Spring Term CS Capstone Design Document

May 20, 2020

Augmented Reality for Real-Time Strain Visualization

Prepared by

Team ARH

An-phong Luu Nguyen

Conner Rhea

Fransisco Javier Hernandez

Abstract

This document describes the design components of an application being developed for the OSU Civil Engineering Project under the supervision of Professor Mike Bailey and Dr. Chris Higgins. It contains an overview discussing the scope, purpose, audience, definitions and context of the project. Additionally it contains the core design components of the project including in strain gauge functionality and data transmission, visual strain gauge tracking, Visual heat map projections, swapping system states and handling strain gauge synchronization. Each of these sections will be explained in depth including potential concerns and the suggested approach the team will take for dealing with each major design component. The design of the system was heavily changed after meeting Dr. Chris Higgins in person, and as such we have appended section 6 in our table of contents to match the client's demands as our base requirements. The previous functionality we have kept in as the client did not deny what we proposed, but insisted on the features we added as the main priority.

Contents

1 OVERVIEW

1.1 Scope

The goal of this project is to develop a system that would allow a user to view concrete strain test results in realtime through a Microsoft Hololens unit. This system will then allow the user to have a more detailed view on tests, allowing for early test modifications or conclusions. The current scope of the project extends to visualizing only one form of concrete testing in which a concrete pillar of set dimensions is embedded with 4 strain gauges at set positions. The project also extends to development on the Hololens unit and an external system. The strain gauges, testing procedure, and test storage device should remain mostly untouched by the project.

1.2 Purpose

The project's main goal is to develop the system so that it can only view test results. This will allow users to visually see changes to test results during tests and make changes as necessary. This is the primary goal and minimum condition for the project's success. However, because the final project will have room for future features such as changing test specifications, the team will design the project in a way that allows for such changes.

1.3 Intended Audience

This document will provide information on the project to the students working on the project, project stakeholders, project sponsors, and CS 461 professors. For the students, this document will provide an outline of the project to more clearly plan and execute project development. The primary stakeholder of the project is the client Dr. Christopher Higgins from the Civil engineering department at Oregon State university. Reading this document will provide the client and project sponsor an overview of the teams plan for the final project. This document will provide professors an overview of the team's project and they will determine if this design plan is acceptable.

2 DEFINITIONS

- **Strain gauge:** A module containing wires of a set resistance. Strain causes wire resistance to change, allowing for the strain felt by the gauge to be recorded
- **Packet:** A set of data sent over a network.
- **User Interface (UI):** The visual components of a system that a user interacts with to modify the state of the system.
- **Augmented Reality (AR):** The concept of overlaying one's view of reality with digital content or images. Mostly just UI elements [1].
- **Mixed Reality (MR):** Similar to AR, but instead of just appearing as a UI element, digital objects are anchored to the user's environment. Objects are tracked and move according to the user's own movements rather than just superimposed over a fixed position in the user's view [1].
- **Hololens 1/2:** Headsets that allow for users to use Augmented or Mixed reality applications.

3 PROJECT CONTEXT

3.1 Hardware

The hardware used in the project will be the current testing environment along with a Microsoft Hololens and an external system.

The testing environment is fixed for the scope of the project. There can be no modifications to these components of the system:

- Strain Gauges
- Strain Recording System

Parts of the system being developed include the Hololens and an arbitrary external computer system. Currently, the team is unsure if a Hololens 1 or 2 will be used for the project. The external system's specifications are mostly arbitrary as it is simply a device used to transmit data between the Hololens and strain gauge transmitters.

The following are the specifications for both the Hololens 1 and 2 [a2, a3, a4]:

	Hololens 1	Hololens 2
Display Resolution (per eye):	1280x720	2048x1080
Field of View:	34°	52°
Processor:	Intel 32-bit (1GHz)	Qualcomm Snapdragon 850 (3GHz)
Camera:	2.4 MP, HD video	8 MP, 1080 video
Battery Life:	2-3 Hours	2-3 Hours
Memory:	2 GB	4 GB
Storage:	64 GB	64 GB
WiFi:	Wi-Fi 802.11ac	Wi-Fi 802.11ac
Bluetooth:	Bluetooth 4.1	Bluetooth 5

3.2 Software

- Unity: The development platform used to create the project. Will be used to generate the UI as well as digital strain gauges.
- Mixed Reality Toolkit (MRTK): Library used to develop the spatial interactions between the digital strain gauges and the user's environment. Used alongside Unity.

4 DESIGN DESCRIPTION

4.1 Design Stakeholders

4.1.1 Project Partner (Mike Bailey)

This project's sponsor/partner, Professor Mike Bailey at Oregon State University, is working with An-phong Luu Nguyen, Fransisco Hernandez, and Connor Rhea to create a system to aid Professor Chris Higgins at Oregon State University's concrete testing facility. Professor Bailey is our main line of communication to the intended user of the product.

4.1.2 Client (Christopher Higgins)

Dr. Christopher Higgins is part of the Civil Engineering department and OSU. He will be providing information on the testing site and the project is designed for his use.

4.2 Design views

4.2.1 Users

Users of this project will be able to view the testing results of concrete strain testing in real time with the use of a Hololens headset. They will be expecting the project to change little about how they conduct tests normally. The project is to act as an addition to the test, providing additional features while changing nothing. Future implementations of the project could possibly allow the user more control of the system to customize their experience, but the scope of the project does not include this.

4.3 Design Rationale

We chose to develop using the Unity engine along with the MRTK for Unity supported by Unity and Microsoft. This was mainly due to the limited options available for Hololens development. Using Unity and the MRTK for Unity is generally viewed as the quickest and most convenient method to learn programming for a mixed environment.

5 APPROACH

5.1 Inclusion of Strain Gauges

The current scope of the project does not include adding additional strain gauges. As of now, the test parameters that the system should record for is fixed, and so the number of strain gauges should remain static. However, each individual strain gauge needs to be uniquely identified by the Hololens unit so that the system can accurately map test results in real time.

5*.1.1 Concerns*

This component is primarily concerned with keeping track of strain gauges in some form. Although the number of strain gauges should remain static in the scope of this project, future updates to the project may need more gauges to be added. Another concern is that the team has not been able to physically see the hardware available at the testing site. Without knowing the exact hardware layout of the strain gauge to storage system, it is more difficult to plan the connection between the Hololens and storage system. In addition, future implementations of this project may require that more than the initial 4 strain gauges be measured by the system. So, this component should be designed with future implementations in mind.

5*.1.2 Approach*

Each packet of information sent from a strain gauge to the Hololens will contain the strain gauge's value as well as a unique identifier enumerated 0, 1, 2… Doing this will allow the Hololens to quickly identify to which strain gauge that value belongs to.

Identifying each strain gauge this way also provides the possibility for the system to interface with additional strain gauges. Although the system will be hardcoded to detect 4 strain gauges, additional strain could possibly be added. This would be accomplished through two steps:

- **Set the Hololens to accept packets with the identifier value N.** This means implementing a "verifier" component to the system. In the data pipeline, this would appear after the data is sent from the receiver, but before the data is converted into digital images. The verifier works by first examining the ID component of the packet and cross referencing it with the list of accepted values. For the current scope of the project, this would be limited to the first 4 indices. Invalid values will be ignored and not sent through the remaining pipeline. In the case of a match, the value is passed further on in the pipeline. This component is necessary so that invalid values don't create unexpected results later on in the pipeline.
- **Modify the transmitter system to send packets with the identifier value N appended.** Because this project does not extend to modifying strain gauges or the current recording device, this refers to modifying the external server that temporarily holds strain gauge values. To implement this change, the external server will be set to send additional data along with each strain gauge value.

