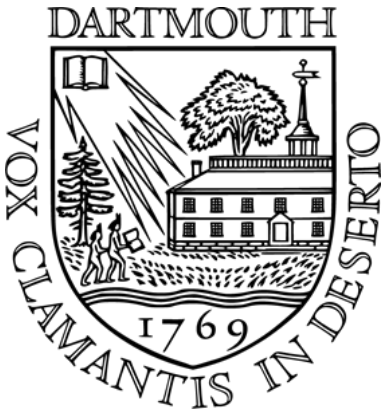
5-31-2018

Overlaying Virtual Scale Models on Real Environments Without the Use of Peripherals

George Hito
Dartmouth College



Overlaying Virtual Scale Models on Real Environments Without the Use of Peripherals

George Hito
Advisor: Professor Wojciech Jarosz
Dartmouth College Computer Science
Technical Report TR2018-858

May 31, 2018

1 Abstract

The Architecture, Engineering, and Construction (AEC) industries have become increasingly reliant on detailed 3D modeling software and visualization tools. In the past few years, the arrival of effective and relatively cheap virtual reality has transformed the workflows of people in these professions. Augmented reality (AR) is poised to have a similar if not greater effect. For that to occur, the transition into using this technology must be seamless and straightforward. In particular, a clear bridge should be made between Computer Aided Design (CAD) models and their corresponding real-world environments. Using Unity and Microsoft's Hololens, I developed a method of automatically overlaying a scale model of a virtual room on its real-life counterpart, allowing a user to walk around their physical environment and see how virtual features correspond to what is already built.

2 Introduction

The release of relatively cheap, consumer-friendly devices in recent years has propelled virtual reality (VR) into the spotlight. The HTC Vive, Oculus Rift, and Windows Mixed Reality are quickly transforming the workflows of professionals in many industries — in particular, architecture, engineering, and construction. As AECMag states "It is one thing to model a building in a 3D CAD system, but using VR to experience how it will feel and function can take design to a whole new level. Architects can exist inside their designs, encouraging bold new ideas and more iteration" [1]. Startups have risen to fill this growing desire for more detailed design tools, with many in the space racing towards the same goals — a good augmented reality product suite being high on the list.

Last summer I had the opportunity to work at IrisVR, a leader in the field of AEC visualization. There I learned that the AEC workflow (*Figure 1*) is highly dependent on extensive iterations. It is impossible to predict every issue that may come up during construction of a building, and constant changes must be made to the initial design during the entire process. As a result, the efficiency and reliability of visualization software becomes paramount. VR allows for an AEC professional to better understand the scope and scale of the space they are constructing and catch errors before they end up in the physical design. The interaction loop between the design and construction teams on a project condenses, and ultimately fewer iterations are required to satisfy the client. For example, when the leading construction firm Mortensen was contracted to build the Atlanta Braves' new stadium, the use of visualization software helped shave $1.7 million from the total cost[4]. While viewing 3D models through a 2D screen is useful, it is an entirely different experience to walk through your design. VR has already begun to transform the AEC industries — but augmented reality is hot on its heels.

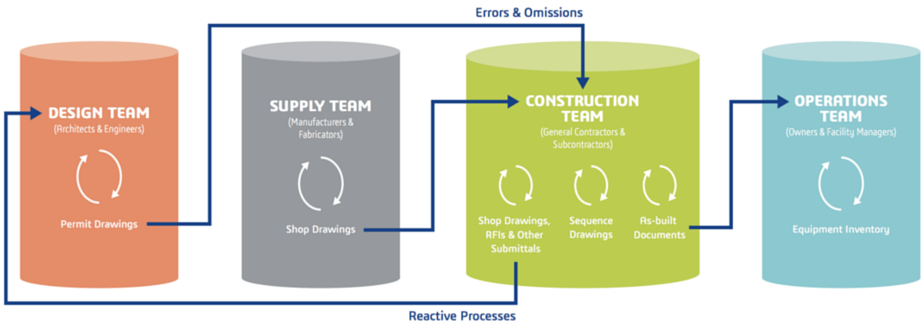


Figure 1: An overview of the AEC workflow.

2.1 QR-Code AR and the Job Site

AR offers an additional step to condense the AEC workflow (*Figure 1*) by bringing these visualization tools directly to the job site. While current leading AR devices sacrifice high resolution and application complexity for the sake of being lightweight, performant, and mobile, my experience with the Hololens and feedback I received from professionals in the industry showed that this shortcoming did not detract from the usefulness of the technology. In fact, AR has been used in the AEC industries before in the form of mobile apps. The aforementioned firm Mortensen designed an app that utilizes Google?s Tango API and QR codes placed around the borders of a project site (*Figure 2*) to overlay a building's 3D model over the real environment. This app is being used by LMN Architects as a central tool for the CSE2 project [5], a $105 million investment by the University of Washington into a new academic building.

Figure 2: A screenshot of Mortensen's AR app that utilizes QR codes to place 3D models of buildings on real environments. Note the low resolution of the building model in comparison to the surrounding environment.

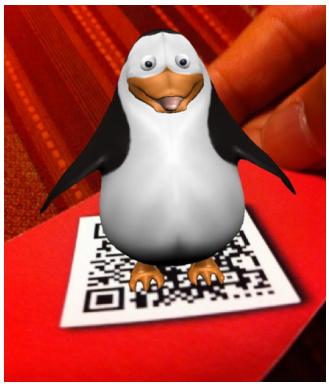


Figure 3: A 3D object overlaid on a QR code in real-time using DAQRI's software.

Using QR codes for virtual overlay is not a new concept — the idea was originally introduced by DAQRI in 2011 [10]. They came up with a system to overlay 3D objects onto a static QR code in real-time (*Figure 3*), combining computer vision and AR in a way that had never been seen before in the consumer market. The initial purpose of it was to attach another dimension of virtual information to physical objects, combining the two more seamlessly than a 2D screen.

However, the disadvantage of QR codes is that a user must always have access to the physical QR code in order for their application to function. The code acts as a visual 'anchor' for whatever 3D object is being displayed on top of it. That object's orientation can then be adjusted to match the orientation of the QR code, but the issue with this method is that it requires a physical QR code to initialize the overlay and any subsequent movement done with the code outside the frame of vision of the device?s camera will result in significant 'drift' accrued over time as the tracking is performed relative to the location of the QR code. Google Tango ameliorates that issue by performing Simultaneous Localization and Mapping (SLAM). SLAM allows a device to process the output of RGB-D (RGB-Depth) cameras and inertial sensors to understand its relative location within a space, which can then be used to generate a 3D map of surrounding physical features (*Figure 4*). SLAM was born in robotics as a way for a robot to gain more understanding of its environment, and it has gradually made its way over to the AR/VR sector, with Tango being the first consumer product relying almost completely on SLAM.

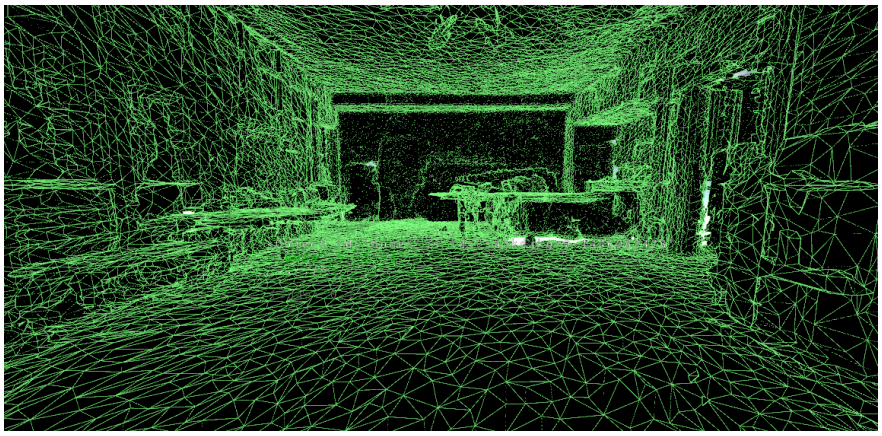


Figure 4: 3D room scan with Google Tango.

However, Tango was only accessible through the purchasing of certain smartphones and tablets that came bundled with depth sensing, and it therefore did not gain a large market share beyond developers and hobbyists. While Tango was the first dedicated product for AR through SLAM, it was not the first exposure the average consumer had to this technology. That came by way of a much cheaper, more ubiquitous device: the Microsoft Kinect.

2.2 Microsoft Kinect and Room Scanning

The Microsoft Kinect (*Figure 5*) is traditionally known as a peripheral device bundled with the Microsoft Xbox 360, used to track the posture of a person's body as input to an application or game.

Figure 5: The first iteration of the Microsoft Kinect.

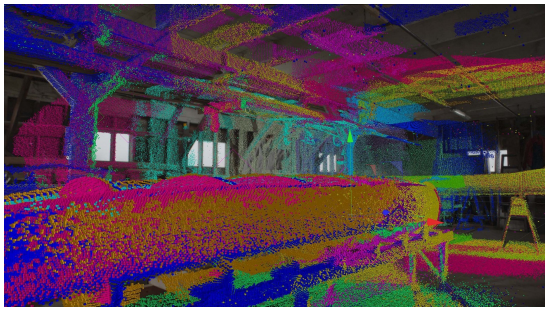


Figure 6: Output of the RGB-D cameras aboard the Kinect, mapping a warehouse.

Microsoft gave developers access to the raw data of this camera stream, presumably to test game functionality and reliability. One of the additional effects of this was to give the public access to a relatively cheap and effective method of obtaining raw RGB-D datastreams (*Figure 6*). With some post-processing on this data, software was developed to convert these streams into 3D models. Companies were even founded on the concept [15]. This experimentation was not limited to 3rd parties, though. A 2011 partnership between the Imperial College of London and Microsoft Research led to a project called *KinectFusion: Real-Time Dense Surface Mapping and Tracking* [16]. KinectFusion outlines a method of high-resolution scanning of 3D objects and in particular discusses how they developed a drift-free SLAM algorithm for spatial navigation using only Kinect sensor output. It also emphasizes the importance of such a technology for AR — five years before the official announcement of the Hololens.

Microsoft discontinued the Kinect in late 2016, moving those engineers over to the Hololens project. The combination of this project and internal move suggests that the Kinect was part of a long-term plan for the generation of a full-fledged dedicated AR platform (Hololens). This was a huge step forward from QR code overlays, and set the foundation for modern professional-grade AR.

2.3 Microsoft Hololens, DAQRI Smart Helmet, and Professional-Grade AR

Microsoft began with body posture tracking on the Kinect, and DAQRI with QR codes on smartphones. However, both have narrowed their focus in the past few years on similar devices — the Hololens and Smart Helmet, respectively.

