Medical Tutorial Generation
&
Performance Evaluation Using Eye Gaze Tracking Data

Team Members: Allan Wang & Prateek Bhatnagar

Mentors: Ehsan Azimi, Dr. Chien-Ming Huang, Dr. Peter Kazanzides, Dr. Nassir Navab, and Dr. Camilo Molina

Table of Contents

1. Background

Augmented reality (AR) technology has been used in a number of surgical applications, helping the surgeon visualize surgical instrument placement. AR systems in the past have been used in more advanced and expensive applications in both medicine and telemedicine; however, many of these implementations have proved to be either too complicated for users or very difficult to set up and maintain. AR has the potential to both train students and assist surgeries in real time. By providing the user with relevant images and/or instructions, AR software in a head-mounted display (HMD) can provide a convenient source of information for doctors and other medical professionals. We have an existing framework that can run and display tutorials as demonstrated in the following diagram.

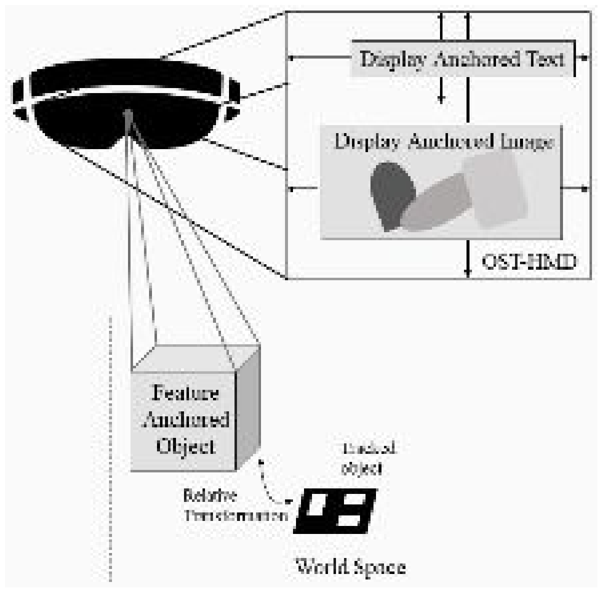


Figure 1. Existing tutorial framework.

1.1. Project Goal

Our project's goal was to create an HMD-based software tool that allows the user to create HMD tutorials with text and images. In addition, we planned to create a module for tracking the user's eye gaze and providing a heatmap of an expert user's gaze to compare to a that of a novice user. This toolset could then be used by the existing training application that runs on the HoloLens. This toolkit would allow for seamless creation of training experiences for the training application as well as provide gaze data for analysis. Group 9 worked in parallel with us to develop other tools to facilitate "smart" training with live feedback and analysis.

The tutorial generation module works on providing an intuitive way to create new training modules for the training application using voice commands. The eye gaze module tracks the eye gaze of the user during the medical procedure to analyze the positions in the scene they are looking at this may be used for performance evaluation and UI optimization.

1.2. Relevance

Use of HMDs for training of medical procedures provides an intuitive and repeatable way to teach complex techniques and develop muscle memory. In their current state the creation of new medical procedure modules for this type of training is done by hand and is a long and time intensive activity. Furthermore, there exist very few performance analytics, or supporting information of the test subjects in the current training systems based on AR HMDs. Most AR HMD-based medical training systems are in their infancy. There are no tools that allow for the average medical professional to easily create or utilize HMD-based tutorials. In order to promote acceptance of these types of medical training systems, ease of use and ability to provide analytics and intelligent feedback based on the users would be would be key factors, and our modules aim to satisfy these needs.

The HoloLens was chosen for this project as it is a non-occluding AR HMD and has a first-person camera. These features allow for non-intrusive tutorials and for easy image capture for tutorial generation. Moreover, Pupil Labs offered a binocular add-on that collects eye gaze tracking data.

2. Technical Approach

Software tools for this project were developed in a combination of Unity, C#, and Python. The tutorial generation app was implemented in Unity with C# scripts. The eye gaze tracking module was developed in Python, with a HoloLens frontend in Unity and C#.

2.1. Tutorial Generation

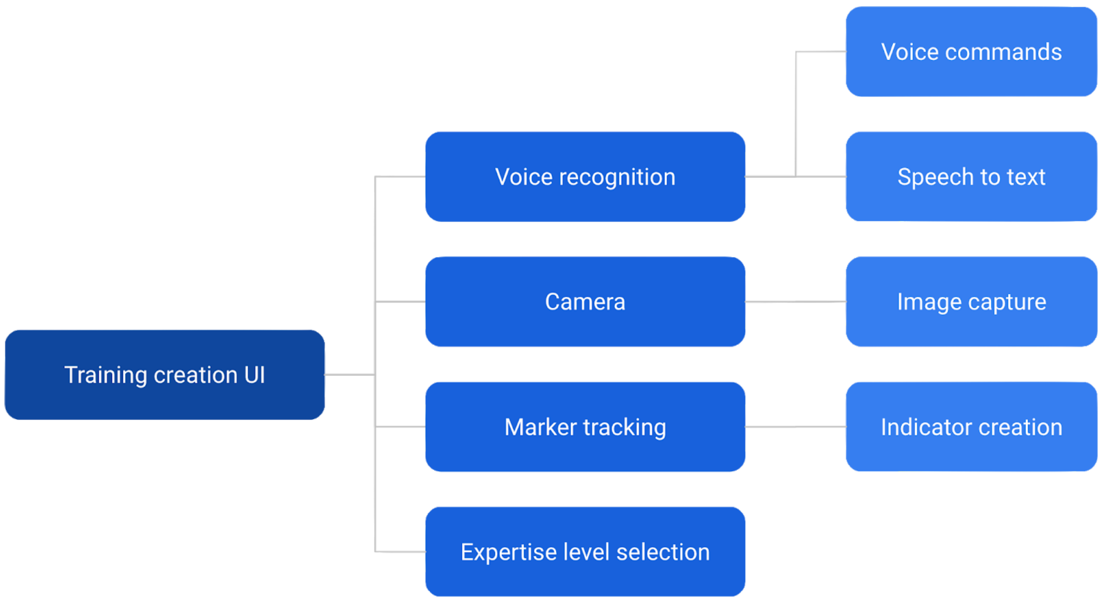


Figure 2. Initial plan for modular development.