One future modification to the project is adding additional strain gauges along with the initial 8. This modification does not fall under the scope of the current project, so the description of its design will not be included in this document. However, the system of strain gauge enumeration has been designed with this change in mind. Additional strain gauges may be included in the system and enumerated by providing id of value greater than the 0-3 of the current set of strain gauges. Although the systems to add additional strain gauges would have to be implemented to achieve this feature, the current design allows for this modification with little change to the original system.

5.2 Visual Strain Gauge Tracking

It is necessary for the system to be able to display digital representations of strain gauges for the user. These representations must be accurately projected over the user's vision of each strain gauge. In addition, the digital images must be tracked so that they remain in the correct orientation as the user moves.

5.2.1 Concerns

The primary concern of this component is how the digital representations of each strain gauge will be tracked in the user's vision. Due to their cost, physical tracking libraries will not be used with the project, so this design must find a way to track strain gauge positions without the use of the Hololens camera to track strain gauges. In addition, because future implementations of the project may include customized tests with more than 4 strain gauges, the system used for physical tracking should accommodate this somehow.

5.2.2 Approach

Because in the current scope of the project there are only 4 strain gauges held in fixed positions, it is possible to hardcode the position of each strain gauge within the system. Hardcoding strain gauge positions will be done by performing the following:

1. **Mapping the testing area by physically scanning surroundings with the Hololens.** This involves the team physically going to the testing site with the Hololens. A brief walkthrough of the area will allow the Hololens to create a digital mapping of the site.

2. **Physically measure where in this mapped space each strain gauge will be located.** This involves physically measuring the location each strain gauge would sit within the physical environment.

3. **Record this data into a configuration file.** Data on the x,y,z coordinates of each strain gauge will be held in a file that will be provided to the Hololens. The format of the configuration file would be as follows:

```
0    xp    yp    zp    xr    yr    zr

1    xp    yp    zp    xr    yr    zr

2    xp    yp    zp    xr    yr    zr

3    xp    yp    zp    xr    yr    zr
```

   The number at the start of each row refers to the strain gauge id. Following this are two triplets of x,y,z coordinates. Coordinates ending in "p" refer to the physical location of the strain gauge. Coordinates ending in "r" form a vector used to determine the strain gauge rotation.

Mapping the test site will let the Hololens create a 3D representation of the environment. The hololens is then able to create its own x,y,z as well as its look-at coordinates to determine its exact location in the environment. Based on that data as well as the hardcoded strain gauge locations, the system can render each strain gauge at its correct location. Tracking of user movement is also solved through this method. There will be no need to track the user's motion through the testing environment. As each system tick generates a new physical coordinate set, the system should be able to know its position and perspective at all times.

Because in the scope of the project the system only needs to map one set of testing conditions, it should be possible to hardcode strain gauge locations within the component that places the gauges instead of reading such parameters from a file. However, this would hamper the projects ability to map other testing conditions if needed in future implementations. Simply reading from a configuration file allows for more flexibility, as the system would arbitrarily read from the file to create the environment. Then, additional features to the system could allow the user to create their own configurations with different testing parameters.

5.3 Visual Heat Maps

One of the core functions of the system is to generate a heatmap given data from a strain gauge over the predetermined locations. As the strain gauges are added and synchronized to a heatmap object the system will be able to display the heat map on the HoloLens at the hard-coded locations. The heatmaps will be updated constantly to reflect test data.

5.3.1 Concerns

A main concern for the creation of heatmaps is whether or not creating a larger overall heatmap in reference to the other strain gauges is possible in terms of performance. Our current frame of reference for how long it will take the HoloLens to do these calculations dynamically whilst having a low latency is unknown.

5.3.2 Approach

In the way of implementing, each object inside of the environment can be scripted. The MRTK gives a multitude of interactable triggers for these objects. Such as: tappable buttons, gestures, and voice commands.

Here will be the order of events for creation of a heat map object:

1. Strain gauge is added to the system(another requirement, assumed to be successful).
2. An object will be generated(hard-coded or potentially dynamic) and inherit the heat map function along with it.

   Order of events once heat map mode is engaged:

3. Projection housing will appear on the lens.
4. Data will be read continuously and color to reflect the data will be chosen.
5. Using the MRTK shader or Unity3D shader, the heat map will be generated in the object.
6. Repeat step 3 until mode is changed.

We will begin with experimenting with heat map creation via using the emulator available to use in Unity[3] to generate an environment with an interactable button. Upon clicking the button, a preset of data will be used to generate a heatmap.

The preset of data will already have the color's for a heatmap determined to compare it to the emulation's data. An object will also be made in the emulator that will be used as a 'physical' object to embed the heat map onto the object.

5.4 Swapping System States

The swapping of system states will be handled by creating an interactable object on the HoloLens screen which will be used to toggle between states.

The creation of this feature will be fairly simple as each function can have their respective object's contain a boolean variable which is checked during their runtime.

1. The following will be the process to swap from heat map mode to idle:
2. Tap a button labeled idle. May be located in a menu.
3. The bool value of mode_heat_map will be turned to false.

Objects will no longer be rendered. Data from strain gauges will no longer be processed. Though the data sent to the HoloLens will still be received, it will be immediately discarded rather than worked on.

5.4.1 Concerns

The main concern for the swapping of modes will be the management of strain gauge data. The purpose of swapping states is to prevent the constant updating of heatmaps and the projection of the objects on the screen when going to idle mode.

5.4.2 Approach

Beginning with experimenting with this using the same emulator referenced earlier in this paper[3]. Once heat maps have been implemented successfully. A new interactable will be implemented using the MRTK in Unity. The button will be scripted to stop rendering the heat map objects and to tell the software to stop processing the data being received and simply discard it.

Once Synchronization mode is implemented, we will work with the team members in charge of it to add the proper values and then begin testing swapping from each possible state.

1. Idle -> Synchronization

2. Idle -> Heatmap
3. Idle -> Idle(no change)
4. Synchronization -> Idle
5. Synchronization -> Heat map
6. Synchronization -> Synchronization(no change)
7. Heat map -> Idle
8. Heat map -> Synchronization
9. Heat map -> Heat map(no change)

5.5 Strain Gauge Data Transmission

In order to properly animate the visual strain gauges in mixed reality as well as generate heat maps to track their movement we will need to be able to transmit data from the strain gauges to the Microsoft HoloLens. The HoloLens is equipped with wireless functionality as well as bluetooth (although its bluetooth support is rather lackluster out of the box) and will need to be able to receive the data recorded by the civil engineering team's hardware setup [5].

5.5.1 Concerns

When it comes to reading in the data wirelessly to the HoloLens, Professor Mike Bailey has described the Civil Engineering teams setup to record the data as archaic. The Machines they use may not be able to send the information to the hololens in packets or a viable alternative for wireless communication. The Team may need to bridge the gap or create software to read the data in first on a laptop or other mobile computer hardware such as a raspberry pi and then wirelessly transfer it to the HoloLens from there using basic networking tools such as tcp/udp [6].