Figure 7: The Microsoft Hololens comes with much the same sensors and physical capabilities as the Smart Helmet, but with the advantage of having a software backend bolstered by engineers that worked on Kinect.

Figure 8: The DAQRI Smart Helmet combines augmented reality capabilities with safety features for a product tailored directly towards industrial worksites.

The Hololens (*Figure 8*) is a full Windows 10 PC with depth sensors and cameras mounted on the front [17]. Dedicated chips run SLAM on the output of these sensors and generate a true-to-scale 3D map of physical features in the environment (*Figure 9*). From that 3D model, a collection of vertices is grouped together behind-the-scenes in a spatial anchor [18], which serves a similar purpose as early QR codes. Assigning a virtual object to one of these anchors allows its orientation in space to be saved relative to the anchor?s physical orientation, and as a result the object can maintain its own position and orientation even during user movement. In addition to this, the sharing of these anchors among different devices can allow multiple users to collaborate within augmented reality, seeing the same objects in the same locations. Microsoft's first venture into AR for AEC was a partnership with Trimble, the maker of popular 3D modelling software SketchUp. In early 2016, they released SketchUp Viewer for Hololens, an application that allowed users to both view 3D models in AR and collaborate with each other on them (*Figure 10*). Its price point of $1,500 made it prohibitively expensive to anyone except those who already used SketchUp as a dedicated part of their workflow. Initial reviews of the product were generally neutral, with feedback best summarized as *interesting, but not great... since the model you're viewing has no spatial relationship with the space you're in* [22]. While the software included *some dream features for professionals that use SketchUp*, it wasn't enough to make the product essential to an AEC professional. More than anything else, it was a proof-of-concept.

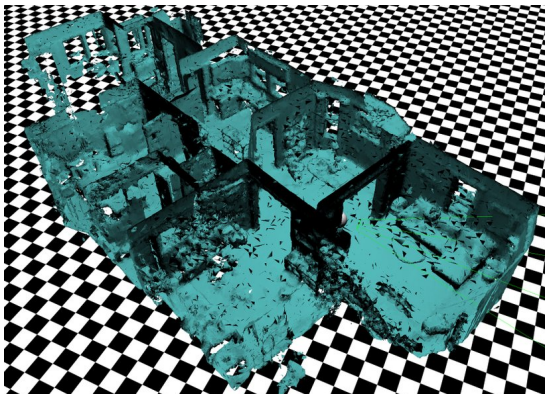


Figure 9: A 3D scan of a space, generated by the Microsoft Hololens.

Figure 10: Users work collaboratively to view a 3D model in SketchUp Viewer, placed using a spatial anchor to share its location and orientation across devices.

The Smart Helmet's (*Figure 7*) hardware specifications are much the same [19] as the Hololens, with the capability to perform SLAM and track a user throughout an environment. However, the Smart Helmet differs greatly on the software side — it does not generate and save a 3D map of its surroundings, and thus lacks much of the hologram persistence and user collaboration features that make the Hololens an attractive product to build for [11]. In addition, the price point varies wildly between the two devices, with the Hololens costing $3,000 and the Smart Helmet a comparably enormous $15,000. Regardless of these striking differences, DAQRI was determined to compete with Microsoft. They partnered with Mortensen in late 2016 [6] to create a Smart Helmet application that involves the overlaying of future building features on an unfinished environment (Figure 11). Ducts, wiring, framework, and more can be visualized with correct location and orientation within the structure.

Figure 11: A screenshot from live operation of the Smart Helmet app produced in partnership with Mortensen. Note the overlay of future building features on the unfinished environment.

This is the logical next step for AR visualization of CAD models and on the surface appears to ameliorate the issues with SketchUp viewer, but users of the Helmet report [21] that its tracking is nowhere near as good as the Hololens'. The frontend concept is there, but the backend tracking and stabilization isn't. Microsoft's extensive prior research with the Kinect and the years of development spent on the Hololens show that solving this is no easy feat.

2.4 Next Steps in AR AEC Visualization + Overview of Thesis

The combination of 3D environment scanning, SLAM, and spatial anchors — all of it bolstered by years of prior internal research — make the Hololens user experience stable and seamless in comparison to the Smart Helmet. The price point is also much more convenient than that of the Smart Helmet.

For those reasons, I chose the Hololens when considering which device has the most potential for prototyping. I have developed and tested an algorithm for aligning true-to-scale CAD models with their corresponding physical environments using only the 3D map output of the Hololens. My goal for this thesis was to attempt to replicate DAQRI's application functionality on the superior tracking of the Hololens. In addition to that, I wanted to extend it for collaborative use by AEC professionals on any project, rather than catering the app to a particular job site as DAQRI did with Mortensen. From a mixture of personal experience in the industry and extensive research on what solutions are already out there, I believe that an efficient, user-friendly solution to overlaying CAD models on physical environments is the necessary feature that AR needs to be widely adopted as a tool for AEC professionals.

3 Methods

3.1 Unity and the Hololens API

Unity is a 3D graphics engine developed by Unity Technologies [24], well known for its ease of use and cross-platform capabilities. Its reliability and popularity has attracted the attention of some of the largest VR/AR companies in the world as the go-to solution for building games, applications, and simulations. In particular, Microsoft directs all interested Hololens developers to a set of tutorials [25] built in Unity that showcase the capabilities of the device. While it is possible to use other engines, Unity is by far the most approachable and well-supported.

Most holographic apps follow the same design principles for barebones tasks like user interaction, environment processing, and usage of device capabilities. Microsoft has acknowledged that repetition and started an open-source project called the Mixed Reality Toolkit (MRTK) [26] that seeks to simplify AR development by providing a set of universal scripts and premade objects for tasks like analyzing user input, connecting multiple devices for collaboration, etc. For this project, we are particularly interested in the Spatial Mapping tools contained in the MRTK.

3.2 Spatial Mapping in the Mixed Reality Toolkit

'Spatial Mapping' refers to the Hololens' functionality of constantly scanning its environment during operation and generating a 3D model from that scan (*Figure 12*). This model is saved as a series of connected triangles which roughly approximate the shape of the scanned room. The level of detail for this scan can be controlled by the user and increased at the expense of scan time — 1,200 triangles per cubic meter is recommended to strike a balance between detail and time. I chose to use this value for my scans as it is standard for most applications.

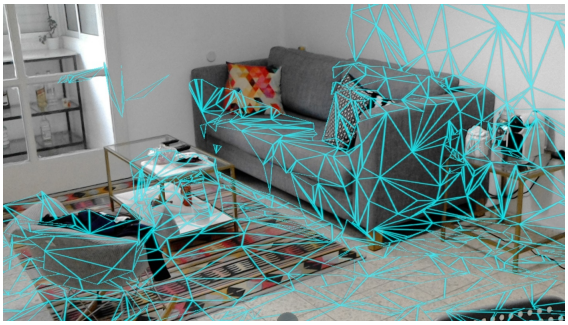


Figure 12: A partial spatial mapping scan performed by the Hololens. It is difficult to tell on a 2D image, but the triangle mesh follows the physical structure of the room.

To test the robustness of the alignment algorithm, I needed an efficient method of compiling several 3D scans of rooms. Fortunately, the MRTK provides a series of scripts and code samples for that purpose. I used the SaveSpatialMapping.cs script for this, and extracted the models from the Hololens using the Windows Device Portal [27], a set of developer tools that allows for more granular control of the Hololens file system, running processes, and power usage. After obtaining a dataset of rooms, the next step was to figure out a method of aligning a corresponding CAD model on each room.

3.3 Iterative Closest Point (ICP) and Assumptions

To accurately align a CAD model with a real-time scan of a physical environment, we must first find a way of minimizing the distance between similar features. This important computer vision problem is solved through a class of algorithm called Iterative Closest Point (ICP). Though there exist many variations [23] of ICP, each one boils down to a simple optimization problem:

1. Start with two sets of points, X and Y. These points should describe two similar objects, 2D or 3D.
2. For each point in set X, calculate which point in set Y is closest and save the correspondence. The total Root Mean Square (RMS) distance between corresponding points is what we are concerned with.
3. Estimate the combination of rotation and translation that will minimize the RMS, and apply the transformation.
4. Return to step 2 and repeat. (*Figure 13*)

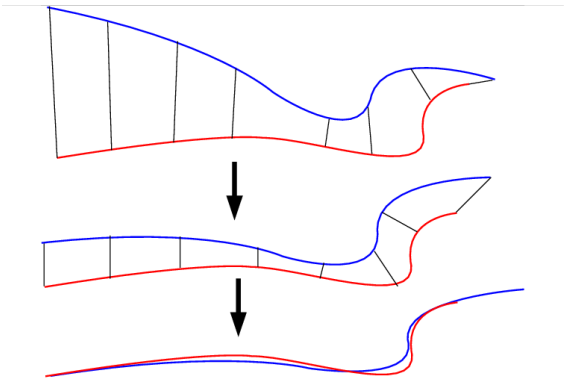


Figure 13: Using ICP to match two similar images of arbitrary curves.

The metric that determines when to stop iterating is usually a target RMS or a preset number of iterations. With small and similar enough point sets, ICP can efficiently run with loose thresholds on iteration count and tight bounds on target RMS (closest as possible to 0). However, in the AEC industry this is frequently not the case. In my experience, the 3D scan of a medium-sized (200 sq. foot) room is usually composed of a few hundred thousand vertices, and while working at IrisVR I frequently came across CAD models that were composed of several million vertices. The sheer number of computations necessary to perform ICP at a reasonable rate to compare these point sets for even a single iteration is well outside the bounds of the limited hardware of the Hololens. However, we can make certain assumptions that greatly simplify the scope of the problem.

1. *First*, let's define the xz plane as corresponding to flat ground, and the y-axis as being the normal to that plane. We know that each room in a building is going to be oriented at a 0 degree angle to the y-axis because of gravity, so there is no need to be concerned about rotation in that direction.

2. *Second*, because we are dealing with 3D scans of spaces in arbitrary states of construction, our ICP comparison should not take into account furniture that may be picked up by a 3D scan performed by the Hololens. We are only interested in the overall shape of the room. In order to reduce the impact of scanning 'noise' produced by furniture or other objects, we need a way to ignore points on the inside of the scan. For that we use a 'convex hull' — the smallest possible shape that encompasses all the scanned points (*Figure 14*).

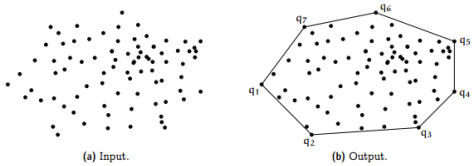


Figure 14: Convex hull generation on a set of points.

The points that end up on the exterior of the hull are what we are really concerned with — the corners and farthest exterior features of a room. As shown in the figure above, a large dataset of points can be condensed to just a few key points.