We originally planned for the primary modules of the tutorial generation app to manage photo capture, dictation, voice commands, and AR markers. The module for expertise levels in tutorials would have been an integration of our tools with the other group's "smart training" project. Similarly to the existing software, user-generated text and images would be overlayed in the real world view and still provide a relatively unobstructed view of the relevant subjects or objects.

JSON files are somewhat difficult to manually create and edit, so dictation provided a more intuitive alternative for tutorial creation; additionally, voice commands allow hands-free usage of the HMD. This entire app was planned to run entirely on the HoloLens, using a separate computer only to move files as needed.

2.2. Eye Gaze Tracking: Heatmap Generation

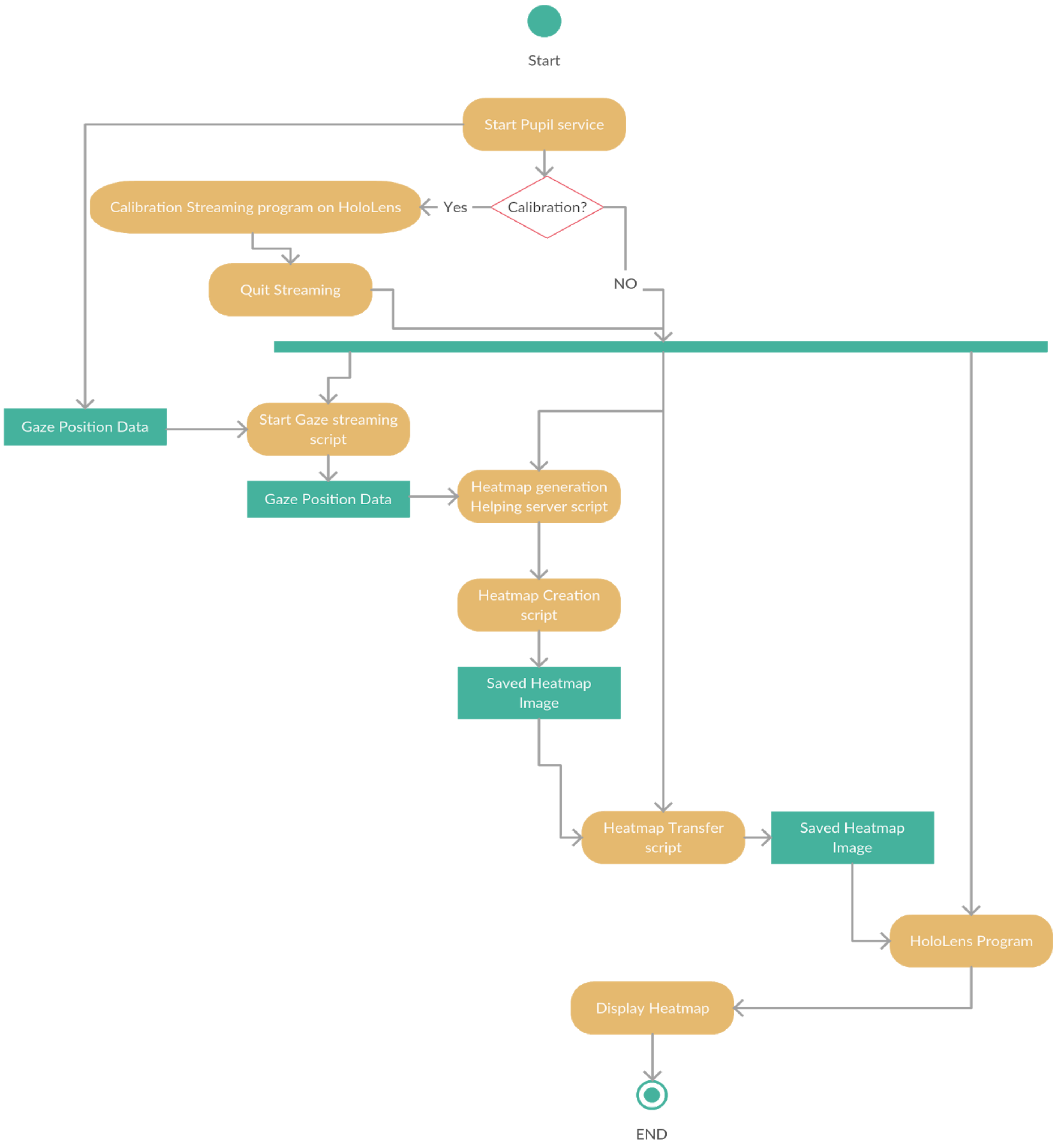


Figure 3. The workflow for the eye gaze data tracking.