5.5.2 Approach

In order to implement the feature successfully, the team will first need to ascertain the hardware we will be using to pass the data to the HoloLens, however the implementation will remain similar. The data from the strain gauges will be read by the Civil Engineering computer and then be transferred to the HoloLens where it will be used for the heat map and visual strain gauge mixed reality projection tasks.

Here is the projected order of events for reading in the strain gauge data:

1. Dr. Higgins and his team will perform the strain test, and the gauge data will be recorded by their tools.
2. The data will either be directly transmitted to the HoloLens through a wireless receiver if the tools they use are able, or the data will be transferred via some means (usb, bluetooth, etc…) to a wireless enabled laptop or mobile computer.
3. The data will be sent over the network to the HoloLens.
4. The HoloLens will save the data locally in order to use it for the Heat Map and Visual Tracking.
5. Once the tests are completed, the data will either be saved into a file for future use, or be deleted from the HoloLens to save space depending on the wishes of Dr. Higgin's Team.

Once the team has set up a system for communicating the information from the strain gauges to the HoloLens we will make use of the packet system as explained within *AN1.2* to transfer packets containing information from the strain gauges with an identifier for the strain gauge in question.

5.6 Synchronization Mode

Synchronization mode is an important system state for the project, allowing for strain gauges to be added or removed depending on the individual test being run as well as providing a hands-on method through mixed reality to adjust the tests. Through the use of the HoloLens' gesture technology users will be able to physically manipulate the strain gauges within real time through a graphical representation in Mixed Reality [7]. The strain gauge representation will also present the data received from the strain gauges visually, allowing users to see the readout as it corresponds to the actual gauge that produced it and its location.

5.6.1 Concerns

According to Mike Bailey the initial form of the project will have the number of strain gauges remain static and thus adding and removing the gauges will not be a primary concern for Dr. Higgins in the initial testing. However in future iterations of the software, the team aims to provide the means to alter tests when needed such as accounting for additional strain gauges for different sized slabs of concrete.

Additionally, properly mapping the locations of the strain gauges will require the data from the test to be available before the test is completed, which may require manual entry or additional wireless transmission during the process, which may require the user to wait or manually enter the data to make use of the mode properly. Depending on the status of the Engineering Team's Hardware, this issue may be negligible or a significant time loss.

5.6.2 Approach

Synchronization mode will be a separate mode to be entered in order to prepare for a strain trial. Once activated from idle mode or heat map mde, the user will see a graphical representation of the current strain gauges as well as any readout received from wireless transmissions.

The user will be able to use the gesture technology of the HoloLens to interact with these gauges directly as well as have the ability to perform a gesture to add or remove strain gauges from the current trial. Adding and removing strain gauges should either create or remove a mixed reality graphic of the gauge that the user can then interact with, additionally synchronization mode should be able to read in the coordinates from the wireless readout to position the gauges in their proper locations ideally.

Once Heat Map Mode and Idle Mode are properly implemented, the team will work together to connect these modes to create the general user framework and enable the ability to switch between states.

Here is the projected order of events for using synchronization mode:

1. From either Heat Map Mode or Idle Mode, the user swaps to Synchronization Mode through the use of a command or gesture.
2. The User is then presented with on screen commands and gestures to adjust the number of strain gauges for the current trial, as well as a graphical representation of the gauges in mixed reality.
    a. If the user chooses to add a strain gauge, a new strain gauge object will appear in the mixed reality projection. If unused strain gauge data exists from the wireless readout, the strain gauge should automatically snap to its proper location and begin displaying. If not, the user can manually move the gauge (represented by a draggable sphere which will change color to represent strain movement) or enter the data themselves.
    b. If the user chooses to remove a strain gauge, the user will be prompted to select a strain gauge to delete using the gesture technology in mixed reality. A deleted strain gauge will no longer be seen within mixed reality projections, and its data will not be represented, however the data will remain on the system if the gauge is re-added.
3. Once the user has finished adjusting the strain gauges to their liking, the data available for the current strain gauges should be accessible, either at a glance or through the use of a gesture. If the number of gauges has been changed, the projection will be updated with the new information.

4. Once the user has finished adjusting the gauges the system can be swapped to either Heat Map Mode or Idle Mode.

6 ALPHA 0.0.1 SYSTEMS BREAKDOWN

6.1 Experiencing the Project

The design of the system was heavily changed after meeting Dr. Chris Higgins in person, and as such we have appended this 6.0 section in our table of contents to match what we have implemented currently whilst we wait on processable data to begin implementing what our original core functionalities would be. The current hypothesis is that we will need the data to be in a raw form that will let us calculate the displacement ourselves rather than the test file they have given us which has displacement already calculated.

6.2 Implementations

The current is on a private github page as we were unsure if the information we would eventually be using is private research or not. We will ask if anything provided to us is as we did not enquire about it beforehand.

6.2.1 configReader

This system manages the placement of strain gauges in the digital environment. It has two accessible functionalities: creating strain gauges and removing them. In the alpha version, the config reader reads triplets of values from a configuration file. It interprets these numbers as coordinate points then creates and places virtual strain gauges into the environment. In the current alpha version, this is done in two ways. In the bottom left corner of the screen, there is a text box as shown in figure 1. The user is able to input the filepath of the configuration file they would like shown in the environment. Pressing the start button loads the configuration file. If left blank, the system defaults to the filepath “Assets/config.ini”. On the system starting, the configReader also automatically loads this default setting. In the final product, these fields will need to be accessible using AR interactions. The configReader system is built to be independent of this. Future UI implementations will simply need to access the configReaders functionality to do the same actions.

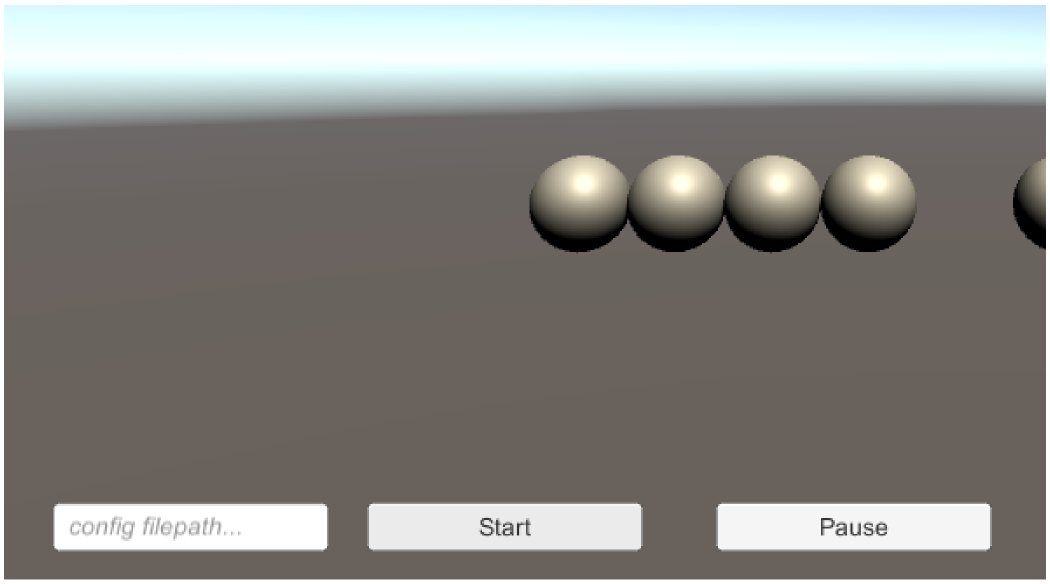


Figure 1: Screenshot of the alpha showcasing the config filepath text box and start button.