3. *Third*, we can assume that the general shape of a room is homogenous from the floor to the ceiling because of the necessity of structural integrity. Walls must be load-bearing and ceiling design must take into account the floors above. In short, for a particular vertical section of wall, each point should have the same (x,z) coordinates and only vary in (y), and the ceiling should generally be flat. So, because we don't have to worry about the z-axis and can assume homogeneity in the walls and ceiling of a room, we can convert the problem from 3D to 2D. In addition to that, the assumed homogeneity along the walls allows for 'sparse' scans to serve the same purpose as 'comprehensive' scans.

4. *Fourth*, note how in the standard description of ICP, scale is not taken into account for the transformation. One of the conveniences of 3D models in the AEC industry is that they are always constructed based on real measurements. Scans made by the Hololens are also true-to-scale, so it is easy to find a conversion factor between the two, and thus no need to incorporate scale calculation using a more complex ICP variant. For example, SketchUp is one of the most commonly used 3D modelling software solutions in the AEC industry, and each SketchUp model is marked by its own particular filetype (.skp). When importing a .skp into Unity, the conversion factor is 1:1 by default — i.e. 1 meter in a SketchUp file corresponds to 1 unit of measurement in Unity [28].

With these assumptions made, we can move forward with a considerably more efficient implementation of ICP.

3.4 Overview of Algorithm

Based on the assumptions and data collection described above, the algorithm's structure follows as so:

1. Obtain a 3D CAD model of a room and a corresponding 3D model of a scan made by the Hololens. Import both models into Unity and ensure the scaling is the same, as per [*Assumption 4*].

2. Raycast through each model from each side (four total sides, i.e. -x,-z; -x,+z; +x,-z; +x;+z), at a predefined resolution (i.e. 5 rays/meter, 10 rays/meter, etc) and save the first collision point of each ray. Increasing the resolution will result in a more detailed dataset, but will also take longer to raycast. In this manner, we map out the perimeter of each room. This helps eliminate scanning 'noise' from furniture and other objects.

3. Generate a convex hull from these perimeter points. Because the scan of the room may be sparse, the first collision point of a ray could be incident on something other than an outer wall. Only taking into account the outermost points mediates this.

4. Perform ICP on the two convex hull point sets.

5. Quantify error in alignment by calculating the RMS distance between all vertices of the scanned area and the corresponding closest vertices on the overlaid CAD model.

3.4.1 Raycasting the CAD Model and 3D Room Scan

Raycasting is the technique of extending a geometric ray outwards from a point in a predefined direction for a predefined distance and analyzing what that ray collides with. It is primarily used in computer rendering, but for the purposes of this project we will use it to trace the shape of our CAD model and 3D room scan. Unity makes this easy with built-in granular control of raycasting [29]. After first computing the dimensions of the room using Unity's Bounds [30] object — Bounds computes the smallest box possible to fit all vertices of the 3D model — we set up four raycasting planes as described in Section 2.4 that correspond to these Bounds. Then, we raycast through the model from each side at a predefined resolution, saving the first collision point of each ray. However, when saving these points note that we ignore the y-coordinate as explained in Section 2.3. So, the resulting dataset of saved points has points of the form (x, 0, z). We are essentially finding the 2D 'shadow' that a 3D model casts on the *xz* plane (*Figure 15*).

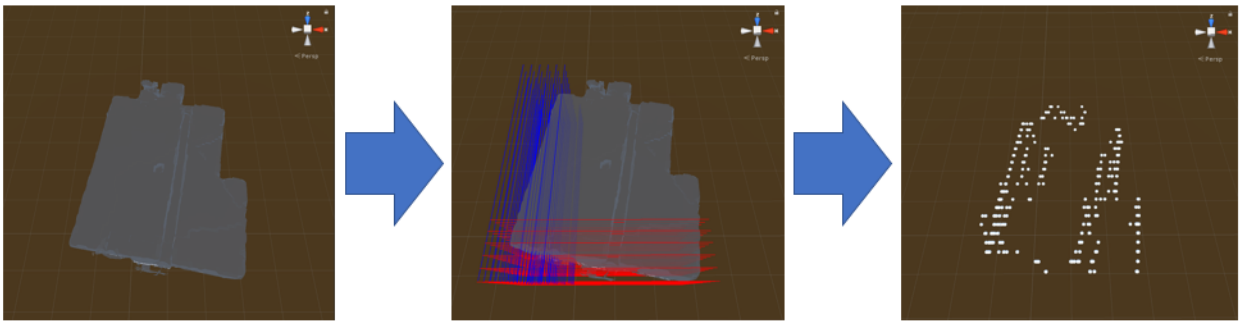


Figure 15: Raycasting a 3D scan of my dorm room to produce a set of 2D points. The resolution here is 4 rays/meter. Note how in the resulting dataset, a significant number of points exist within the perimeter of the room. This is due to the walls being occluded by furniture, and will be mediated by the creation of a convex hull in the next step. The grid squares in the background are 1 meter x 1 meter in dimensions.

However, scan noise and sparseness can generate collisions away from the perimeter. We use a convex hull to eliminate these unnecessary points. For this thesis, I chose to use a raycast resolution of 4 rays/meter for the generation of a point cloud. This threshold was chosen after testing many different values and not seeing a noticeable increase in captured detail past a certain resolution. In future work, this can be more finely tuned through further testing.

3.4.2 Calculating a Convex Hull for Each Set of Points

After generating a point set from raycasting the model and scan, the next step is to account for scan 'noise'. We calculate a convex hull (Figure 16) from the point set by using Jarvis' Algorithm [31]. This is particularly important for noisy and sparse scans, as the first collision point of a raycast can be furniture rather than an exterior wall.

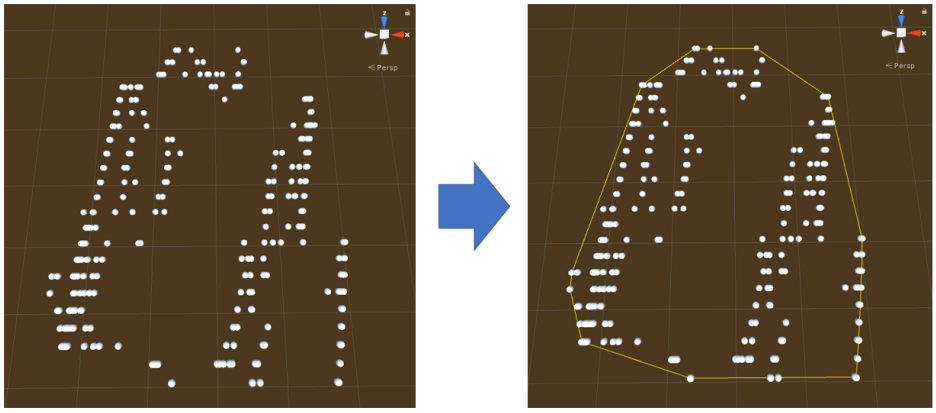


Figure 16: Generating a convex hull from the point set. The points comprising the hull are connected with yellow line segments. Note that the 3D scan used for the generation of this hull was comprehensive in that there weren't any gaps in the scan.

One of the most useful advantages of a convex hull in combination with [*Assumption 3*] listed above is that to generate a hull that captures the perimeter of the room you only need to scan one small segment of each vertical section of wall. Note that the convex hull does eliminate some detail in the scan, particularly in concave sections of the room (i.e. window indents, curved walls, etc). This can be mediated with the use of a concave hull. It is similar to the convex hull in that it captures the entire point set using only exterior points, but its definition of 'exterior' is more flexible in that it can more closely trace the shape of a room (*Figure 17*).

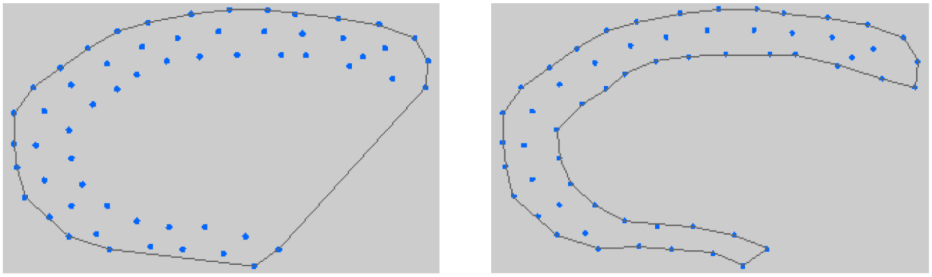


Figure 17: A convex hull vs a concave hull. Note how the concave hull more closely follows the shape of the dataset.

I adapted an already existing implementation [32] of generating concave hulls in Unity to show how the concave hull of a dataset can differ from the convex hull (*Figure 18*).

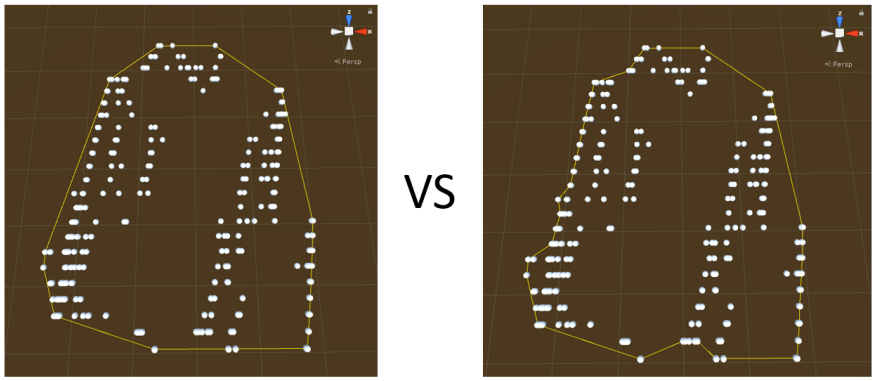


Figure 18: The convex and concave hulls side by side. The difference between the two is not significant and is primarily due to scan noise and furniture occlusion.

3.4.3 Performing ICP on the Hulls for Alignment

After raycasting both models and generating convex hulls, we use ICP to find the correct transformation for matching the hull of the CAD model onto that of the 3D room scan. One of the most challenging problems in ICP is figuring out which points correspond with one another. We can solve that easily through a byproduct of convex hull — after calculating a convex hull of a point set, the saved points in the hull are ordered clockwise in the resulting array. So, we can generate correspondences by matching array indices in both models (i.e. CAD[0] could correspond to SCAN[0], CAD[1] corresponds to SCAN[1], etc). Every combination of clockwise correspondences is tried, ICP is performed on the convex hull, and then the error in overlay is calculated. Note that this error calculation does not compare the entirety of both 3D models to each other, but instead only compares the points present in each convex hull. This allows the algorithm to run considerably more quickly than if we used the latter. The option with the lowest error is then chosen as the optimal overlay (*Figure 19*).