2.2.1. Components of the Eye Gaze Module

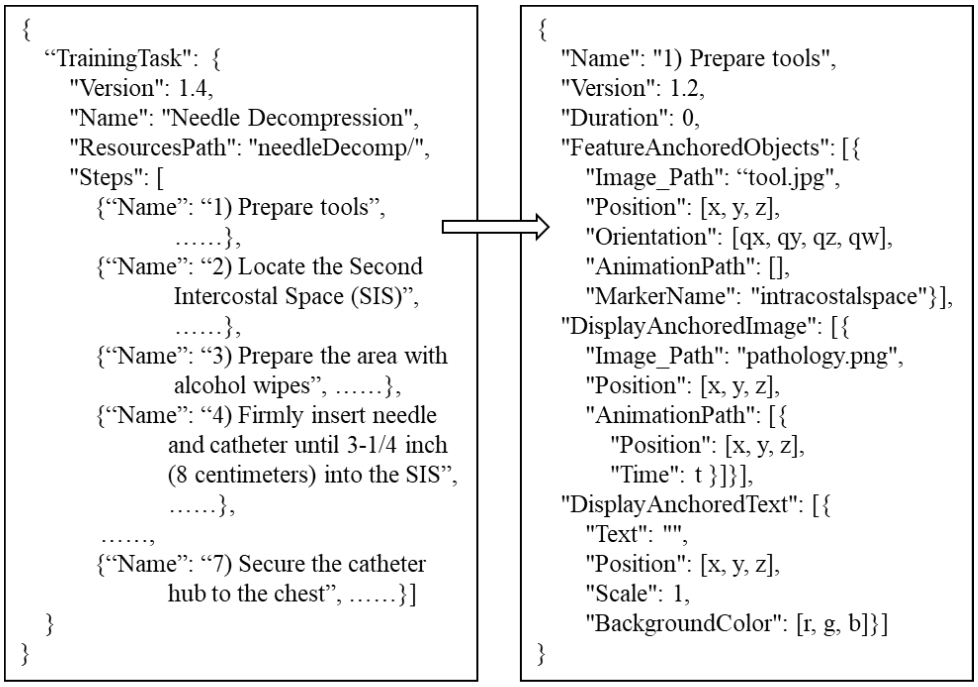
1. Pupil Labs Service: Pupil Labs proprietary software required to interface with the Pupil Labs cameras. Stores Calibration information, as well as facilitates transfer of gaze coordinate positions to python scripts via a *zmq* networking paradigm.
2. Calibration Program for Hololens: Unity project provided by Pupil Labs to allow for calibrating the HoloLens user's eye gaze to with the pupil labs service.
3. Gaze Streaming: A script that starts a *zmq* subscriber to listen to the gaze coordinate positions from the pupil labs service then streams this data over a UDP connection to required destinations on the python scripts running in parallel.
4. Heatmap Server Script: Receives raw gaze coordinate data that is then transformed to the screen frame coordinates for the known display screen size of the HoloLens (approximately 720p). Coordinates are cleaned then transferred to the Heatmap Creation script.
5. Heatmap Creation Script: Stores a history of gaze coordinate data and creates a heatmap image from the data. The image is updated after a set number of entries.
6. Image Transfer Script: Code to transfer image of the heatmap using a TCP/IP connection to the HoloLens.
7. Hololens Scene: All the required code for displaying the heatmap image is condensed within a unity canvas and a couple of managing components allowing for easy integration with other unity applications

3. Development

The content generation app was developed in Unity 5 and Visual Studio (VS) 2017 with C# scripts. Pupil Labs libraries and scripts were in C/C++ and Python was used to stream data from a computer to the HoloLens. Like the content generation app, the frontend for heatmap display was done in Unity 5 and VS 2017.

3.1. Tutorial Generation

The goal of this component of the project was an HMD-based app that would easily allow a user to generate JSON files and images that can be ported to the existing training application developed by our mentor, Ehsan Azimi. As shown in Figure 4, the JSON file includes the text displayed at each step, as well as the filepaths needed to access the required images. The training app then reads these files and displays the tutorial: the tutorial is controlled through voice commands.



```
{
  “TrainingTask”: {
    "Version": 1.4,
    "Name": "Needle Decompression",
    "ResourcesPath": "needleDecomp/",
    "Steps": [
      {“Name”: “1) Prepare tools”,
              ……},
      {“Name”: “2) Locate the Second
              Intercostal Space (SIS)”,
              ……},
      {“Name”: “3) Prepare the area with
              alcohol wipes”, ……},
      {“Name”: “4) Firmly insert needle
              and catheter until 3-1/4 inch
              (8 centimeters) into the SIS”,
              ……},
      ……,
      {“Name”: “7) Secure the catheter
              hub to the chest”, ……}]
  }
}
```

```
{
  "Name": "1) Prepare tools",
  "Version": 1.2,
  "Duration": 0,
  "FeatureAnchoredObjects": [{
    "Image_Path": “tool.jpg",
    "Position": [x, y, z],
    "Orientation": [qx, qy, qz, qw],
    "AnimationPath": [],
    "MarkerName": "intracostalspace"}],
  "DisplayAnchoredImage": [{
    "Image_Path": "pathology.png",
    "Position": [x, y, z],
    "AnimationPath": [{
      "Position": [x, y, z],
      "Time": t }]}],
  "DisplayAnchoredText": [{
    "Text": "",
    "Position": [x, y, z],
    "Scale": 1,
    "BackgroundColor": [r, g, b]}]
}
```

Figure 4. Excerpt from JSON training file.

