Creating Digital Twins of Smart Sensors for the HoloLens 2

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

James McCampbell

Spring, 2023

Technical Project Team Members

Andrew Tamberrino

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Brad Campbell, Department of Computer Science

Abstract

With Extended Reality (XR) becoming more accessible in the recent past (XR headsets are cheaper and development tools are free and open to anyone willing to learn how to design software) brand new possibilities of understanding our real environment in a different way and new interactions between the physical and virtual world are available. With this in mind, we have taken the opportunity to develop a new Augmented Reality (AR) application to be run on the HoloLens 2 that can link sensors in the physical world to their respective digital twins in the virtual world. What makes this application interesting is that the digital twin sensors are mapped to the physical sensors so that a user could walk around in the physical space and view the digital twins mapped to their real world locations. With this feature, users will be able to better understand and interpret the data in a new and more intuitive way.

Our main goal is to make this experience as seamless as possible to the end user(s) so that, when wearing the HoloLens, the application runs smoothly, the data read by each sensor is visible to the user, and the data updates as soon as it can – essentially as soon as the database updates from the sensor readings being posted. In these set goals, we aim to make the application valuable and credible for the potential end user and, in doing so, we will be making the data read by these sensors more accessible.

To complete these requirements, we were given access to the Link Lab at UVA; many spaces around UVA are monitored by a variety of sensors, but the Link Lab has many sensors of different types concentrated in a relatively small area. We were also given access to two HoloLens 2 headsets. With them, we set out to understand how interactions work within a HoloLens 2 and then moved on to understand how the development, deployment, and use of a custom-built application would work with the headsets. Once we understood the basics, we

moved on to understand more complex features and added them to our working project until we had a completely usable application.

Now that the project has come to an end, we were able to create something that shows the data posted to the database on a sensor object in the virtual world that overlays the sensor in the physical world. Further than this, our application is portable to other devices as the auto-population of these sensor objects utilizes a cloud-based service. We feel that with this application, we are further opening the doors to a more deeply interactive world.

Introduction

A recurring issue in the design of large systems is the lack of accessibility to crucial information at times when it is most needed. In the context of our project, there aren't many risks or threats to the building, at least that could be detected by the sensors placed around the Link Lab, but the application is similar. Having a more readable and understandable way to interpret data could be useful in an environment where sensors are difficult to read or the information is unintelligible. Even in the case where the information is read in from some other software, even with simple queries, the visualization of the data is lost. With our proposed HoloLens application, we are proving there exists the possibility of an app that can create tangible visuals, setting a baseline for future use here at UVA and beyond. On top of this, VR and AR development is a topic that has been interesting for us both and we wanted to tackle a brand new problem in a brand new environment.

With our new app in mind, we had to think about what we wanted to accomplish by the end of the semester. To ensure our app was up to our standards, we wanted to verify that our

average frames per second (fps) were within a tolerable range (greater than 45 fps, and since we can't achieve higher than 60 fps, no more than 60) – both in a smaller use case and in a case where every digital sensor object may be loaded in the app all at once. We also wanted to verify that the sensor objects accurately displayed the data read on the physical sensors during runtime; the data appears, is readable, and is updated as frequently as the database. Finally, using the application should be intuitive; the placement of digital twins makes sense to its physical counterpart, the addition and removal of sensors works well, and sensors are interactable.

When thinking about the design of the application, we considered what the closest alternative solutions are that exist to read the sensors' data in the Link Lab. Besides our work, the best way to view the data is by querying the InfluxDB database, and to do that there are loops to jump through since the database, especially while we've been interacting with it, has proven to be over-cluttered and not always accurate or up to date on information. Another lacking project for this idea is a loose implementation for different sensors to have their own site, as there are QR codes on some of the sensors that take you to a unique URL, but there is no sensor-specific data displayed there. In terms of implementations outside of UVA, there are libraries and projects for digital twins of Internet of Things (IoT) devices and many different libraries and projects for linking the virtual world to the physical one, but we will discuss these later.