The config reader has to follow a procedure to spawn in new gauges properly. It starts by reading in values from the config file, creating a new object placed at the origin (0,0,0), moving that object to the correct position, and then providing the empty object with a strain gauge script. The development process of a strain gauge is described in figure 2.

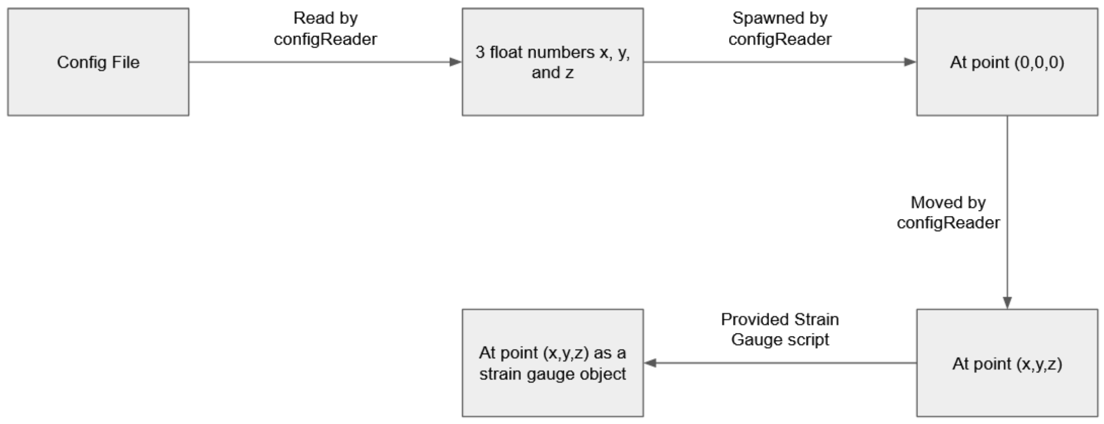


Figure 2: Flow chart of how strain gauge coordinates are converted into a digital strain gauge.

Besides the spawning functionality, the config reader is also in charge of deleting excess strain gauges. This is necessary because if the user decides to use a different config file, old strain gauges need to be removed before new ones are created.

This component of the alpha addresses design components 5.1 Inclusion of Strain Gauges and 5.2 Visual Strain Gauge Tracking. This component determined that strain gauges would be tracked through a numerical system starting from 0 and incrementally increasing, solving issues with 5.1. Physical coordinates are now being tracked by the system through the config files, addressing what 5.2 was asking.

Currently, the API of the configReader only includes these functions:

- readConfig(string)
- readConfig()
- clearGauges

6.2.2 CSVreaderMock

This is the temporary component used to interface the team's mock data with the alpha. Its job is to occasionally read through the .csv file containing strain values. After reading values, it goes through each existing strain gauge and tells them to displace and recolor based on their respective values. It should be noted that this system does not physically move strain gauges or modify their colors. All it does is access the strain gauge script and call the displace() and color() functions.

The current csvreader for the alpha is a temporary version. The final project will be using a modified version in order to adapt to the .csv format provided by the test site. Currently, the team is trying to get different formats of the .csv file that can be more easily parsed and understood by the system.

The csvreader also includes pause functionality. As seen in figure 1, there is a pause button that when pressed, causes the csvreader to no longer read and update values.

Currently, this system is set to poll its .csv file 50% of the time. This is done to avoid I/O exceptions where the system in charge of updating the .csv is accessing the file while the csvreader attempts to read it.The 50% benchmark is to allow the other system share half of the filetime with csvreader. The current system is set to stop polling the .csv file 50 out of 100 milliseconds, meaning it only polls during the latter 50 milliseconds of each 100 milliseconds cycle. The system is also built to handle issues where the reading time of .csv reader accidentally intersects with the system updating the .csv. Everyime there is a conflict, csvreader will speed up its read window by 1 millisecond. Over time, this will push the two conflicting windows into a more unobstructed pattern. Figure 3 shows an example of how csvreader cleans its access rhythm with other systems.

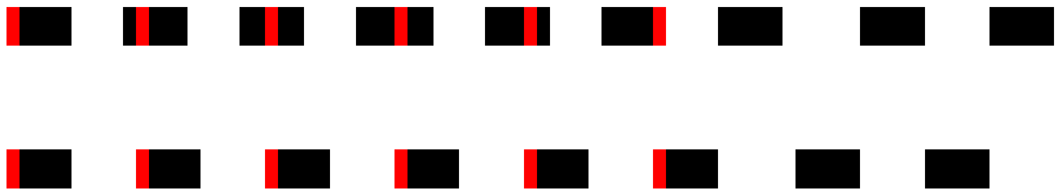


Figure 3: Example of csvreader fixing file access issues with access windows of 5 milliseconds. Red frames show the first millisecond of conflict. Following a conflict, csvreader only waits 4 milliseconds before attempting an access.This continues until no more conflicts occur, where it goes back to 5 second rest periods.

The csvReader doesn't include any public functions or an API. When put into the digital environment, it works on its own and can't be modified.

6.2.3 displacement

We have been asked to focus on displacement rather than heatmaps and slip strain from the client. The main functionality of displacement is to calculate the displacement of strain gauges in the experiment. It would function similarly to Heatmaps except it would model the actual changes of the model in a physical sense based on the displacement done. In our current alpha version there is no way to do this without proper output examples from the software. As I require to meet with our sponsor and most likely the client to find out what equations or formulas I use to calculate the values desired.

Displacement will be a script that is executed while the HoloLens is running with a touch of a button. Initializing the actual logic that will affect the models placed by the HoloLens itself. It will update the gauges and their displacement while reading from the csvReader we will be using.

Gauges are essentially just classes that can call an update function on themselves should they be passed new data relating to said gauge

Once the logic for displacement is correct. We can then build the functionality of combining *displacement* and *configReader* for consistent updates to the models that will be presented in the HoloLens.

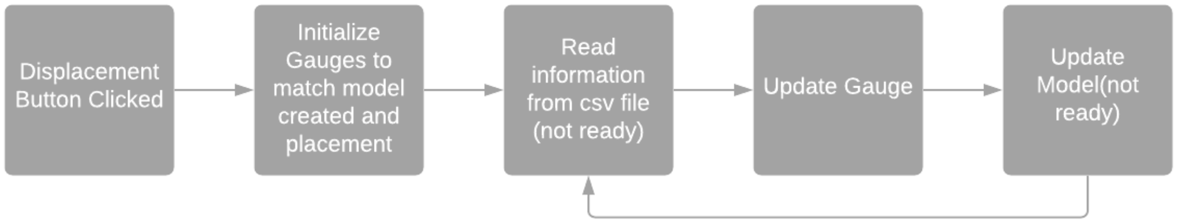

Displacement Button Clicked
Initialize Gauges to match model created and placement
Read information from csv file (not ready)
Update Gauge
Update Model(not ready)

Figure 4: Flowchart of how *displacement* will function once activated

6.2.4 gaugeData

This script is in charge of managing both all and individual strain gauges. When added to an object in the digital environment, it holds positional data meaning where the object was spawned, and where it currently is. Along with this, it includes functionality to move and recolor the object that it represents.