The first step in development was using Unity to create GameObjects that would display various text-based information and images. These displays included the step count, current step’s text, current image capture, camera status, and some instructions. These objects were placed on a Canvas object that would follow the gaze of the HoloLens user, rather than staying at a fixed location in the HoloLen’s virtual space. Next, Microsoft’s HoloToolkit libraries for speech input and dictation were imported under a “Managers” GameObject. The managers listened for specific keywords and handled dictation interpretation.

Using the aforementioned libraries, an all-purpose SpeechHandler C# script was implemented to perform various actions after certain commands were given. Specific keywords, such as “Next,” “Start recording,” or “Snap,” were assigned to specific SpeechHandler functions through Unity. SimpleJSON, a publicly available JSON parsing class, was used to save and index tutorial data. Overall, SpeechHandler managed all operations to create the JSON file. The ImageCapture GameObject and C# script handled image capture, display, and saving. The filename was a public field that SpeechHandler could access, save, and then write to the JSON file. More details on the software's structure can be found in Appendix A.

3.2. Eye Gaze Tracking: Heatmap Generation

Our initial approach included understanding Unity as a part of the HoloLens deployment pipeline. This took significantly more time than expected due to the duality of a Unity UWP application, these applications are able to utilise Unity libraries while testing on the editor however when built with visual studio they are able to utilise the UWP libraries. As we explored further we realised certain limitations with respect to freedom of accessibility on the HoloLens especially regarding the ability to transfer files programmatically. Next the Pupil Labs pipeline was explored to understand the extraction and interaction of gaze data. Initial attempts were made to create our own calibration client however in the end we decided to use the one provided by Pupil Labs due to ease of use and accuracy. Once the gaze data was streaming in, setting up the heatmap code was straightforward. The next problem to be solved was transferring this heatmap image to the HoloLens itself. The initial implementation did not actually run as an application on the HoloLens itself but was rather streamed onto it using Unity Remote connect and the unity editor. For the final implementation, the required code for HoloLens was condensed down to fewer Unity Objects that can be easily exported and integrated into other projects.

3.3. Obstacles

Our first obstacle in development was learning how to use Unity as a HoloLens app development tool. While it did provide convenient ways to connect various components and scripts, we experienced many bugs due to Unity version changes and inconsistent (and sometimes absent) compiler warnings and debugging tools. Our Unity projects would sometimes compile "successfully" but would be missing various assets in the HoloLens deployment.

Other challenges were a result of the proprietary nature of the Microsoft HoloLens. When developing the file-writing portions of our software, we encountered issues with writing JSON and image files. The HoloLens only allows write permissions in the local app folder, a few main library folders (Pictures, Videos, etc.), and OneDrive. As such, it was not possible to directly write JSON files, images, or folders to the training application for direct access after creation.

4. Results

The final tutorial generation app creates JSON files associated with image files that are compatible with the current framework. The interface closely follows the layout of the original training software, with additional visual components that are useful to a tutorial creator.

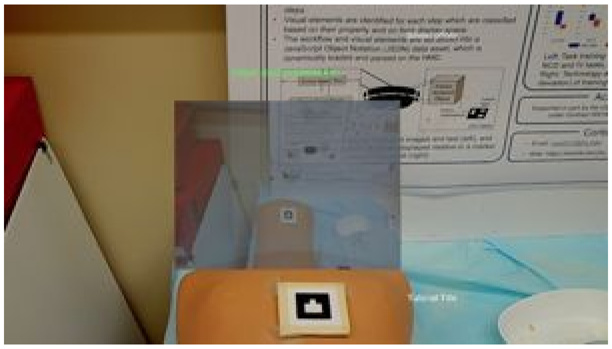


Figure 5. User view of tutorial generation app.

Voice commands are used to navigate the tutorial as detailed in Table 1 below. UI elements include camera status, step count, and a color indicator for recording status. Finally, the JSON and image outputs could be successfully transferred to and used in the original software with no alterations to the files.

Table 1. Available Voice Commands in the Tutorial Generation App.

Phrase	**Action**
Start recording	Begins speech-to-text recording for current step. Can be repeated to overwrite incorrect text.
Snap	Takes picture, which is saved locally and in the pictures library. Can be repeated multiple times per step to overwrite unsatisfactory images.
Next	Resets screen, increments step count, and moves to next section of the JSON file.
Finish	Concludes the tutorial and saves the JSON file locally.