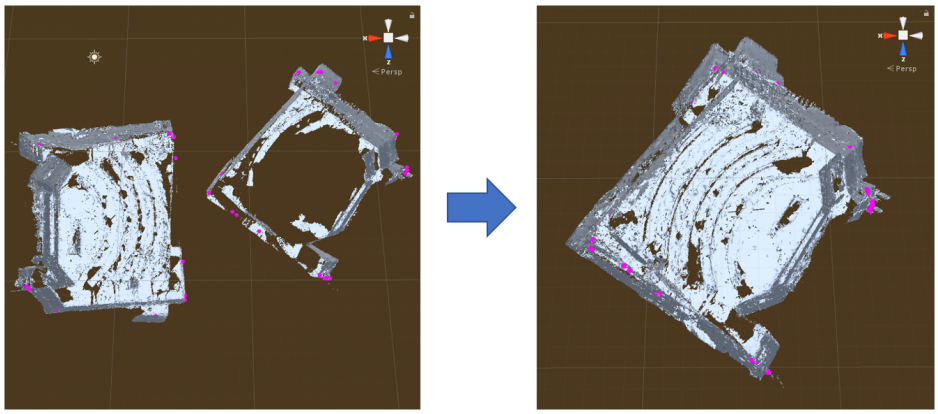


Figure 19: Before and after ICP is performed. It's difficult to tell due to the nature of the 3D scan, but the second image has the two models overlaid on each other.

3.4.4 Calculating Error in Overlay

After calculating the relevant transform from the previous step, we apply it to the full CAD model from before to overlay it on the 3D environment scan from the Hololens. While it was easy to qualitatively analyze the accuracy of the overlay, figuring out an accurate quantitative metric proved difficult. I initially thought of calculating the RMS distance between portions of the 3D scan and the CAD model by:

1. Beginning with the overlay of the 3D model on the scanned environment, compute the centroid of the 3D scanned points.
2. Raycast outwards from that centroid in a sphere with a predefined resolution and record the farthest points of impact on both the CAD model and scanned environment.
3. Compute the distance between each pair of impact points and save the value.
4. Repeat for every raycast and compute a total RMS distance. The final value will be an average offset distance between the 2 models, i.e. ".15 meters". (*Figure 20*)

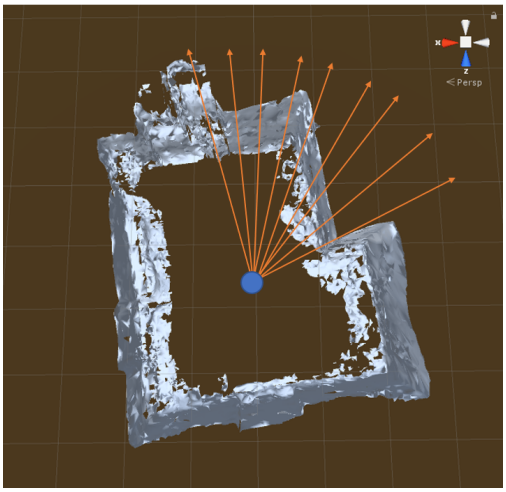


Figure 20: Raycasting outwards from a central point. Notice how closer surfaces have a higher density of collisions.

However, a problem arises. Raycasting from a central point gives a greater weight to points closer to the centroid in the RMS calculation, due to there being a lower density of collisions the farther away you get from the center. To resolve that, my first idea was to weight the RMS by collision distance from the centroid, but my advisor, Prof. Jarosz, came up with a better idea.

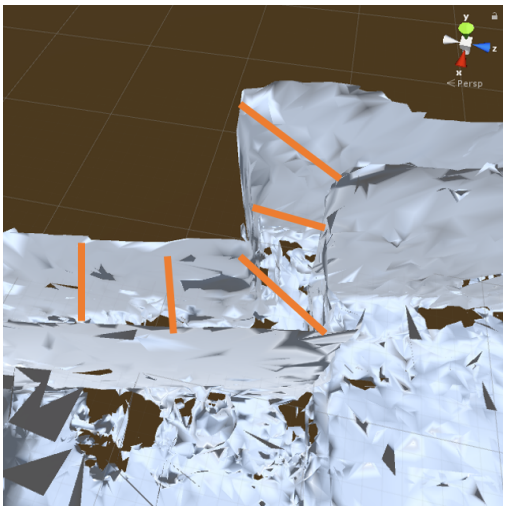


Figure 21: An overlay of a sparse scan (left) on a comprehensive scan (right). The orange lines show the closest points on the full model to points on the sparse scan.

Instead, we can iterate through all points of the 3D scan and for each one calculate which point on the CAD model is closest to it, then save the distance between those two (*Figure 21*). After iterating through every point, we compute an RMS from those values to get an average offset. This solves the issue of accidental weighting. However, iteration across every point for both models takes $O(n^2)$ time and caused visible slowdown even with smaller scans. If the error in offset was used as an active metric in an application then we would need to mediate this slowdown. Using octrees [33] to segment the 3D space would allow for significantly faster search, as we could choose to only query points in the same general area rather than having to test every point in the model.

4 Results, Analysis, and Discussion

4.1 Dataset of Rooms, Complexity Chart

In order to test the robustness of the algorithm, I needed to gather a dataset of room scans and their corresponding CAD models. A problem immediately arose — I did not have access to the models for the spaces that I was scanning. Dartmouth is an old institution and most buildings were constructed before CAD models were popularized for use in construction. I had the option to measure the dimensions of relevant rooms and generate the models on my own, but it did not make sense to spend time doing that when a true-to-scale model generation solution was already in front of me: the Hololens itself. With 1,200 triangles saved per cubic meter, we get a little over half a centimeter of scan accuracy. The vast majority of rooms do not have small enough identifiable features where inaccuracies at this scale show up in the convex hull perimeter, so we can assume that a comprehensive 3D scan can replace the model in our analysis. The impact of furniture on the alignment immediately comes to mind, but our choice to use convex hulls combined with the lack of lost detail in hulls generated from sparse noisy scans shows that the Hololens scans can serve as replacements for premade CAD models. Qualitative analysis also shows promising results (*Figure 22*). However, this assumption is worth further testing with actual models and comprehensive scans compared side by side.

Figure 22: A study room and the resulting comprehensive 3D scan with the Hololens. Note how all major features are accurately saved.

When choosing rooms to scan for testing the algorithm, I drew inspiration from Michael Firman's paper [35] on the state of RGB-D datasets. He defines the 'realism' of a scene using a qualitative three-part scale (*Figure 23*) that ranges from orderly laboratory scenarios to more chaotic real-world scenes.

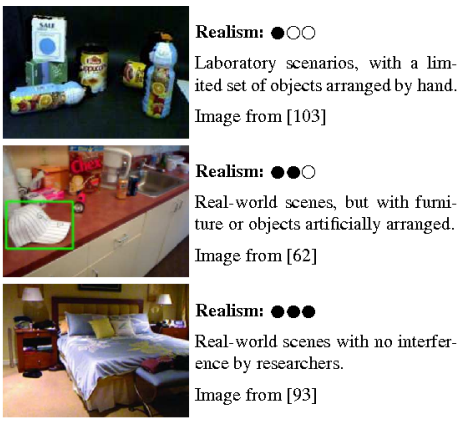


Figure 23: Firman's qualitative scale for judging the realism of a scene.

I created a two-part scale to judge rooms by how complex they were. I took into account qualitative factors like room size, amount of furniture along the walls, number of windows, curvature in walls, elevation changes within the room, etc. For example, I used a 3D scan of a small study room in a dormitory as one of the simpler data points, and a scan of a messy dorm room as one of the more complex data points (*Figure 24*). Note that this scale is entirely subjective and a more quantitative scale for rating rooms would be useful for further work. However, there are so many variables involved that coming up with a useful baseline metric would require a significant amount of data.

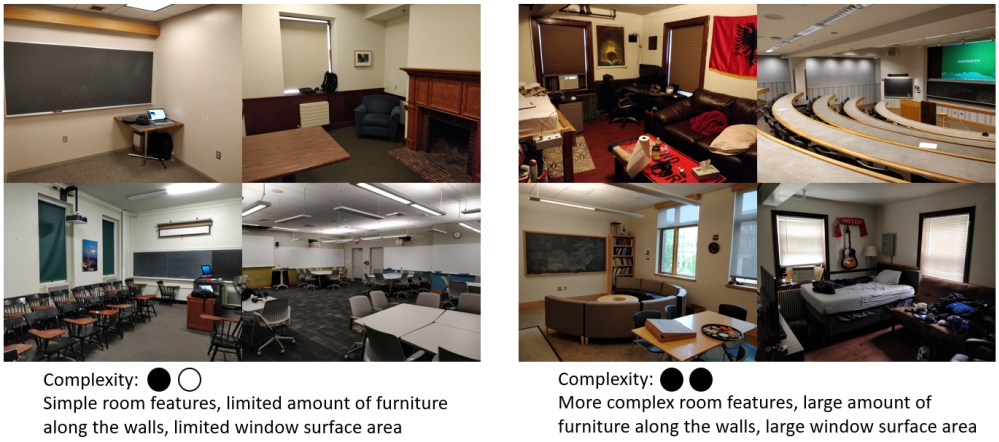


Figure 24: My complexity scale for rooms, included with a few photos of rooms that fit each side of the scale.

I scanned ten different rooms for testing and have included the dataset in Appendix 7.3. Each room has a 'sparse' scan and a 'comprehensive' scan associated with it. The sparse scan involves walking around a room with the Hololens actively scanning, but only collecting scan data of the wall at head height. The result is a fraction of the comprehensive scan with two main differences: most of the floor and ceiling is missing, and there is less detail in the wall scan. This reflects real-world use cases as it would be difficult to teach a new user of the Hololens how to perform a comprehensive scan. In addition, the nature of the convex hull algorithm makes it easy to ignore any noise generated from furniture that does not border the edges of the room, along with any noise from less precise scans of the outer edge (*Figure 25*). The comprehensive scan involves slowly walking around the room and scanning surfaces from many different heights, positioning the Hololens closer to objects to capture a greater level of detail. This mimics the increased detail in a user-generated CAD model.

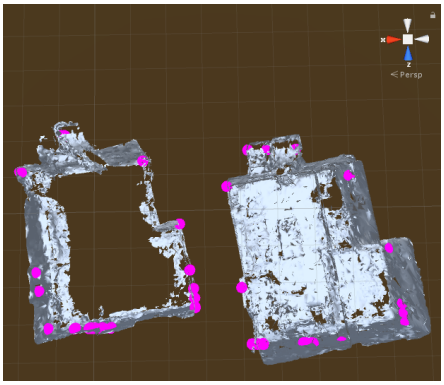


Figure 25: Difference between raycasting a comprehensive scan and a sparse scan. The pink dots signify vertices present in the convex hull. Notice how the convex hull shape and structure differs very little between the two scans, suggesting that sparse scans capture the perimeter of the room just as well as comprehensive scans.