This script is not intended to be used on its own, but rather created through the configReader's readconfig() function. While it can be used alone, the configReader is intended to spawn the exact number of strain gauges that the system will be using. Adding additional gauges may cause strange behavior in the csvReader as it is expecting as many gauges as the configReader creates.

This script includes a static variable objCount. This number is shared across all strain gauges and is used to index each one. This is how the system accomplishes keeping track of each individual strain gauge. It should also be noted that the configReader will call gaugeData's resetCount() function each time it reads from a new config file. This is to reset the strain gauge index, putting it back to 0.

It should also be noted that the current gaugeData doesn't do any calculations on how much it should be displaced. At present, the displace function will just move the gauge however many units are passed to it. Later iterations of gaugeData will calculate for exact displacement as noted in the above section.

6.3 Research Performed

6.3.1 MRTK & HoloLens Emulator

The Mixed Reality Toolkit(MRTK) has had some confusing results across the team as the steps given to configure it feels scattered. Redirecting to multiple pages in which it is not clear what exactly a user needs to use it. In addition it is not clear if some plugins are required or not to run the emulator.

The emulator relies upon the version of windows one is using so some team members were unable to run the emulator due to having Windows Home. We have since remedied this by being directed to getting a version of Windows through the Azure Services. It is unclear if our machines cannot run the emulator effectively, or if our configuration is making the performance take a heavy hit.

Otherwise once the MRTK is added, creating interactables to add scripts is streamlined and fast in terms of editing how they work.

7 BETA 0.0.2 SYSTEMS BREAKDOWN

7.1 Experiencing the Project

As we described our beta video, professor Bailey has told us to sideline our primary concerns and create a 1-dimensional solution showcasing displacement in order to showcase it to our client.

Here is our github page, now public: https://github.com/hernafra/AR-for-Realtime-Strain-Visualization

In order to see the code, minus the test data, as An-phong presented in the beta video showcasing the objects being created and reading through a test file that matches the information we have been given you can look at Anphong-branch in the github to see the code.

In order to see the excel reader that connor has been working on, you may look at the GammaRhea-excel-reader branch to see what he has been working on.

As previously mentioned in the Beta video, Fransisco had a medical emergency after meeting with professor Bailey. Thus was not able to contribute to the aforementioned solution created so far.

7.2 Implementation

7.2.1 sceneManager

This script works along with the two new Unity scenes to allow the user to swap between mock data and test results modes. It operates by simply providing a public function that switches the active Unity scene depending on the public string destination.

This script will be used to also swap into the realtime data collection mode. The inclusion of a public string variable allows the Unity programmer to modify the location of the scene swap. In this way, the script is flexible and won't need any variations.

Along with the script, 2 new scenes were added to the project. They contain all the components needed for the project to run in either mock or results mode.

7.2.2 csvReaderTestResults

This is a temporary script that the beta uses to read the example .csv data provided to the team and simulate what those results would look like. It starts by reading the entire .csv file and loading that data into a C# object: a 2d List (declared List<List<float>>). Included are 21 values including the time of the measurement and the east/west slip values for each strain gauge.

The script updates each strain gauge by measuring the internal C# variable "time". The script starts at the top of the list, and at each update interval, compares the time value of the current index to the C# time. Once they match, an update signal is sent to each strain gauge, causing them to displace. Once this check goes through, the script moves to the next line in the data structure and waits for the C# time to advance past the time value in that row.

Currently this script has two major flaws that need to be fixed in future versions. The first is the lack of scaling or balancing between strain gauges. Because no range is recorded for maximum or minimum gauge values, it's possible that upon running the project, the strain gauges will appear stationary and won't change color. In reality, there are changes occurring, just at such minute levels that they appear to not move at all. The lack of balancing or zeroing for gauges creates an issue where the digital representation may appear unbalanced. In the current example data, west gauges appear to be generally lower than east gauges, as shown below.

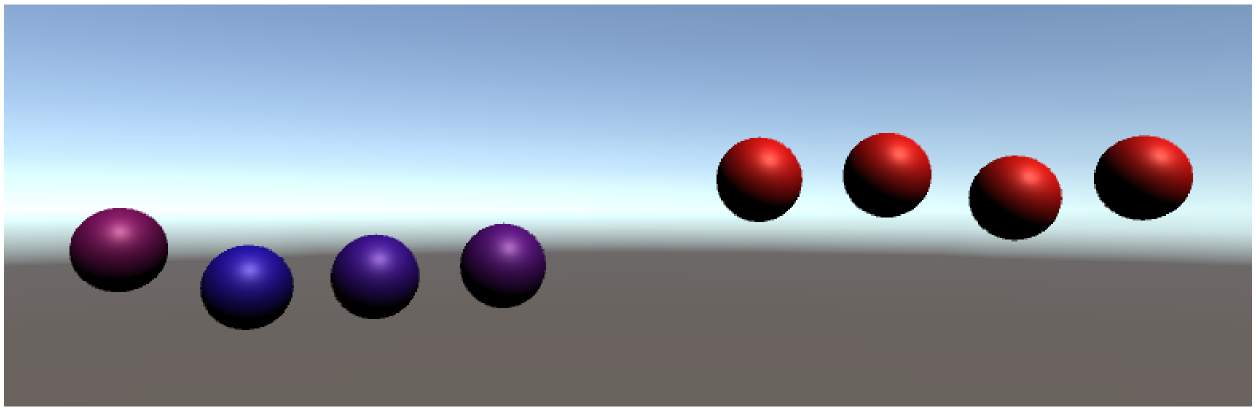


Figure 5: Example of how not zeroing or balancing initial gauge values can create inherent offsets in the data. Although at the start of the test, east gauges are significantly displaced more than west ones.

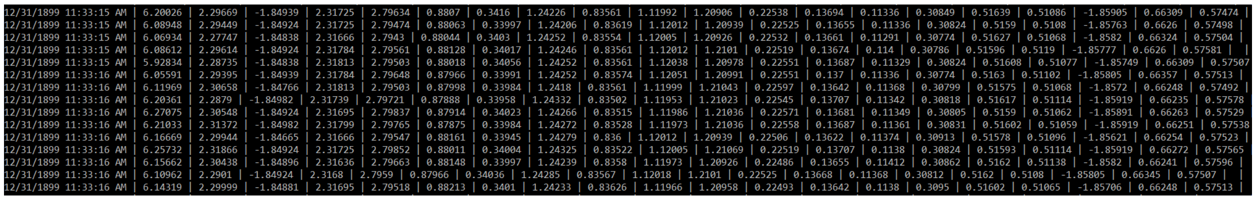
Along with this, the current csvReaderTestResults script loads all data from the .csv into the List array. Only 9 values from each row are used whereas the script loads in all 21 values. This inefficiency will be corrected in future versions to reduce memory strain.

7.2.3 Correction of config.ini to reflect test environment

One minor change from this version to the alpha was correcting the coordinates provided by the default config.ini. These modifications were made to reflect the actual diagram of the test site provided by the client Professor Higgins. Once the team implements full AR support for the system, it is possible that another correction will need to be made to properly scale the gauges. Currently, the conversion from inches to Unity units is 1 in = 0.1 units. This scaling may not actually be correct but the team cannot test this scaling without the Hololens II.