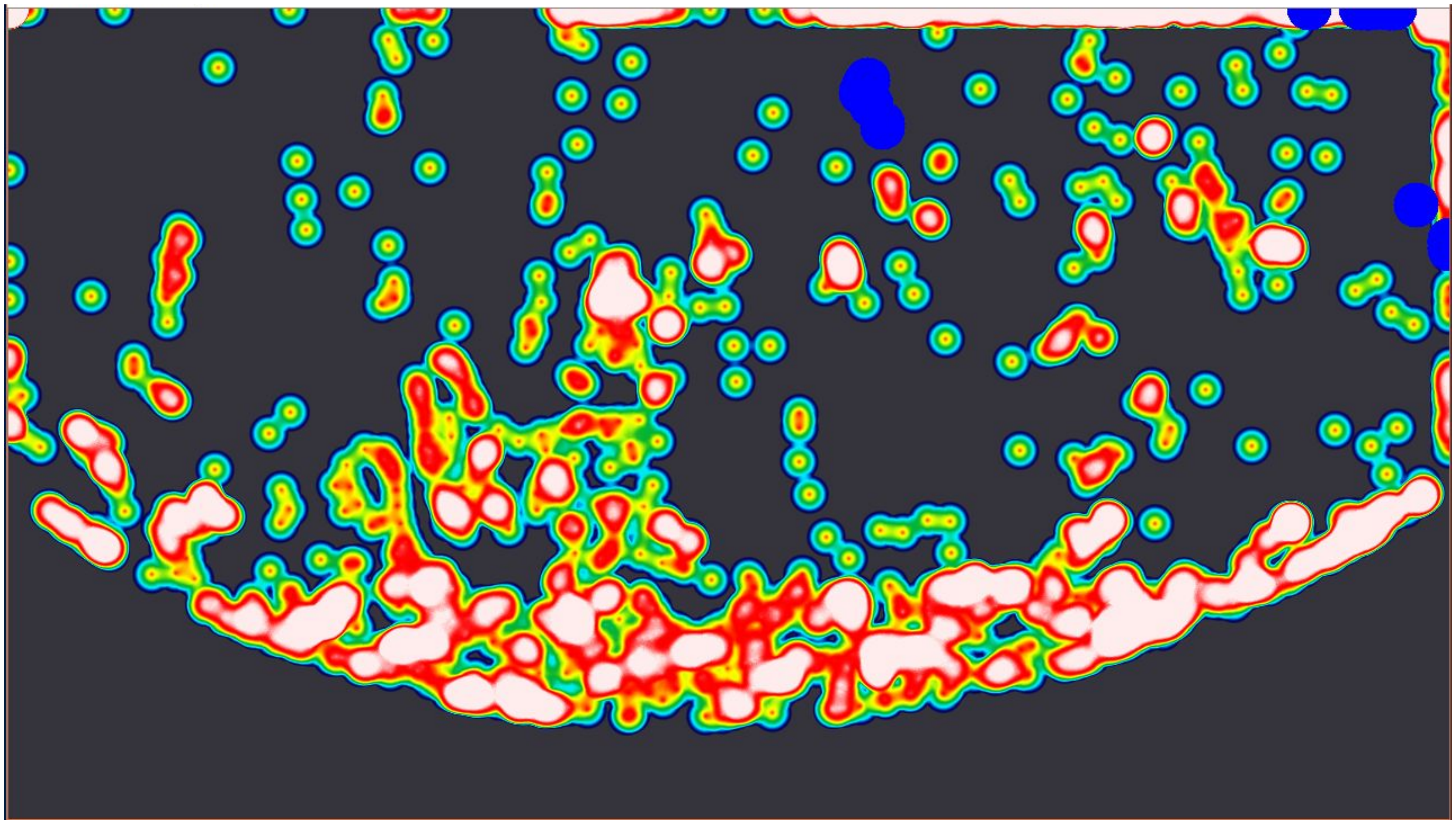


Figure 6. Heatmap with Gaze Point (Gaze Point Sprite Represented in BLUE).