4.2 Effectiveness of Alignment, Testing Metrics

To judge the effectiveness of alignment, we use the method outlined in section 2.4.4. Average offset was compared against room complexity, scan sparseness (where sparseness is computed as the ratio of the number of vertices in a sparse span as compared to the number of vertices in a comprehensive scan), and room size. In addition, time to perform ICP and calculate error in alignment was recorded and compared to room size.

For each room tested, many factors were recorded: room complexity, number of vertices per model, sparseness, time taken to perform ICP, time taken to calculate error, and the model offset. See Appendix 7.3 for photos of each room, and (*Figure 26*) for the recorded data.

Room	Complexity	Vertices in Full Model	Sparseness	Time to Perform ICP (s)	Time to Calculate Error (s)	Model Offset in Overlay (m)
1	Complex	134330	.4919	27.7886	636.2083	.3608
2	Simple	105464	.6117	44.7638	572.9205	.0835
3	Simple	53599	.4121	37.5486	171.4891	.0750
4	Complex	30373	.4086	10.5308	48.5491	.0310
5	Complex	104506	.2964	38.8457	335.1049	.4181
6	Simple	45507	.4601	23.3644	117.0221	.0645
7	Complex	50275	.6366	13.6432	136.5160	.0885
8	Simple	11745	.6146	15.1141	28.4877	.6322
9	Simple	28571	.5346	11.6417	60.7838	.3655
10	Complex	25593	.6552	12.5269	54.7821	.0486

Figure 26: Table of recorded data from processing each room through the algorithm.

4.2.1 How does room complexity impact alignment?

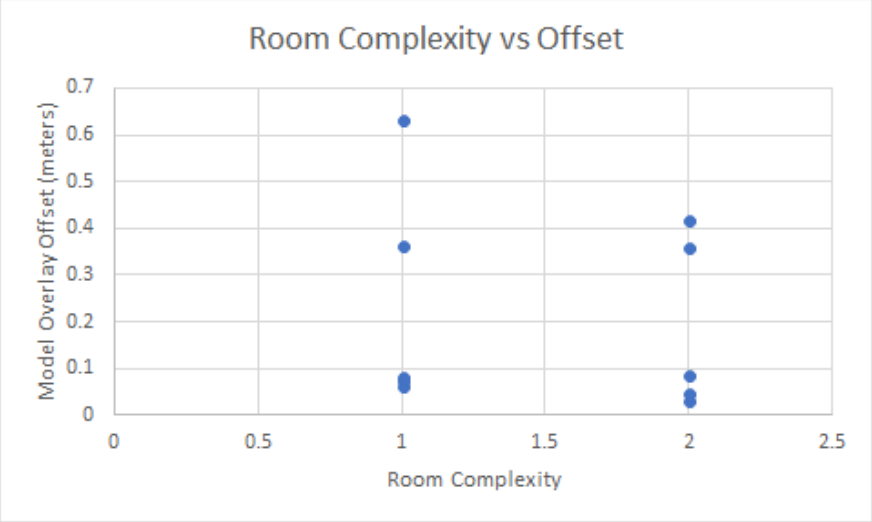


Figure 27: Graphing room complexity vs model offset. A complexity of '1' corresponds to simple rooms, and a complexity of '2' to complex rooms.

The data shows no clear trend in terms of room complexity and alignment. However, the qualitative nature of the 'complexity' scale makes it difficult to assign to a room and thus difficult to draw quantitative results from. A correlation could be found in the future with more data and a more precise scale.

It is also worth discussing the highest offset value in the 'simple' category. Model 8 was the smallest one in the dataset and the room itself was relatively featureless and symmetrical. This resulted in its convex hull being less detailed in comparison to the other models, leading to a less precise final alignment. For future iterations of the algorithm, the case of symmetry and small model size must be tackled to prevent such outliers from existing.

4.2.2 How does scan sparseness impact alignment?

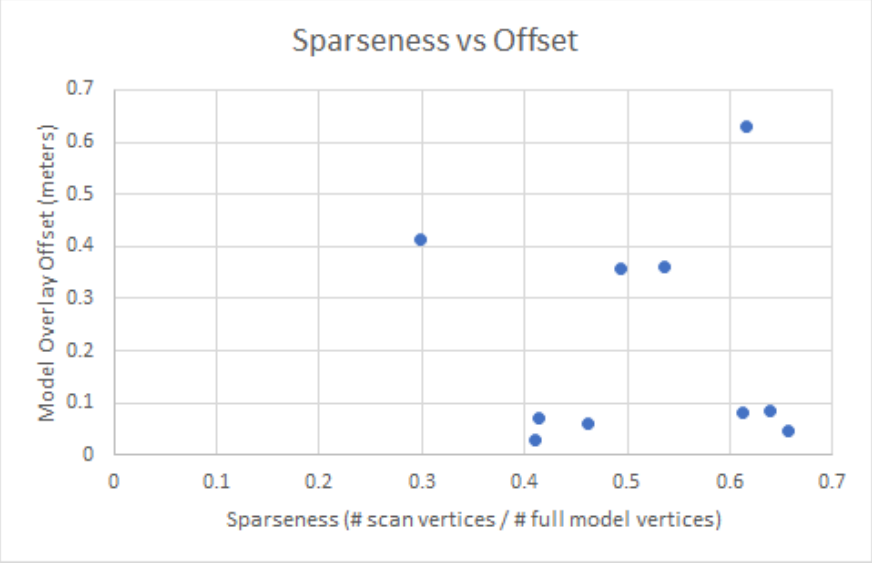


Figure 28: Graphing sparseness of scan vs model offset.

The data does not show any significant trend in sparseness of scan and alignment. This evidence in addition to the nature of convex hull affirms that we do not need much of the 3D detail in a room in order to perform effective alignment. Only the overall shape is required. I postulate that values as low as 10% or 5% sparseness would give a model offset of the same magnitude as data points in the 40% or higher range, but this would need to be tested.

4.2.3 How does room size impact alignment?

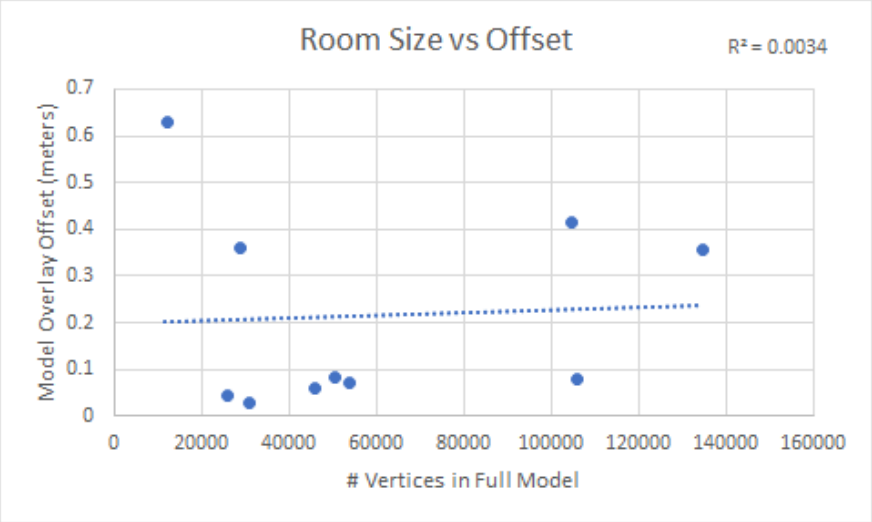


Figure 29: Graphing room size vs model offset.

From first glance, there appears to be no correlation in room size and alignment. However, let's assume that the smallest model scan is an outlier and recalculate the trendline without it. The result is a much clearer correlation:

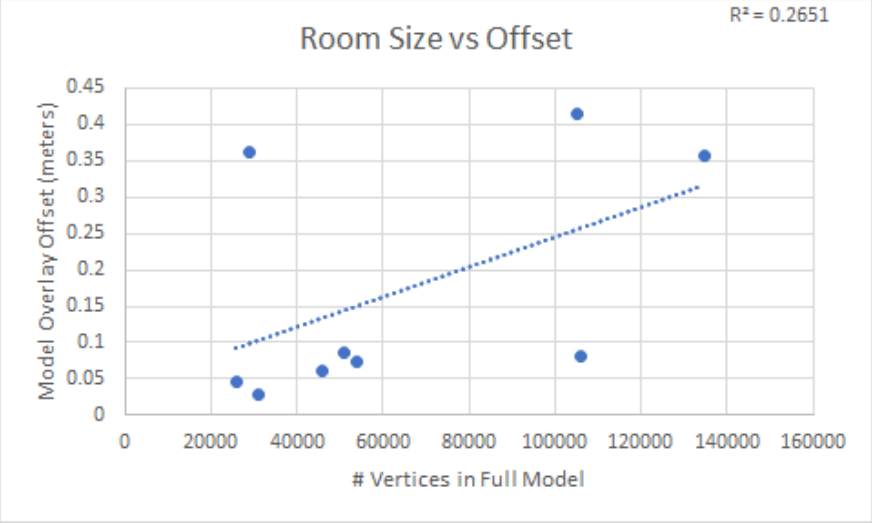


Figure 30: Graphing room size vs model offset with an outlier removed.

The room in question (model 8) was discussed briefly in section 3.2.1 as a potential outlier. The room's symmetry and small size likely led to its relatively high error, and it is worth testing more rooms of this size or this degree of symmetry in order to affirm whether or not this is truly an outlier case.

4.2.4 How does room size impact time taken to perform ICP and calculate error?

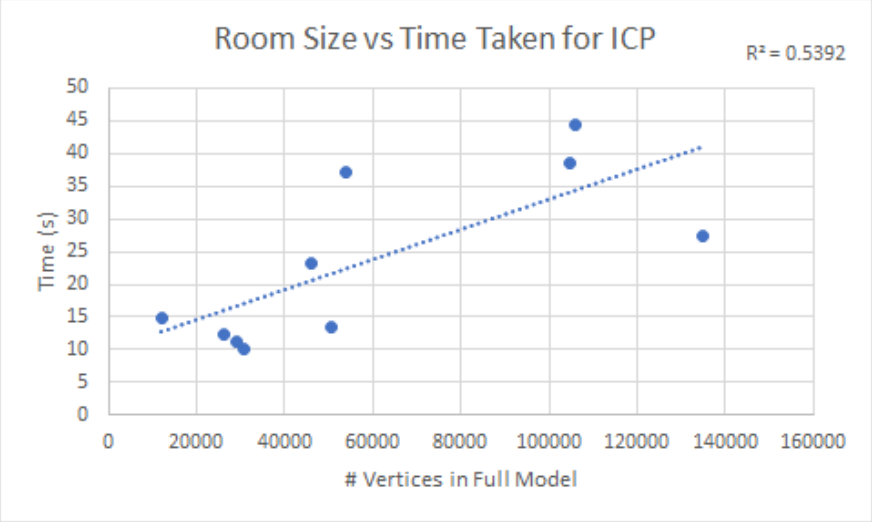


Figure 31: Graphing room size vs time taken to perform ICP.