By the end of the project, we found that our goals set at the beginning of the project, especially that of implementing an in-app user interface (UI) and the functionality that would come with that, were ambitious. While the final application is still missing some features and functionalities we intended to include at the beginning of the project, these can be expanded on in the future.

Related Works

As for research in the field, there are plenty of options available to interact with IoT devices with an AR application, including Azure Digital Twins. There are also many libraries for placing digital items in physical space, but some libraries that we looked at were not supported any longer by newer versions of Unity, and so research for some of these projects have slowed down more recently. Further than this, there aren't many robust implementations of both IoT-linking and world-linking together.

Probably the most robust project, most similar to ours, was the XRShark project in which they created an AR application (with VR capabilities) that interpreted the data read in by the packet sniffing software wireshark and created visuals for how the devices in their space communicated with each other over networks. The XRShark project utilized a Unity application to visualize the packets passing between devices. In order to localize their Vive Pro headset within their space, they used a localization system called Harmonium which uses ultrawideband (UWB) RF localization. This was something that seemed too complex for our purposes and would have required the purchase of, and dependency on, other devices for the HoloLens application to work (Clark et al., n.d.).

Another project that is a little closer to our implementation, "Digital Twins for Smart Cities," discusses how their digital twins for their sensors were connected by sending a Hypertext Transfer Protocol (HTTP) request to query a database. The data accessed was saved from sensors in street lamps, possibly similar to the sensors in the Link Lab at UVA, and utilized in a way that could be visualized by the user wearing a HoloLens 2 in real time. One difference between our plan and this project is the lack of implementation for connecting this visualization

of streetlamps to the real world streetlamps. The game objects were just interactable objects that could be moved around by the user (Piper et al., 2022).

Finally, the last project similar to ours that we looked at was one in which Industrial IoT (IIOT) devices could be visualized in different states as parts of a large machine. In their case, they utilized small and large models of the same machine that utilized sensors detecting the flow of cold and hot air going in and out of the machine. The smaller model didn't need to line up with the actual larger machine, but the temperature of the air could be visualized with the pipes in the hologram showing red and blue for hot and cool air respectively. While the IIoT side of the project was intriguing, their localization process for placing the larger model was more useful to us. Using some sort of coordinate system calibration, the large model of the machine was able to be lined up with the real machine almost perfectly after tapping three points near the machine to create two vectors that, with an orthogonal vector, create a coordinate system with an origin. This idea of the small-scale localization of digital twins mapping to physical objects was what we ended up roughly using in our project (Husinsky et al., 2022).

As stated before, previous localization processes included the UWB localization and then this three point localization, but there were other methods that we looked into as well including utilizing spatial anchors or QR codes as space pins (which is a topic we will delve into later). We decided to use spatial anchors because this does not require us to add anything to the Link Lab for our app to work and because they can work without any user input. In terms of gathering data from the sensors located in the Link Lab, our two options seemed to be to connect directly to the IoT devices or query the database that is already set up to store the sensor readings. Though connecting directly to the sensors would potentially yield more up to date results, the sensor data is currently only sent to a gateway that prepares the data to be written to the database.

Because of this, querying the database seemed like the simpler option without losing any significant functionality or accuracy, as the sensors post data fairly regularly.

As for our improvements over the previous work, we intended to create an app that, after an initial set up stage, could be deployed to any HoloLens 2 and each headset would be able to visualize the same data just by walking around the space. As for requesting the data from the database, in order to save resources, we only query for updated sensor information if the user's gaze is on the sensor. Since the processing power behind the HoloLens isn't amazing, this helped us achieve great performance.