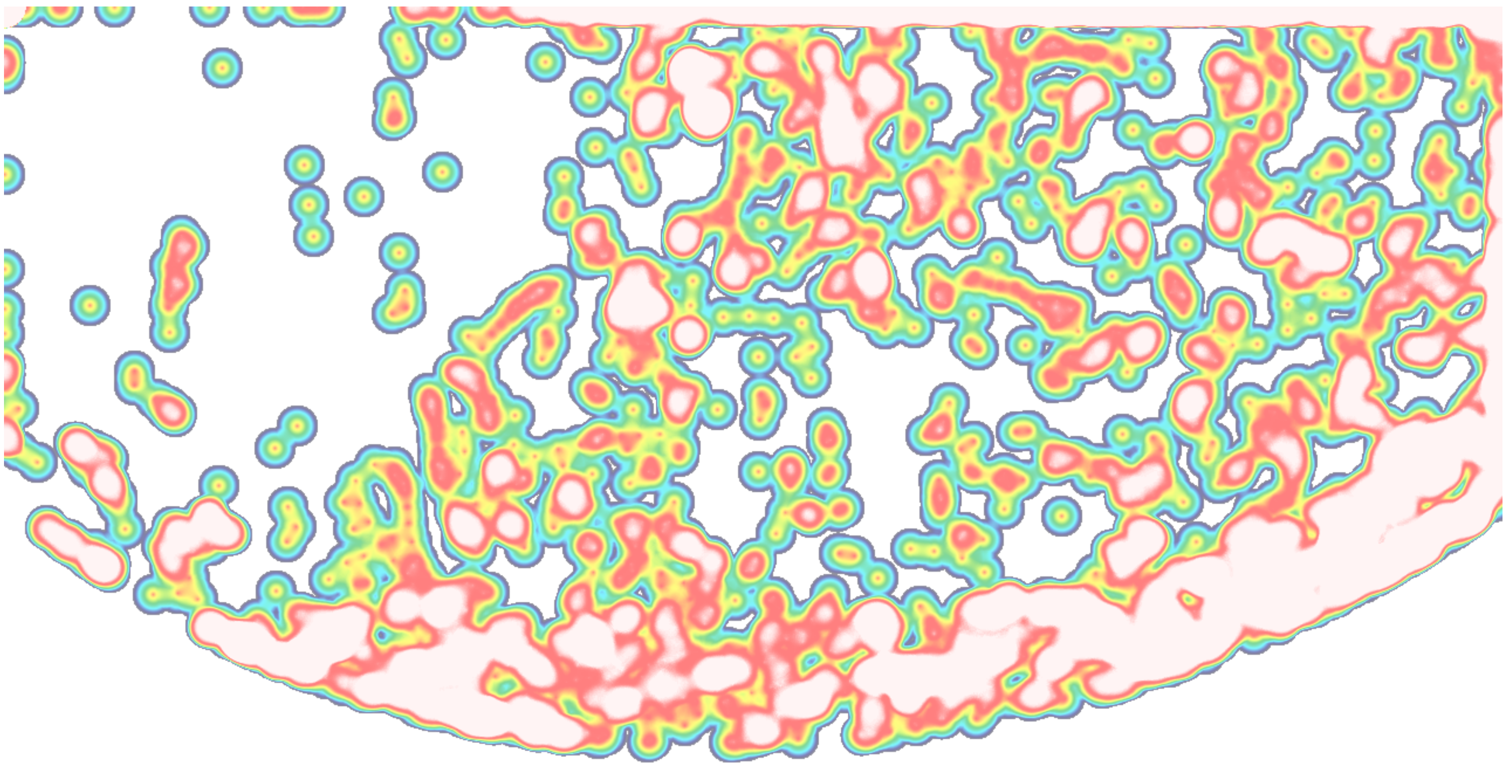


Figure 7. Heatmap Display Rendered on HoloLens.

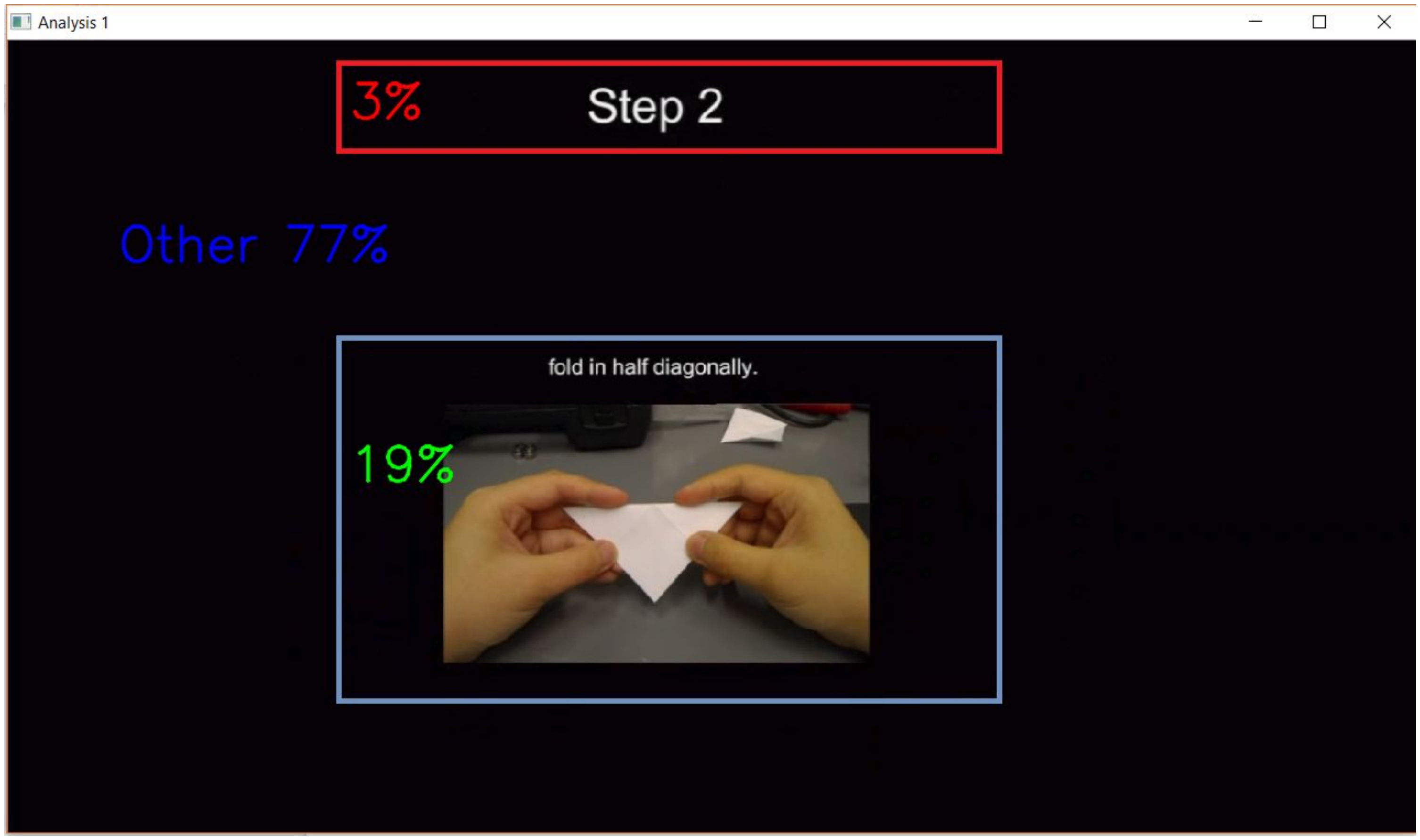


Figure 8. Basic Analytics Provided through Python on the PC

4.1. Actual Results vs. Planned Results

Virtual marker creation and expertise levels were removed from our maximum deliverables due to the additional time needed to fix the bugs mentioned in Section 3.3 and integrate our respective project components. Our integration efforts primarily went into our own components. Virtual marker creation proved to be beyond the scope of the remaining time, as it required the integration of even more toolkits to live-track a tool in order to place objects in the 3D virtual space. Despite these shortcomings, expected deliverables were still fulfilled and we have produced a tool that provides a user-friendly interface for HMD tutorial content generation.