There is a moderately strong linear relationship between room size and time taken to compute ICP. In addition, even for very large rooms the algorithm does not take more than 45 seconds to compute the optimal overlay. This is reasonable for use on-device.

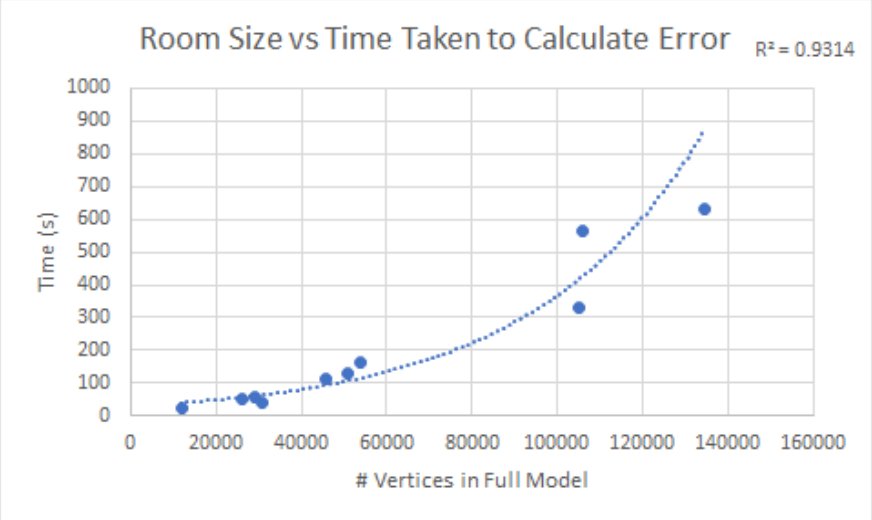


Figure 32: Graphing room size vs time taken to calculate error in overlay.

There is a strong exponential relationship between model size and time taken to compute error. For larger models it can take upwards of ten minutes to perform the computation, which is absolutely unacceptable for a product designed to be used on-site. Using octrees would drastically improve this time complexity.

4.3 Successes and Shortcomings of this Method

Results in average offset from ICP show promise for both sparse scans and more complex rooms. On average, an overlay error of .2167 meters was obtained using this algorithm. This is nowhere near perfect, and with the Hololens being able to map spaces to an accuracy of less than a centimeter there is still much more testing and iteration to be done before we reach the limits of this method. However, for a pure-software method without any physical peripherals or user intervention this is already a large step forward. Overall, I would consider the results of this research to be successful and promising.

5 Conclusion and future work

5.1 Current State of the Industry

Microsoft and DAQRI introduced high-grade AR to the consumer world, but it still has a ways to go before it is adopted by any significant proportion of AEC professionals. Other promising contenders in the space include Magic Leap [36] (*Figure 33*) and Meta [37] (*Figure 34*), the former being backed by over two billion dollars of investor funding [38] and being one of the most anticipated product releases in the industry. The device is rumored to cost a fraction of the price of the Hololens and contain the same level of precision in environment reconstruction.

Figure 33: Magic Leap's developer kit, shipping later this year.

Figure 34: Meta's developer kit, released last fall.

I believe that augmented reality is poised to replace the smartphone as the average person's primary way of interfacing with the digital world. While the price point and convenience isn't quite there for the average consumer, there are already clear use cases of AR in the professional world. DAQRI presented an interesting proof-of-concept in their partnership app with Mortensen, but the device and hardware that the app runs on was too expensive and niche to reach a large group of people. The Hololens presents a platform for such an app to become popular, and other hardware from companies like Magic Leap and Meta presents additional opportunities for growth.

5.2 Where does my work fit in, and how can it be expanded on?

The content of this thesis can first and foremost be used as the foundation of a scale model to real environment 'overlay' feature within an augmented reality application for the AEC industry. I am confident that such a feature would be essential for a future AR app to be widely adopted, and hope to include it in future iterations of my work at IrisVR. However, this thesis is just a foundation — further testing and iterating on the concept is required before this work can exist in a product. Initial results show promise though, and I believe that with more development this could be shaped into a reliable and essential product for AEC professionals.

- The speed of error calculation in overlay can be improved significantly by using octrees, as mentioned in section 2.4.4. This is vital if overlay offset ends up becoming an active metric to gauge how effective an alignment is, perhaps to recompute an overlay if the user leaves and re-enters an environment. Error in offset was by far the most computationally intensive part of the algorithm.
- The current error metric is weak in situations with highly symmetrical rooms. This is a rare case as most rooms have some sort of asymmetry in window placement, door location, etc. However, to account for this a future application could present multiple possible overlays to the user and have them choose the correct one. Ideally, a solution would not require manual intervention though.
- A user interface for initializing and managing the overlay from on-device is necessary.
- Other AR headsets with similar hardware to Hololens — in particular, Magic Leap's future prototype and Meta's current flagship product — present opportunities for this algorithm to reach a larger audience due to their lower price point compared to Hololens.
- The developed ICP algorithm can be tuned to be faster and more accurate. This will require significantly more data to test with and likely the generation of a better heuristic for moving through the iterations of ICP.
- Expanding the project's scope to handle multi-room models. Using a similar method to Stanford's paper on 3D Semantic Parsing of Large Indoor Spaces [39], we could segment a building into separate rooms and then find the best-fit match for the 3D room scan.

6 Acknowledgements

I would like to thank the folks at IrisVR for inspiring me to work on this idea, and I hope my work proves useful when I start my job as a software engineer there. I would like to thank my Writing Professor, Deanne Harper, whose constant feedback, encouragement, and enthusiasm helped keep me on track to complete a thesis I could be proud of. I would also like to thank Professor Wojciech Jarosz for graciously accepting to be my advisor for this project. His guidance and wisdom helped steer me towards using an ICP algorithm and figuring out a more useful metric for measuring error in alignment. Finally, I would like to thank my parents and brother for being my bouncing board for ideas throughout this process.

7 Appendices

Included here are: a link to the repository where the project is stored, the sources for figures, and extra images of the dataset I used for testing.

7.1 Code Repository

https://github.com/gjrgj/thesis

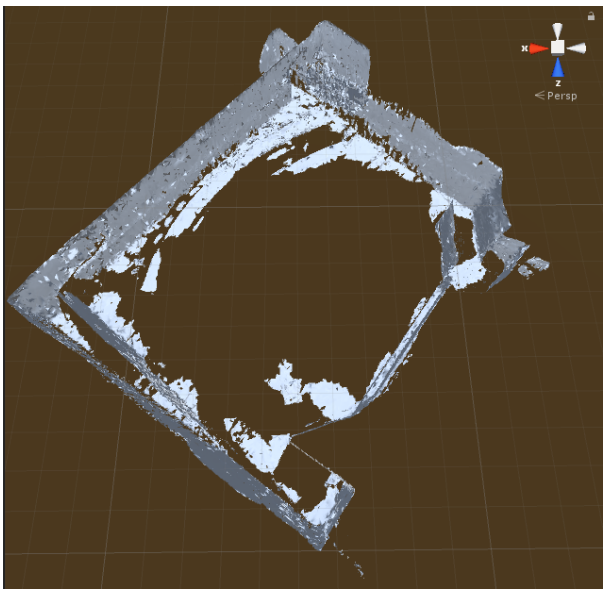
7.2 Sources for All Figures

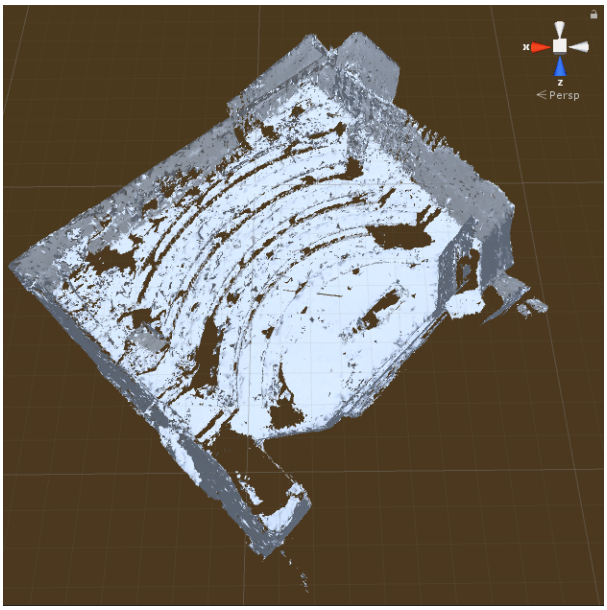
1. https://thebimhub.com/2014/11/28/how-traditional-aec-processes-and-bim-level-2-rein/
2. https://www.bdcnetwork.com/MortensonARapp
3. https://techli.com/2011/07/daqri-using-qr-codes-to-create-experiences/
4. https://hackaday.io/project/5077/logs?sort=oldest&page=3
5. https://msdn.microsoft.com/en-us/library/hh438998.aspx
6. https://medium.com/@philspitler_29845/the-art-of-using-the-microsoft-kinect-in-visual-effects-production-ea148ec79acd
7. http://www.businessinsider.com/daqri-smart-helmet-2016-3
8. https://www.straight.com/life/1024876/why-metro-vancouver-has-become-world-leader-virtual-and-augmented-reality
9. https://arxiv.org/pdf/1706.08096.pdf
10. https://skarredghost.com/2017/03/20/sketchup-viewer-for-hololens-review/
11. https://www.youtube.com/watch?v=U9t6Osl1Lbc
12. https://forums.hololens.com/discussion/6340/miniature-map
13. https://www.researchgate.net/figure/Illustration-of-the-iterative-closest-point-method-to-align-two-lines-A-set-of-points-is_fig5_269703385
14. https://medium.com/@harshitsikchi/convex-hulls-explained-baab662c4e94
15. Generated in Unity.
16. Generated in Unity.
17. http://ubicomp.algoritmi.uminho.pt/local/concavehull.html
18. Generated in Unity.
19. Generated in Unity.
20. Generated in Unity.
21. Generated in Unity.
22. Photo taken by me, scan generated using Hololens Developer Tools.
23. https://pdfs.semanticscholar.org/1077/856e4479ffbc3a99d187fb14543da318f845.pdf
24. Photos taken by me.