Design

1. System Background

Our app is designed around an Azure Spatial Anchors (ASA) system that allows the HoloLens 2 to recognize where it is within the Link Lab (*Azure Spatial Anchors Overview - Azure Spatial Anchors*, 2022). Before discussing this though, it's important to understand a bit about the spatial mapping that the HoloLens does in the background. While it is being used, a HoloLens will create a map – made out of small triangles that, when put together, is called a "mesh" – out of its environment that the HoloLens can use to track its location and determine where to render holograms around it. Unfortunately, the HoloLens is expected to have up to a 10% error rate with the distances that it measures within its surroundings (*Space Pins*, 2022) . Because of this, Microsoft's mixed reality documentation divides HoloLens applications into different categories based on the size of the area that the app covers and each category has different strategies that are recommended for keeping track of the environment (*Coordinate*

Systems - Mixed Reality, 2023). According to this classification, our application that services the entire Link Lab is considered a "world-scale" application, as the space needed for the app is larger than a 5 meter radius, and it is recommended to use spatial anchors for localizing the HoloLens in the playspace.

Spatial anchors are virtual objects that allow the HoloLens to take and save snapshots of the map it has built of the surrounding area that can be used to find its location in the future (*Spatial Anchors - Mixed Reality*, 2023). These spatial anchors can be linked together to form a more solidly defined space within the scope of the application. To do this, we used an open source project called World Locking Tools (WLT) which automatically places these spatial anchors in the background of the app, locking in the playspace (*The Basic Idea*, 2022). In the case where the space is not well defined, it is called "spongy space." As spatial anchors are placed, that space becomes closer to being "World-locked" where the 10% error problem is reduced and the app can support larger scale projects. This helps to link the digital world to the physical world for large, static spaces.

In the case of a space where large items are moved often or spaces may be changed, the spatial anchors can become "confused" as their saved mesh data is no longer accurate. In testing, we also found that the world-locked space would often get lost between app sessions if the HoloLens was unable to localize itself when the app was first opened. This means that WLT cannot be used alone to reliably localize the user in our application. Instead, we decided to split our spatial anchors strategy into two parts. Firstly, the WLT package is still used to reduce the 10% error. Using WLT means that, due to the size of the Link Lab, the app has much less difficulty keeping track of the location of sensor holograms throughout the space as the user walks across the building and back. To avoid the issues of persisting spatial anchors getting lost,

particularly between app sessions, we clear the WLT anchors each time the app is closed. Secondly, we manually place intentional spatial anchors in the app so that the HoloLens can recognize different areas in the Link Lab independently without worrying about spaces changing across the building. These spatial anchors were placed in the form of "space pins" which are, simply put, spatial anchors that have their own coordinate system. With these independent coordinate systems, we are able to auto-populate sensors in a given space once the HoloLens recognizes a spatial anchor and not have to worry as much about spatial anchors getting lost in a dynamic space like the Link Lab.

We decided to implement the space pins using Azure Spatial Anchors. Azure Spatial Anchors are a cloud based system for storing and searching for spatial anchors using Microsoft Azure (*Azure Spatial Anchors Overview - Azure Spatial Anchors*, 2022). Using Azure Spatial Anchors allows us to easily port our app between different HoloLens headsets without worrying about saving and transferring the map and spatial anchor data between the HoloLenses. Any HoloLens can query Azure for the list of spatial anchors and use them to recognize the points specified throughout the Link Lab as long as it has a sufficient understanding of its surroundings. Using WLT in conjunction with Azure Spatial Anchors also means that we do not have to worry as much about the holograms populated around a space pin drifting in space as the user walks far away and back to them. WLT, in addition to the space pins, helps the app keep track of the Link Lab space better so that the holograms are kept in their location relative to the Link Lab without having to query for the spatial anchors again.

2. System Implementation

For our specific app implementation, we placed Azure Spatial Anchors throughout the Link Lab near all of the sensors. It is recommended not to place holograms more than 3 meters away from the spatial anchor that they are populated around so the spacial anchors in the Link Lab were placed accordingly based on the locations of the sensors (*Spatial Anchors - Mixed Reality*, 2023). In order to create the anchors, we used a modified C# script from an Azure Spatial Anchors tutorial that demonstrated how to create and request Azure Spatial Anchors (*Tutorial*, 2023). This script is still present in the Unity project for the app so that we can reactivate it to add or update Azure Spatial Anchors in the future.