7.2.4 Excel Reader and Data Set

After meeting with Professor Bailey the team implemented a C# solution for reading the excel files generated by the DASYLab software used by Professor Higgins to model their data. This software makes use of a NuGet package to read in Excel Files and turn them into C# data sets which can then be manipulated into structs for calculating displacement for modeling the change over time. The system still needs the ability to check the minimum and maximum values for each component to be able to correctly model the data. Once connected to the HoloLens 2 directly the team will be able to directly read the data from the file generated and begin performing live tests at the Tsunami Wave Basin.



```
12/31/1899 11:33:15 AM | 6.20026 | 2.29669 | -1.84939 | 2.31725 | 2.79634 | 0.8807 | 0.3416 | 1.24226 | 0.83561 | 1.11992 | 1.20906 | 0.22538 | 0.13694 | 0.11336 | 0.30849 | 0.51639 | 0.51086 | -1.85905 | 0.66309 | 0.57474 |
12/31/1899 11:33:15 AM | 6.08948 | 2.29449 | -1.84924 | 2.31725 | 2.79474 | 0.88063 | 0.33997 | 1.24206 | 0.83619 | 1.12012 | 1.20939 | 0.22525 | 0.13655 | 0.11336 | 0.30824 | 0.5159 | 0.5108 | -1.85763 | 0.6626 | 0.57498 |
12/31/1899 11:33:15 AM | 6.06934 | 2.27747 | -1.84838 | 2.31666 | 2.7943 | 0.88044 | 0.3403 | 1.24252 | 0.83554 | 1.12005 | 1.20926 | 0.22532 | 0.13661 | 0.11291 | 0.30774 | 0.51627 | 0.51068 | -1.8582 | 0.66324 | 0.57504 |
12/31/1899 11:33:15 AM | 6.08612 | 2.29614 | -1.84924 | 2.31784 | 2.79561 | 0.88128 | 0.34017 | 1.24246 | 0.83561 | 1.12012 | 1.2101 | 0.22519 | 0.13674 | 0.114 | 0.30786 | 0.51596 | 0.5119 | -1.85777 | 0.6626 | 0.57581 |  |
12/31/1899 11:33:15 AM | 5.92834 | 2.28735 | -1.84838 | 2.31813 | 2.79503 | 0.88018 | 0.34056 | 1.24252 | 0.83561 | 1.12038 | 1.20978 | 0.22551 | 0.13687 | 0.11329 | 0.30824 | 0.51608 | 0.51077 | -1.85749 | 0.66309 | 0.57507
12/31/1899 11:33:16 AM | 6.05591 | 2.29395 | -1.84939 | 2.31784 | 2.79648 | 0.87966 | 0.33991 | 1.24252 | 0.83574 | 1.12051 | 1.20991 | 0.22551 | 0.137 | 0.11336 | 0.30774 | 0.5163 | 0.51102 | -1.85805 | 0.66357 | 0.57513 |
12/31/1899 11:33:16 AM | 6.11969 | 2.30658 | -1.84766 | 2.31813 | 2.79503 | 0.87998 | 0.33984 | 1.2418 | 0.83561 | 1.11999 | 1.21043 | 0.22597 | 0.13642 | 0.11368 | 0.30799 | 0.51575 | 0.51068 | -1.8572 | 0.66248 | 0.57492 |
12/31/1899 11:33:16 AM | 6.20361 | 2.2879 | -1.84982 | 2.31739 | 2.79721 | 0.87888 | 0.33958 | 1.24332 | 0.83502 | 1.11953 | 1.21023 | 0.22545 | 0.13707 | 0.11342 | 0.30818 | 0.51617 | 0.51114 | -1.85919 | 0.66235 | 0.57578
12/31/1899 11:33:16 AM | 6.27075 | 2.30548 | -1.84924 | 2.31695 | 2.79837 | 0.87914 | 0.34023 | 1.24266 | 0.83515 | 1.11986 | 1.21036 | 0.22571 | 0.13681 | 0.11349 | 0.30805 | 0.5159 | 0.51062 | -1.85891 | 0.66263 | 0.57529
12/31/1899 11:33:16 AM | 6.21033 | 2.31372 | -1.84982 | 2.31799 | 2.79765 | 0.87875 | 0.33984 | 1.24272 | 0.83528 | 1.11973 | 1.21036 | 0.22558 | 0.13687 | 0.11361 | 0.30831 | 0.51602 | 0.51059 | -1.85919 | 0.66251 | 0.57538
12/31/1899 11:33:16 AM | 6.16669 | 2.29944 | -1.84665 | 2.31666 | 2.79547 | 0.88161 | 0.33945 | 1.24279 | 0.836 | 1.12012 | 1.20939 | 0.22506 | 0.13622 | 0.11374 | 0.30913 | 0.51578 | 0.51096 | -1.85621 | 0.66254 | 0.57523 |
12/31/1899 11:33:16 AM | 6.25732 | 2.31866 | -1.84924 | 2.31725 | 2.79852 | 0.88011 | 0.34004 | 1.24325 | 0.83522 | 1.12005 | 1.21069 | 0.22519 | 0.13707 | 0.1138 | 0.30824 | 0.51593 | 0.51114 | -1.85919 | 0.66272 | 0.57565
12/31/1899 11:33:16 AM | 6.15662 | 2.30438 | -1.84896 | 2.31636 | 2.79663 | 0.88148 | 0.33997 | 1.24239 | 0.8358 | 1.11973 | 1.20926 | 0.22486 | 0.13655 | 0.11412 | 0.30862 | 0.5162 | 0.51138 | -1.8582 | 0.66241 | 0.57596 |
12/31/1899 11:33:16 AM | 6.10962 | 2.2901 | -1.84924 | 2.3168 | 2.7959 | 0.87966 | 0.34036 | 1.24285 | 0.83567 | 1.12018 | 1.2101 | 0.22525 | 0.13668 | 0.11368 | 0.30812 | 0.5162 | 0.5108 | -1.85805 | 0.66345 | 0.57507 |  |
12/31/1899 11:33:16 AM | 6.14319 | 2.29999 | -1.84881 | 2.31695 | 2.79518 | 0.88213 | 0.3401 | 1.24233 | 0.83626 | 1.11966 | 1.20958 | 0.22493 | 0.13642 | 0.1138 | 0.3095 | 0.51602 | 0.51065 | -1.85706 | 0.66248 | 0.57513 |
```

Figure 6: Example of reading the Data from the Excel Sheet provided by Professor Higgins into the .net framework.

8 Release 1.0.0 SYSTEMS BREAKDOWN

8.1 Experiencing the Project

As we described in the Beta video, we have scaled the project back massively to the most simplest form due to the barriers we have faced in development and delays we faced earlier in the project. Given the circumstances of our project, we feel that to stop working at where we are is inappropriate. We aim to have a functioning application that can playback a recorded experiment through either a csv or excel file rather than the live one due to time constraints.

We do now have a ported version of application that works on the HoloLens itself, but other than that this term has been very difficult as the roadblocks we face during development have made productivity very difficult. The largest majority of time spent during this project has been on researching and bug fixing basic functionality.

Additional development time was lost last term as Fransisco had a medical emergency and was left unable to develop for a portion of the term. This term, Connor had a bad reaction to his Covid-19 vaccine. Leaving his Excel Reader left in a partially finished state.