Due to the many factors including the hardware limitations as well as software accessibility with respect to file transfer- available on the Hololens it was not possible to create a absolute true 3D heatmap of rooms or regions

5. Management Summary

5.1. Deliverables

Minimum

- Working demo of tutorial editor
    - Speech-to-text
- Generation of 2D heatmap of gaze

Expected

- Working demo of tutorial editor
    - Speech-to-text
    - Image capture
- Generation of 2D ~~and 3D~~ heatmap of of gaze
- Ability to view heatmap data in using a graphical aid

Maximum

- Working demo of tutorial editor
    - Speech-to-text
    - Image capture
    - **JSON files generated on HoloLens**
    - ~~Marker creation~~
    - ~~Expertise levels~~
- ~~Using 3D and 2D gaze tracking heatmaps to optimize processes~~
- Testing ~~with medical procedure~~ with simple step-based procedures, and Dr. Molina trying a demo.

Note: Strikethrough = removed (reasons detailed in previous section), bold = additional details of original deliverables

5.2. Responsibilities

Allan developed the tutorial generation application, and Prateek was responsible for the eye gaze tracking module with heatmap generation. Both worked on integrating these tools with the original software.

5.3. Next Steps

- Part of our team will continue working on the project in order to enhance the analytics capabilities with regards to the eye gaze tracking capabilities and analysis and expand it to be useful in a 3D work environment.
- Furthermore, the goal of integrating our teams modules with our sister project team that is working on specialized training of particular procedures is also an area that will be worked on.
- Future implementations should add the ability to create virtual markers via the HoloLens tutorial generation app.
- The camera instance in the currently used PhotoCapture API should be integrated with video streaming to allow for live trainee evaluation.

6. Acknowledgements

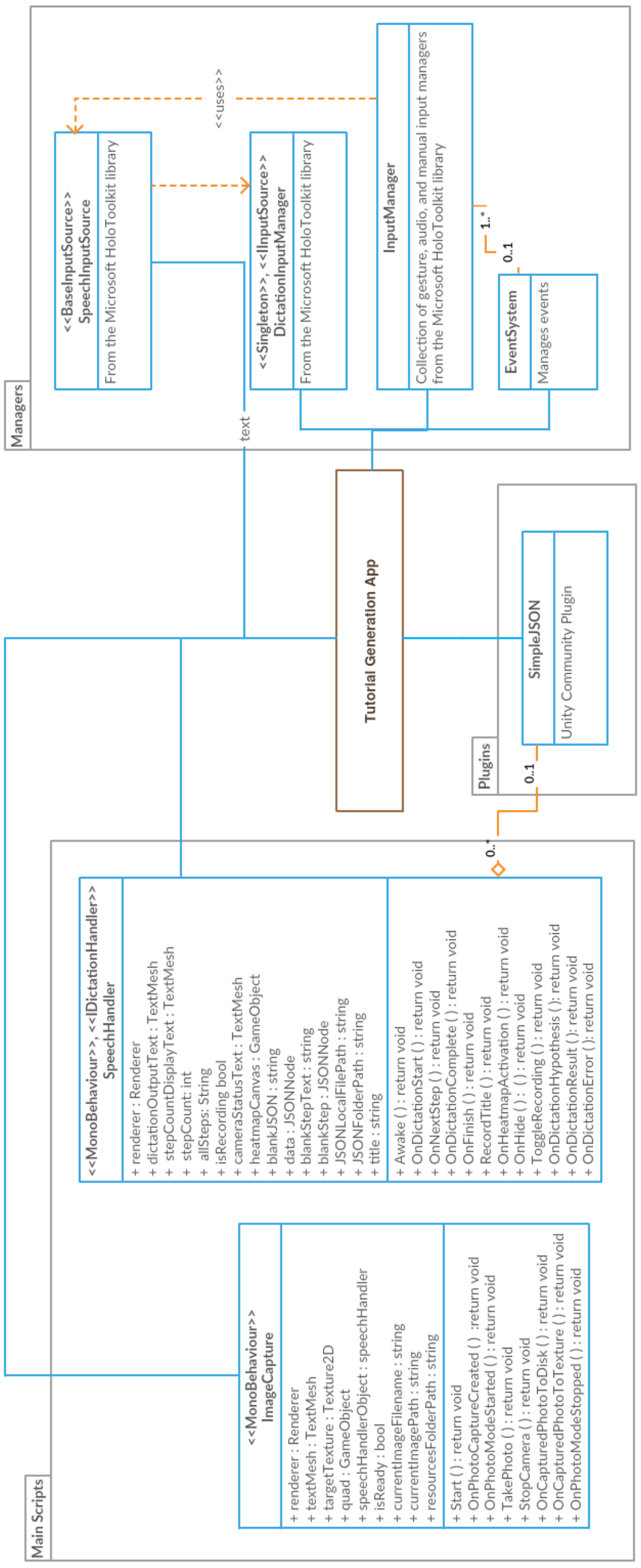
We would like to thank all of our mentors for their guidance and assistance throughout this semester. Dr. Huang and Dr. Kazanzides generously provided us with the hardware we needed to develop on. We would like to extend special thanks to Dr. Molina for providing professional feedback and encouragement on our project and for commuting from the JHMI campus throughout the semester. Finally, we would like to thank Ehsan for being a wonderful TA for the course in addition for keeping our group on track through the semester as our mentor.