25. Generated in Unity.
26. Generated in Excel.
27. Generated in Excel.
28. Generated in Excel.
29. Generated in Excel.
30. Generated in Excel.
31. Generated in Excel.
32. Generated in Excel.
33. https://gizmodo.com/this-is-magic-leap-s-first-headset-1821461007
34. https://pages.metavision.com/meta2-demo-sign-up

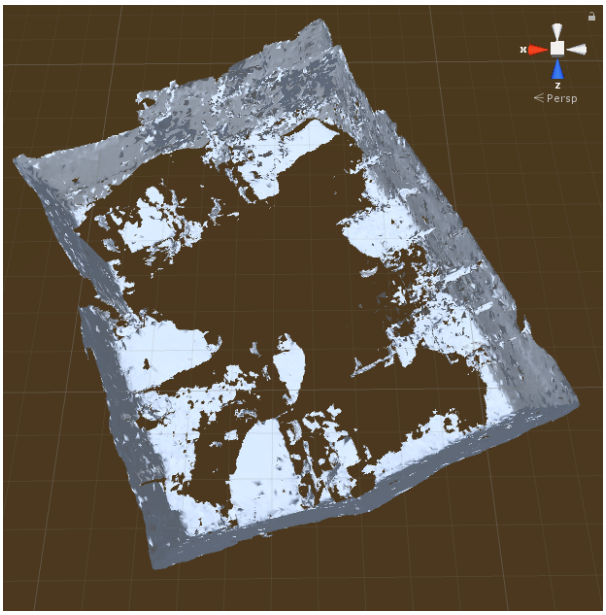
7.3 Dataset of Rooms

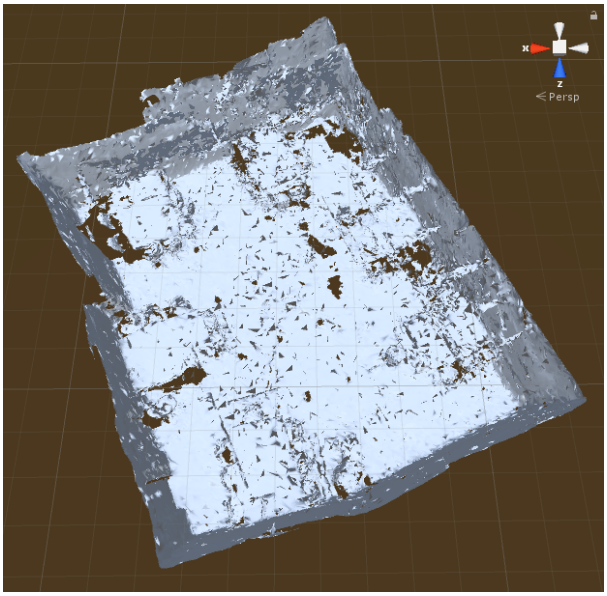
1. Berry Library Basement Auditorium



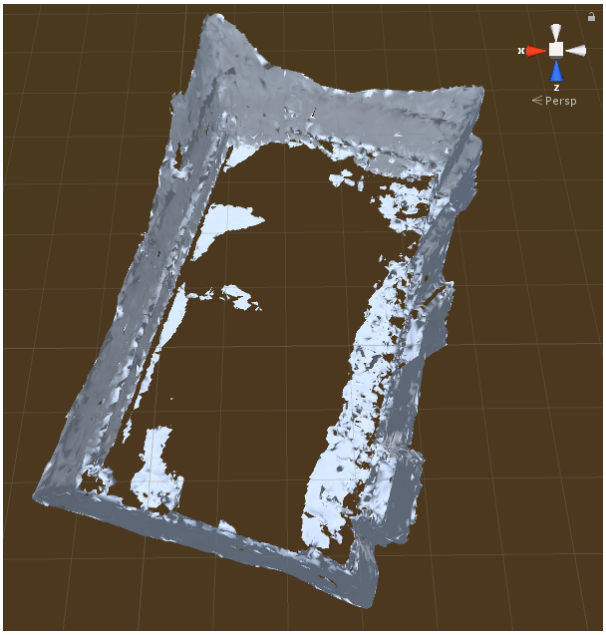


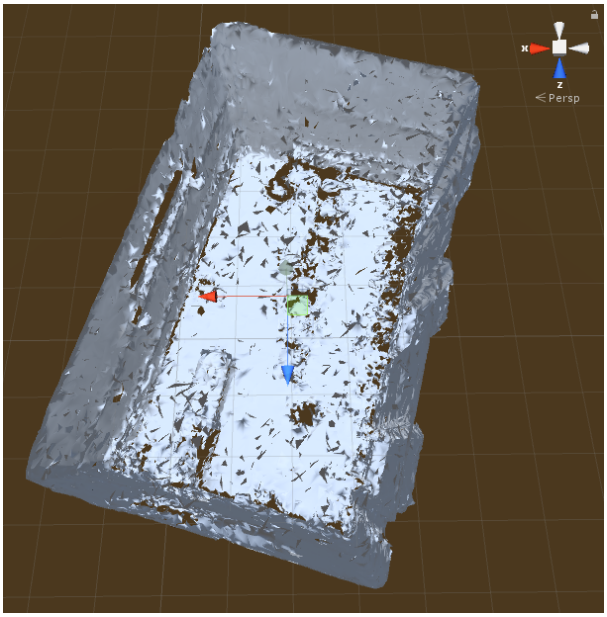
2. Berry Library Innovation Classroom



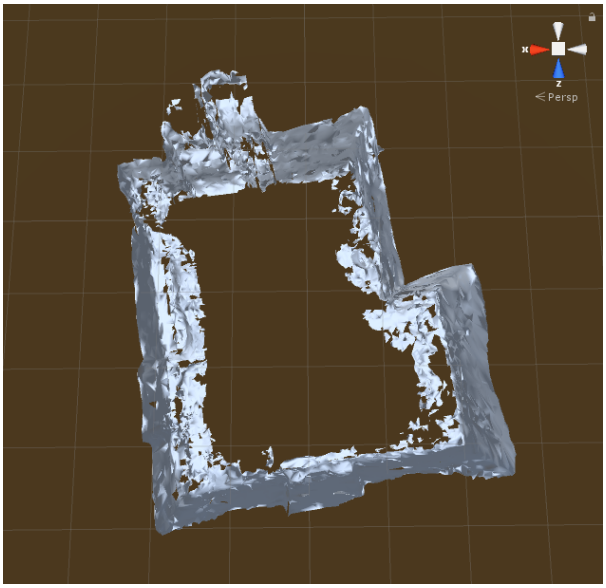