Each spatial anchor automatically populates holograms of the sensors around it once it is found by the HoloLens and the holograms are updated with the most recent sensor readings by making HTTP requests to the InfluxDB database storing the sensor data. A hologram for each sensor type that the app supports can be seen in Figure 1, along with the sensors that the holograms represent. Initially, the large number of HTTP requests being created as the app updated each sensor would cause a significant lag spike. We solved this problem by using gaze detection, which determines which hologram the headset is pointing towards, to determine which sensor the user is looking at and solely updating that sensor once per second. Updating the sensor holograms only when necessary allowed us to reach our goals for the app to run smoothly at a high frame rate.

Figure 1: Sensor and Hologram Types (from left to right: Awair sensor, temperature and humidity sensor, CO2 sensor, door sensor, light level sensor, and dual motion sensor)

We chose to use Azure Spatial Anchors for recognizing spaces around the Link Lab instead of alternative means such as through the use of QR codes because most viable alternative methods require making modifications or placing external hardware to the space. We wanted our app to be as unobtrusive to the Link Lab as possible. Some other design choices that we made were the results of failing to implement alternative methods. For example, we were unable to use the InfluxDB C# library to query sensor data for our holograms despite using older versions of the library that should have supported the dated version of the database. This led us to manually create HTTP requests to the database for information. We had also intended to use MRTK3 UIs, interfaces pre-built to be used for HoloLens devices, in order to add the ability to interact with the sensor holograms, but we were unable to get them working properly. This led us to use custom cubes and 3D text objects as our holograms. Lastly, we were originally using Unity version 2020.3.25 for the project as the Azure Spatial Anchors tutorial suggested, but this Unity version did not work with the HTTP request methods that we were using until we updated our project to use the latest version of Unity 2020.3 (2020.3.47).