Citations

1. Azimi, Ehsan, et al. Evaluation of Optical See-Through Head-Mounted Displays in Training for Critical Care and Trauma. 2018.
2. Wang, S., Parsons, M., Stone-McLean, J., Rogers, P., Boyd, S., Hoover, K., … Smith, A. (2017). Augmented Reality as a Telemedicine Platform for Remote Procedural Training. Sensors (Basel, Switzerland), 17(10), 2294. https://doi.org/10.3390/s17102294
3. Carbone M., Freschi C., Mascioli S., Ferrari V., Ferrari M. A Wearable Augmented Reality Platform for Telemedicine; Proceedings of the International Conference on Virtual, Augmented and Mixed Reality; Toronto, ON, Canada. 17–22 July 2016; pp. 92–100.
4. Cui N., Kharel P., Gruev V. Augmented reality with Microsoft HoloLens Holograms for Near Infrared Fluorescence Based Image Guided Surgery. Proc. SPIE. 2017;10049 doi: 10.1117/12.2251625.
5. Kato, H., & Billinghurst, M. (1999). Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. In Proceedings of the 2Nd IEEE and ACM International Workshop on Augmented Reality (p. 85--). Washington, DC, USA: IEEE Computer Society. Retrieved from http://dl.acm.org/citation.cfm?id=857202.858134
6. Birt, J., Cowling, M., & Moore, E. (2015). Augmenting distance education skills development in paramedic science through mixed media visualisation.
7. Armstrong, D. G., Rankin, T. M., Giovinco, N. A., Mills, J. L., & Matsuoka, Y. (2014). A heads-up display for diabetic limb salvage surgery: a view through the google looking glass. Journal of Diabetes Science and Technology, 8(5), 951–6. https://doi.org/10.1177/1932296814535561
8. Tai, B. L., Rooney, D., Stephenson, F., Liao, P.-S., Sagher, O., Shih, A. J., & Savastano, L. E. (2015). Development of a 3D-printed external ventricular drain placement simulator: technical note. Journal of Neurosurgery, 123(4), 1070–6. https://doi.org/10.3171/2014.12.JNS141867
9. Atkins, M. S., Tien, G., Khan, R. S. A., Meneghetti, A., & Zheng, B. (2013). What do surgeons see: capturing and synchronizing eye gaze for surgery applications. Surgical Innovation, 20(3), 241–8. https://doi.org/10.1177/1553350612449075
10. Kersten-Oertel, M., Jannin, P., & Collins, D. L. (2012). DVV: a taxonomy for mixed reality visualization in image guided surgery. IEEE Transactions on Visualization and Computer Graphics, 18(2), 332–52. https://doi.org/10.1109/TVCG.2011.50
11. Eck, U., Stefan, P., Laga, H., Sandor, C., Fallavollita, P., & Navab, N. (2016). Exploring Visuo-Haptic Augmented Reality User Interfaces for Stereo-Tactic Neurosurgery Planning. In G. Zheng, H. Liao, P. Jannin, P. Cattin, & S.-L. Lee (Eds.), Medical Imaging and Augmented Reality (pp. 208–220). Cham: Springer International Publishing.

Appendix

A. UML Diagram for the Tutorial Generation Tool



Managers
<<BaseInputSource>>
SpeechInputSource
From the Microsoft HoloToolkit library
<<uses>>
<<Singleton>>, <<IInputSource>>
DictationInputManager
From the Microsoft HoloToolkit library
InputManager
Collection of gesture, audio, and manual input managers from the Microsoft HoloToolkit library
1..*
0..1
EventSystem
Manages events
text
Tutorial Generation App
Plugins
SimpleJSON
Unity Community Plugin
0..1
0..*
Main Scripts
<<MonoBehaviour>>, <<IDictationHandler>>
SpeechHandler
+ renderer : Renderer
+ dictationOutputText : TextMesh
+ stepCountDisplayText : TextMesh
+ stepCount: int
+ allSteps: String
+ isRecording bool
+ cameraStatusText : TextMesh
+ heatmapCanvas : GameObject
+ blankJSON : string
+ data : JSONNode
+ blankStepText : string
+ blankStep : JSONNode
+ JSONLocalFilePath : string
+ JSONFolderPath : string
+ title : string
+ Awake ( ) : return void
+ OnDictationStart ( ) : return void
+ OnNextStep ( ) : return void
+ OnDictationComplete ( ) : return void
+ OnFinish ( ) : return void
+ RecordTitle ( ) : return void
+ OnHeatmapActivation ( ) : return void
+ OnHide ( ) : ( ) : return void
+ ToggleRecording ( ) : return void
+ OnDictationHypothesis ( ): return void
+ OnDictationResult ( ): return void
+ OnDictationError ( ): return void
<<MonoBehaviour>>
ImageCapture
+ renderer : Renderer
+ textMesh : TextMesh
+ targetTexture : Texture2D
+ quad : GameObject
+ speechHandlerObject : speechHandler
+ isReady : bool
+ currentImageFilename : string
+ currentImagePath : string
+ resourcesFolderPath : string
+ Start ( ) : return void
+ OnPhotoCaptureCreated ( ) :return void
+ OnPhotoModeStarted ( ) : return void
+ TakePhoto ( ) : return void
+ StopCamera ( ) : return void
+ OnCapturedPhotoToDisk ( ) : return void
+ OnCapturedPhotoToTexture ( ) : return void
+ OnPhotoModeStopped ( ) : return void