Here is our github, the working project branch is one AnphongBranchFreeze:
https://github.com/hernafra/AR-for-Realtime-Strain-Visualization

A demo video of the application being used is shown using the HoloLens is shown in our zoom recording here:
https://media.oregonstate.edu/media/t/1_crtwqa3e

What is shown currently in video is the array of strain indicators is lined up, to scale, to replicate the location of where the gauges are shown in the experiment specifications given to us. The hue of the indicators change demonstrating how once we can read in the information from the excel/csv file, can be changed.

Since we cannot spatially map currently the best option is to let the user reposition the array in a location that is comfortable for them. An interactable green box controls the indicators positioning by making a pinch gesture on the box while looking through the HoloLens.

8.2 Implementation

8.2.1 ConfigReader.cs removed

Difficulties with the Hololens system made it so the team removed the ConfigReader.cs script from the project. What this component did was read coordinate points from a text file and dynamically create strain gauges in the environment.

This feature was removed from the system because of difficulties finding ways for a user to both access and edit a .txt file that could be read from using the Hololens. With more time, the team would have been able to implement internal systems to edit this, however with current time constraints the feature was removed.

This change mainly affects the system by reducing how dynamic it is. Instead of creating strain gauges based on an input file, the system now has 8 hard-coded strain gauges in its environment. Overhauling this system also took a substantial amount of time, so the team was unable to implement features to bring back the old system.

8.2.2 System Grabbable

Section 5.2 notes that one major part of the system was needing to be able to superimpose the strain gauges over the physical testing site. The team had intended to include Azure's spatial mapping to create anchors in the physical test environment that the Hololens could track and map the digital environment over. Because the team was unable to access Azure’s spatial mapping tools, another solution was used.

Upon starting the application, the project loads the gauge array along with a green, grabable box. Using the Hololens' pinch feature, the user is able to relocate the entire array. With this new functionality, the user is able to adjust the gauge's location and superimpose the gauges over the test site.

8.2.3 Excel Reader partial

The old csv reader code has become a the new Excel Reader which reads the excel files generated by the civil engineering team and pulls the data into a usable format. Due to issues with the HoloLens system it is currently in need of some minor adjustments to be fully implemented as a result of conflicts with the project and Covid 19.

The feature makes use of the C# ExcelDataReader package to be able to read any kind of microsoft excel file and then parse that data into an array or other structure for use in the project. This allows the system to be able to change with potential updates to the software used by the Civil Engineering team as well as the client currently uses an old version of Excel that uses an older version of xlsx files.

As a result of time lost to other courses and Conner's poor reaction to the Covid-19 vaccine, it will take a short amount of time to finish the implementation for it and merge it into the master branch.

8.2.4 Hardcoded Gauge Distancing

One issue mentioned in the Beta section of this document is that Unity's scaling of objects may be different or obtuse to use in the Hololens environment. This proved to be the case. Upon testing, it was discovered that 1 unit in Unity accounted for roughly 3 ⅛ feet in real distance (coordinates for the system were given to the team in Imperial units).

To fix this, the diagram provided to the team was studied and the gauge positions carefully mapped in the digital representation of the test site.

9 Final Archive 1.1.0 SYSTEMS BREAKDOWN

9.1 Experiencing the Project

The team has filmed several demo videos as well as a demo video with commentary.

Group Commentary: https://www.youtube.com/watch?v=62BuYEoOjsA

Demo Playlist: https://www.youtube.com/watch?v=62BuYEoOjsA

9.2 Implementation

9.2.1 csvReader added

Following the issues finishing the proper reader before the code freeze, the system now implements a proper csv reader that pulls the data from the file provided by Dr. Higgins. The data is pulled via C# stream reader and then stored into a custom C# object that stores the parameters for the test in a list which can then be iterated over to play back the results in real time.

9.2.1.1 Value scaling

One issue in the project was the difference between Unity units, real life units, and the strain units used in the provided csv files. As noted in section 8.2.4, objects needed to be rescaled in Unity to 1unit=3.5 feet in order for the objects to be small enough to be handled by the user.

The next issue that arose was that the .csv file measured values between 0 and .5 units. To prevent strain gauges from moving either too much and moving out of the user's line of sight or too little and not allowing the user to detect patterns in movement, the project rescales each strain gauge value before passing them onto the gauges and displacing them.

For all gauges, a collective minimum and maximum value is calculated. From there, the difference between the two values is used to determine how to scale each value such that each gauge has a maximum displacement from their zero. Using this, the system can prevent gauges from moving too far or as force them to move enough as to be noticable.

9.2.1.2 Value zeroing

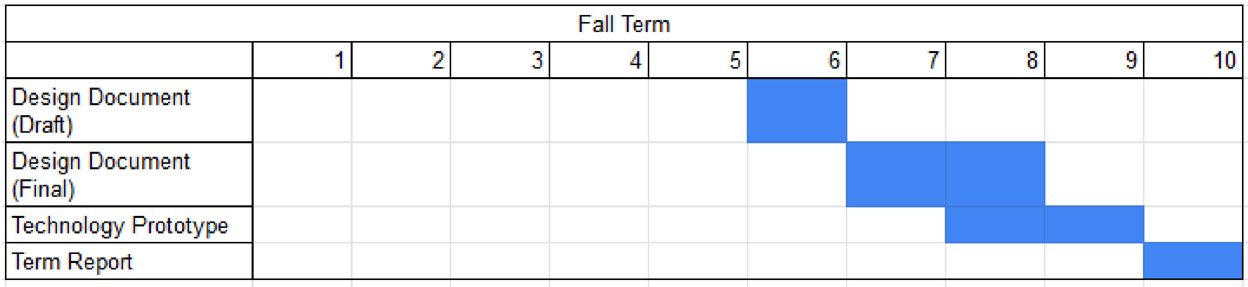
In the .csv file provided to the team, the initial values of each strain gauge varied from 0 to 1.4 units of strain. The file also indicates that at time=0, the load has yet to be put on the specimen, meaning each gauge is currently at a strainless point of equilibrium.

The problem is that as stated above, the strain of each gauge only shifted up to 0.5 units. With a range of 0 to 1.9 units, each gauge can only ever move between 25% of the available space provided, meaning each movement appears extremely small. In addition, as some gauges start with relatively high strain values, it may appear that certain gauges are constantly under massive strain.

To fix this, the system zeroes out each strain gauge, recording their initial values and using that to offset the rest of the strain values.