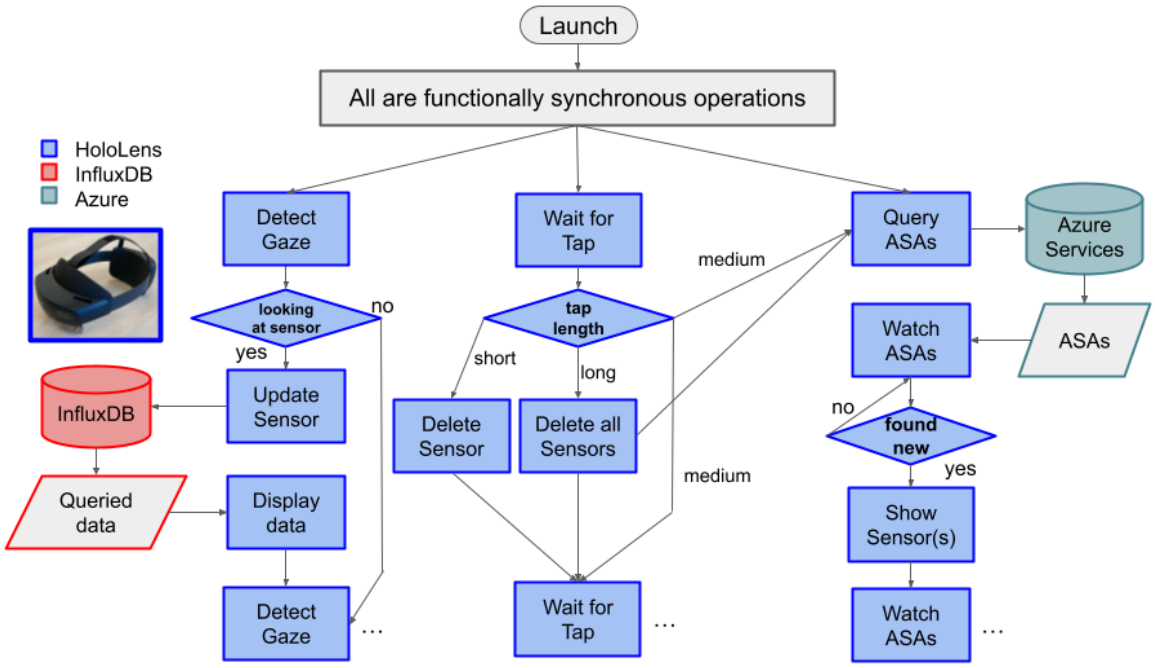

Figure 2: System Diagram

Evaluation

Ultimately, we were successful in creating an app for a HoloLens 2 that allows a user to walk around the Link Lab and view digital twins of the sensors throughout the space. Many of the spatial anchors placed around the Link Lab are easy for the HoloLens to detect, such as the anchors in the copier room and the kitchenette, due to the unique features of the building around those locations. However, due to the limitations with how well a Hololens 2 can detect and recognize its surroundings, it will occasionally mistake different spaces for each other and place sensor holograms in the wrong location. In that event, the user is able to make a 0.5 second air tap near the holograms to delete them. In the event that a spatial anchor is not found – possibly because it was previously placed in and deleted from the wrong location – the user is able to make a 2 second air tap to restart the search for spatial anchors. A user can also make a 10

second air tap in order to reset all of the sensors that have been found by the app so far, which can be useful if the hololens loses its position in Link Lab and cannot find it again. While these controls may not be intuitive, they make it easy for the user to correct issues that the app has with populating holograms.

In addition to using the app to view sensor information, a user can set the app to developer mode in Unity and reflash it to a Hololens so that they can create new Azure Spatial Anchors and nearby sensor holograms. This allows a user to short tap to place new anchors and the new anchor ID will be output to the debug log. The anchor ID and any associated sensor holograms can then be added into the Unity project so that the app can display new sensor holograms.

For the purpose of testing the performance of the app, we created a test scene that had 200 Awair sensor holograms populated in a grid in front of the user, simulating the large number of smart sensors placed throughout the link lab. Having this number of holograms populated around the HoloLens did not prove to be an issue, as the app continued to run at about 60 frames per second. During normal use, the app also runs at about 60 frames per second with an occasional short frame rate drop. Overall, the app reaches our goal for running smoothly at a high frame rate.

For determining the portability of the app, we deleted all of the mapping data off of a Hololens 2 and reflashed our app to it. The Hololens was quickly able to find the Azure Spatial Anchors around it and repopulate the sensor holograms. This means that the app should be able to be deployed to any Hololens 2 without issue.

One of our goals that we were unable to accomplish was to allow the user to interact with the sensor holograms. We had hoped to allow the user to see different information such as historical data from the sensor or statistics about data from the sensor. Unfortunately, due to the time frame of the project, our inexperience with the software used for the project, and the lack of support for the software we were trying to use, we were unable to realize this goal. While we did not fully realize our goals for this project, we were able to produce a scalable app that could be used to provide greater accessibility to the data from smart sensors throughout Olsson Hall and the rest of the University.

Discussion

When analyzing our approach, there are some possible issues that may develop over time. A few of these examples include:

1. If a sensor is ever relocated or replaced, the app would not be able to recognize this and the Azure Spatial Anchors and sensor ID objects and queries would need to be updated.
2. If the surrounding area around an Azure Spatial Anchor changes enough, it's possible that the spatial anchor gets lost and the sensors mapped around it will no longer populate as the anchor can no longer be successfully found.
3. If a user moves too quickly through the Link Lab, it's possible the HoloLens will get lost in its space, but this is more of a limitation of the HoloLens itself.
4. If the Azure account is no longer active, then the Azure Spatial Anchors will not be accessible anymore and the sensor objects will no longer populate.

5. If the application ever needs to be updated, then it must be redeployed to every HoloLens with the outdated software.
6. Because our sensor objects query a database rather than the sensors themselves, we are not always getting the most recent and accurate data.

