Rahayfeh, Amer, and Miad Faezipour. “Eye Tracking and Head Movement Detection: A STATE-OF-ART SURVEY.” IEEE Journal of Translational Engineering in Health and Medicine, vol. 1, 6 Nov. 2013, pp. 2100212–2100212., doi:10.1109/jtehm.2013.2289879.

Bernhard, Matthias & Stavrakis, Efstathios & Hecher, Michael & Wimmer, Michael. (2014). Gaze-To-Object Mapping During Visual Search in 3D Virtual Environments. ACM Transactions on Applied Perception. 11. 10.1145/2644812.

de Leeuw, J. R. (2015). jsPsych: A JavaScript library for creating behavioral experiments in a web browser. Behavior Research Methods, 47(1), 1-12. doi:10.3758/s13428-014-0458-y.

Google. “Marzipano.” Github, 7936a4e0791fcfe55d199c03080be4f89ccf71bd, 22 Feb. 2016, https://github.com/google/marzipano.

Hachman, Mark. “Tobii EyeX Review: The 'Eye Mouse' Is Magical, but Just Not for Everyone.” PCWorld, PCWorld, 17 Feb. 2016, www.pcworld.com/article/3014523/tobii-eyex-review-the-eye-mouse-is-magical-but-just-not-for-everyone.html.

Li, Zhi et al. “BayesGaze: A Bayesian Approach to Eye-Gaze Based Target Selection.” Proceedings. Graphics Interface (Conference)vol. 2021 (2021): 231-240. doi:10.20380/GI2021.35

MDN. Web Speech API, Mozilla, https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API.

Olivieri, P.. “EagleEyes: An Eye Control System for Persons with Disabilities.” (2013).

Papoutsaki, Alexandra, et al. “WebGazer: Scalable Webcam Eye Tracking Using User Interactions”. Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI). AAAI, 2016. 3839–3845. Print.

R. G. Lupu, F. Ungureanu and V. Siriteanu, "Eye tracking mouse for human computer interaction," 2013 E-Health and Bioengineering Conference (EHB), 2013, pp. 1-4, doi: 10.1109/EHB.2013.6707244.

SU, MU-CHUN, et al. “An Eye Tracking System and Its Application in Aids for People with Severe Disabilities.” Biomedical Engineering: Applications, Basis and Communications, vol. 18, no. 06, 2006, pp. 319–327., doi:10.4015/s1016237206000476.

Zhang, Xuebai, et al. “Eye Tracking Based Control System for Natural Human-Computer Interaction.” Computational Intelligence and Neuroscience, vol. 2017, 2017, pp. 1–9., doi:10.1155/2017/5739301.