10 TIMELINE

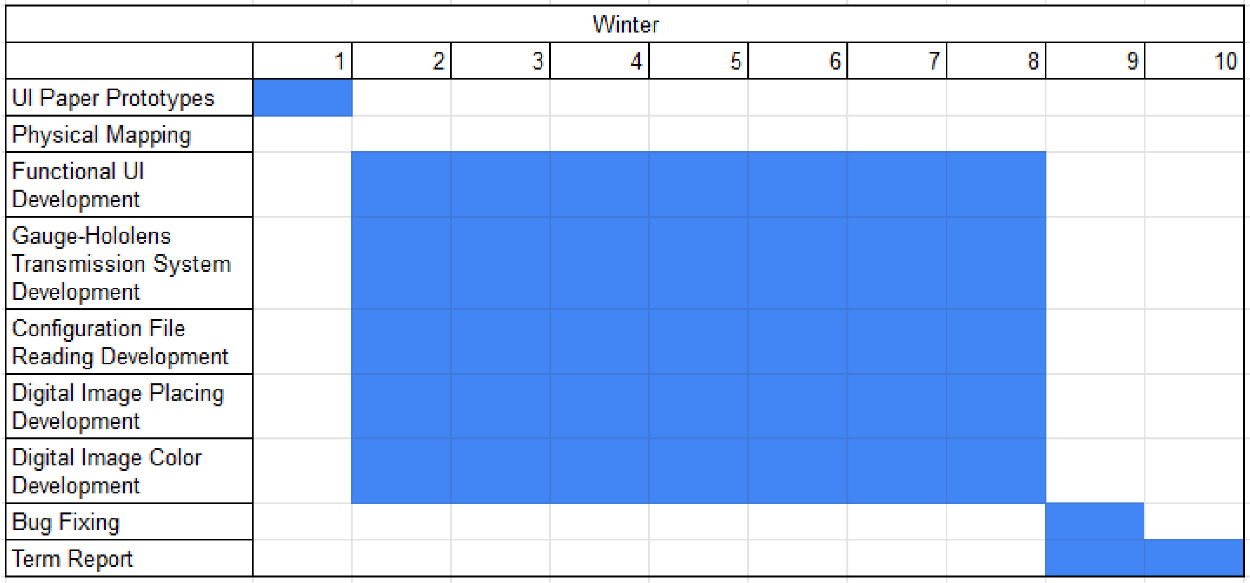
The following is the preemptive project plan for the 2020-2021 year. Note that Fall term has left out any components made prior to this document. In addition, as full details haven't been released for the term report, the team may change the timeline as needed.



Fall Term										
	1	2	3	4	5	6	7	8	9	10
Design Document (Draft)						🟦				
Design Document (Final)							🟦	🟦		
Technology Prototype								🟦	🟦	
Term Report										🟦

Figure 7: Plan for Fall Term
Current plan for remaining time in Fall Term. Likely won't change, but team may push technology prototype earlier to accommodate time

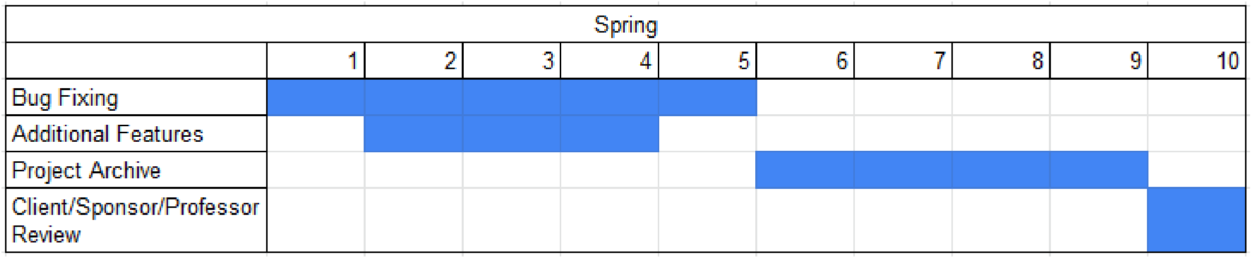
Winter term has more uncertainty than Fall term. As of now, the team is uncertain when they will be able to physically go to the testing site to make measurements of strain gauge locations and create 3d mappings of the environment. In addition, as each part of the project has been designed to be modular in that one component only depends on the output of others, weeks 2-8 will include development on all components in no particular order as of now.



Winter										
	1	2	3	4	5	6	7	8	9	10
UI Paper Prototypes	🟦									
Physical Mapping										
Functional UI Development		🟦	🟦	🟦	🟦	🟦	🟦	🟦		
Gauge-Hololens Transmission System Development		🟦	🟦	🟦	🟦	🟦	🟦	🟦		
Configuration File Reading Development		🟦	🟦	🟦	🟦	🟦	🟦	🟦		
Digital Image Placing Development		🟦	🟦	🟦	🟦	🟦	🟦	🟦		
Digital Image Color Development		🟦	🟦	🟦	🟦	🟦	🟦	🟦		
Bug Fixing									🟦	
Term Report									🟦	🟦

Figure 8: Plan for Winter Term
Physical mapping is left uncertain as team us unsure when possible. Will likely take 1 week but will not interfere with other development. Project development is split between all components between weeks 2 and 8.

The course of spring term will depend on how Winter Term goes. Bug fixes to finalize the project will continue on, but if the team deems possible, additional features to improve the project not listed in this document can be implemented. The remaining five weeks of the term are dedicated to project archive and project review. This timeline was based on the example timeline provided by the course.



	Spring									
	1	2	3	4	5	6	7	8	9	10
Bug Fixing	🟦	🟦	🟦	🟦	🟦					
Additional Features		🟦	🟦	🟦						
Project Archive						🟦	🟦	🟦	🟦	
Client/Sponsor/Professor Review										🟦

Figure 9: Plan for Spring Term

Bug fixing may be cut short if the team decides additional features are either unnecessary or there is not enough time to implement them. Project archive and review based on the provided example.

11 REFERENCES

[1] Quora, “The Difference Between Virtual Reality, Augmented Reality And Mixed Reality,” *Forbes*, 02-Feb-2018. [Online]. Available: https://www.forbes.com/sites/quora/2018/02/02/the-difference-between-virtual-reality-augmented-reality-and-mixed-reality/?sh=2f0290262d07. [Accessed: 08-Nov-2020].

[2] “HoloLens 2 vs HoloLens 1: what's new?: 4Experience's AR/VR Blog,” *4Experience Virtual Reality Studio*, 15-Jan-2020. [Online]. Available: https://4experience.co/hololens-2-vs-hololens-1-whats-new/. [Accessed: 8-Nov-2020].

[3] Mattzmsft, “HoloLens (1st gen) hardware,” *Microsoft Docs*. [Online]. Available: https://docs.microsoft.com/en-us/hololens/hololens1-hardware. [Accessed: 10-Nov-2020].

[4] “HoloLens 2: Find Specs and Features - Microsoft HoloLens 2,” *Microsoft Store*. [Online]. Available: https://www.microsoft.com/en-us/p/hololens-2/91pnzzznzwcp?activetab=pivot%3Atechspecstab. [Accessed: 10-Nov-2020].

[5] Teresa-Motiv, “Connect to Bluetooth and USB-C devices,” *Microsoft Docs*. [Online]. Available: https://docs.microsoft.com/en-us/hololens/hololens-connect-devices. [Accessed: 10-Nov-2020].

[6] Mike Taulty and Mike Taulty, “Windows 10, UWP, HoloLens & A Simple Two-Way Socket Library,” *Mike Taulty*, 21-Feb-2017. [Online]. Available: https://mtaulty.com/2017/02/21/windows-10-uwp-hololens-a-simple-two-way-socket-library/. [Accessed: 10-Nov-2020].

[7] Mamaylya, “HoloLens 2 gestures (for example, touch, hand rays, and gaze) for authoring and navigating in Dynamics 365 Guides - Dynamics 365 Mixed Reality,” *HoloLens 2 gestures (for example, touch, hand rays, and gaze) for authoring and navigating in Dynamics 365 Guides - Dynamics 365 Mixed Reality | Microsoft Docs*. [Online]. Available: https://docs.microsoft.com/en-us/dynamics365/mixed-reality/guides/authoring-gestures-hl2. [Accessed: 10-Nov-2020].