Some concerns some reviewers may have is the unpredictability of the application itself. Sometimes everything works as intended and other times, we experience errors with sensor objects updating in-app, accurately placing sensor holograms, and with the HoloLens getting lost in its space.

As for addressing some of these issues, there are many directions to go from here to create a better application. Firstly, while our spatial anchor approach works, it isn't as robust as we would have liked it to be. Each Azure Spatial Anchor has its own definition and has no relation to the others. This means that the Hololens must always be able to find an Azure Spatial Anchor for the surrounding sensors to be populated, and there is no way to view the sensors if an anchor fails to be found. If these anchors were instead linked together in the same coordinate space, it could help to more reliably and accurately populate the sensor holograms in terms of their physical counterparts. It is also possible to still look into QRCode space pins that would act as fixed, wall-mounted coordinate systems, or to look into the possibilities of UWB RF localization. Further, the app would be more complete if every sensor in the Link Lab were implemented with a digital twin. This would be relatively simple to do as the prefab files are already created. Another extension to the project could be to add more functionality to the sensor objects. Finally, these options may require more initial research, but it could be productive to look into implementing the application in Unreal Engine or look into porting the HoloLens application as a Mobile AR application or to other AR devices.

Of these suggestions, one of the most plausible ones that would make a great contribution to the project would be linking the spatial anchors together to create a more solid coordinate system to populate the sensors off of. This would work similarly to how the application works now, however, each Azure Spatial Anchor wouldn't populate new sensors, but instead each successfully found anchor would help align the coordinate system within the app with the entire Link Lab. The second suggestion is to look into implementing more functionality behind the sensor objects. One example of this would be adding a function to query historical data points in time or even find average values for user-specified ranges of time.

Conclusion

Our smart sensor application effectively improves access to and the understandability of the sensor data readings from smart sensors placed throughout the Link Lab. Using Azure Spatial Anchors, the app is able to create digital twins that are placed over top of each smart sensor to display the sensor readings. The use of Azure Spatial Anchors allows the device to be ported to any Hololens 2 headset without any additional setup. The app is also scalable and simple to use. This means that it could be adapted for use in a variety of settings, either across UVA, in industrial environments, or even in someone's smart home.

This new way of interfacing smart sensor data could be used to help monitor large and complex spaces or to help people visualize the differences within varying parts of their homes. Sensor holograms can help the user find systems that are inefficient or malfunctioning. Sensor holograms could also display measured electricity use which could be added to the app so that

users can see the impact that certain devices in their home have on their electricity usage. Ultimately, our app can greatly improve our ability to understand and maintain smart buildings.

References

Azure Spatial Anchors overview—Azure Spatial Anchors. (2022, February 21). https://learn.microsoft.com/en-us/azure/spatial-anchors/overview

Clark, M., Newman, M. W., & Dutta, P. (n.d.). *XRShark: Mixed Reality Network Introspection*.

Coordinate systems—Mixed Reality. (2023, January 19). https://learn.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems

Husinsky, M., Schlager, A., Jalaeefar, A., Klimpfinger, S., & Schumach, M. (2022). Situated Visualization of IIoT Data on the Hololens 2. *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, 472–476. https://doi.org/10.1109/VRW55335.2022.00104

Piper, W., Sun, H., & Jiang, J. (2022). Digital Twins for Smart Cities: Case Study and Visualisation via Mixed Reality. *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, 1–5. https://doi.org/10.1109/VTC2022-Fall57202.2022.10012753

Space Pins. (2022, January 31). https://learn.microsoft.com/en-us/mixed-reality/world-locking-tools/documentation/concepts/advanced/spacepins

Spatial anchors—Mixed Reality. (2023, March 3). https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-anchors

The basic idea. (2022, January 31). https://learn.microsoft.com/en-us/mixed-reality/world-locking-tools/documentation/concepts/basicconcepts

Tutorial: Create a new HoloLens Unity app - Azure Spatial Anchors. (2023, March 10). https://learn.microsoft.com/en-us/azure/spatial-anchors/tutorials/tutorial-new-unity-hololens-app