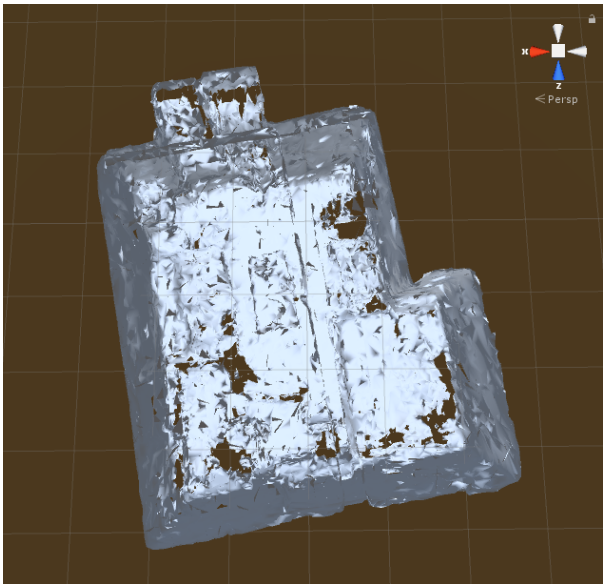
3. Dartmouth Hall Classroom



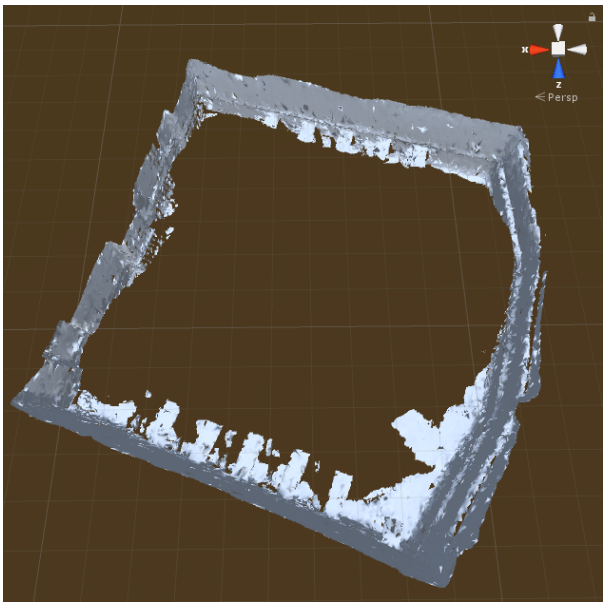


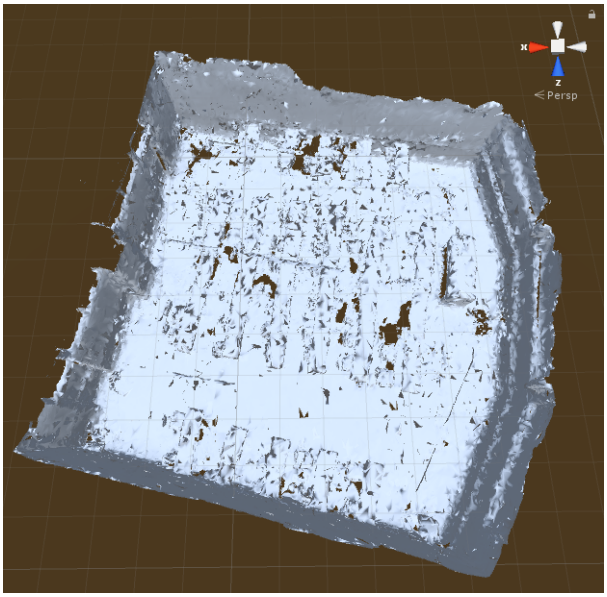
4. Dorm room 1



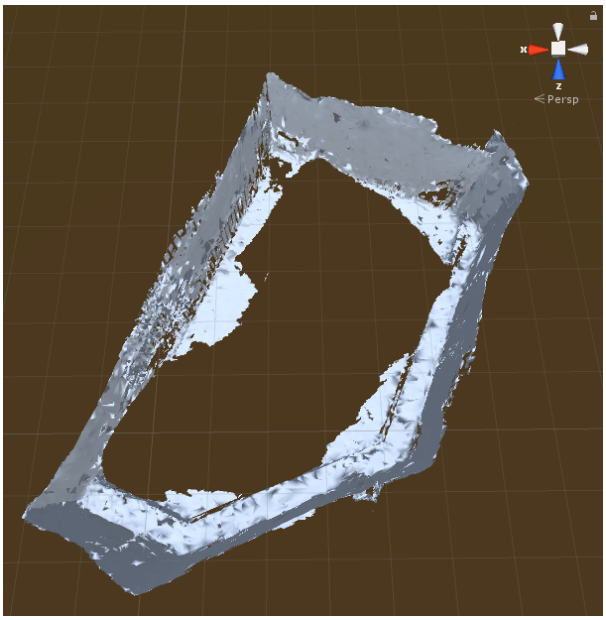


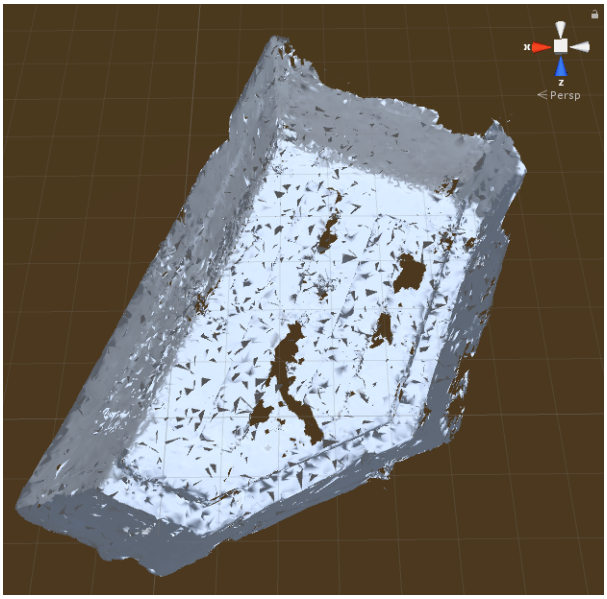
5. Kemeny Classroom



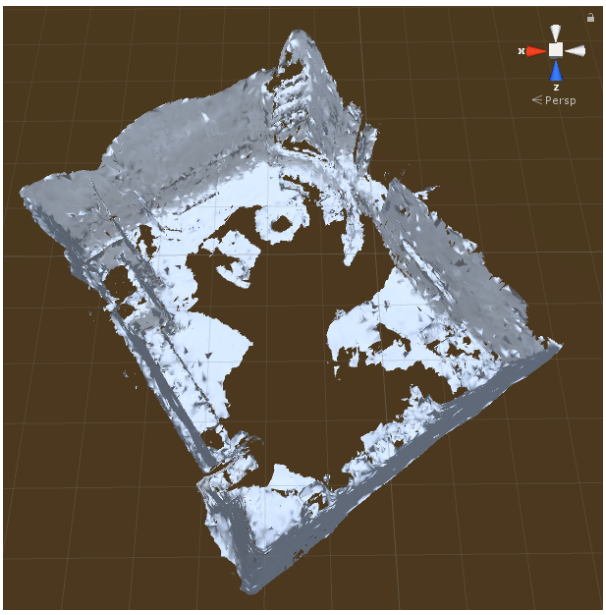


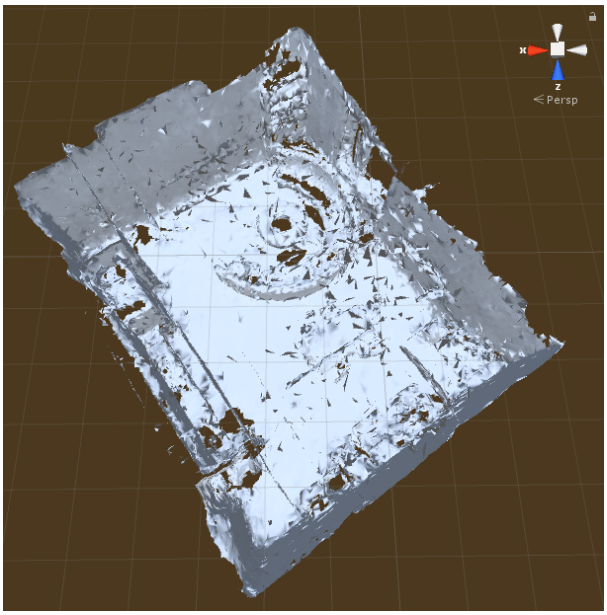
6. Kemeny Conference Room



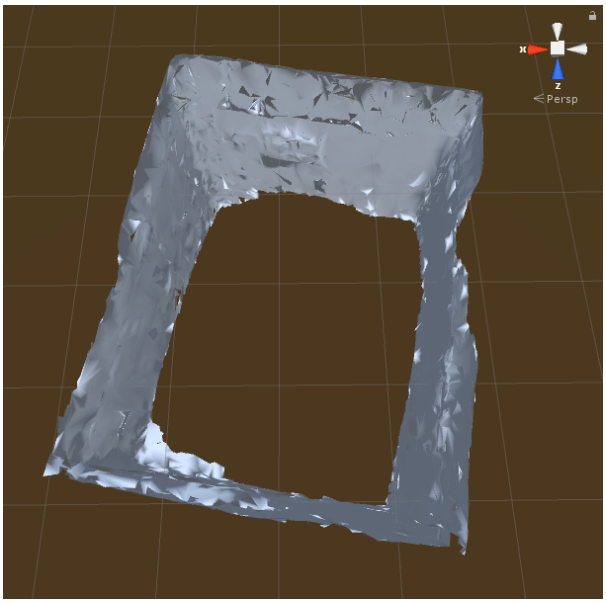


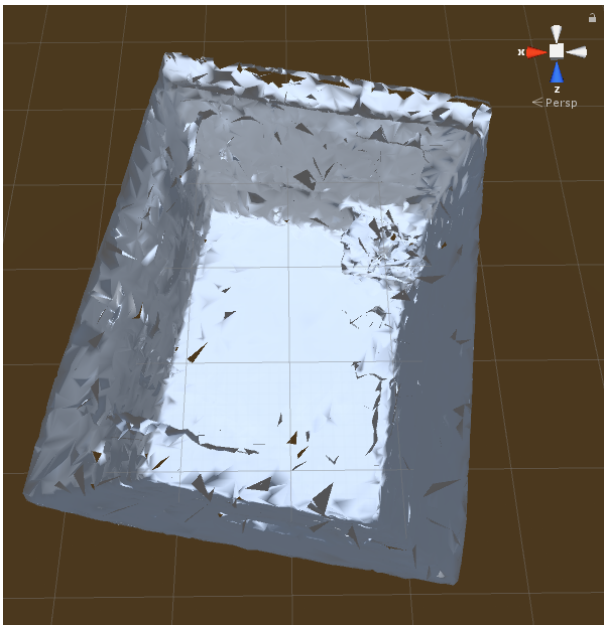
7. Kemeny Graduate Commons



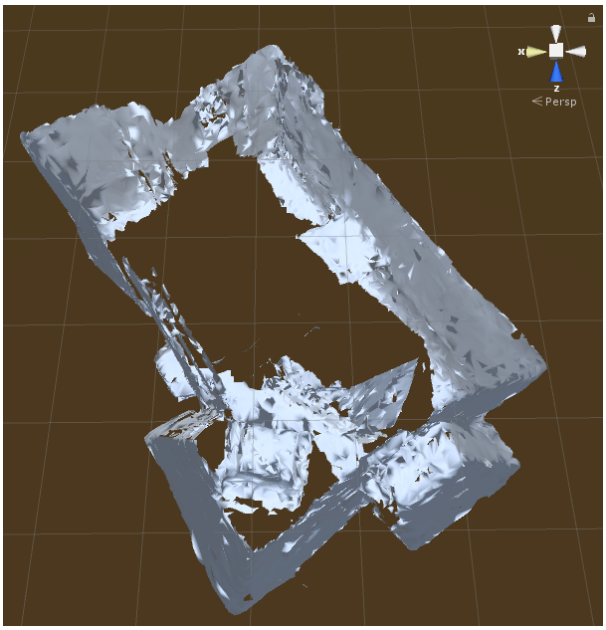


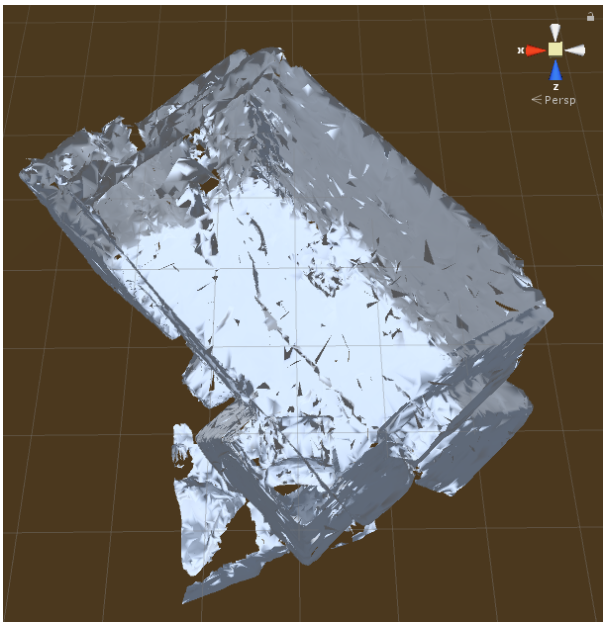
8. Morton Study Room



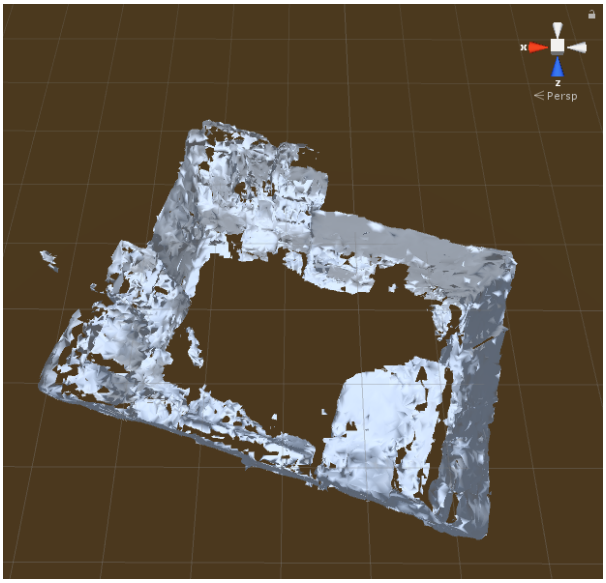


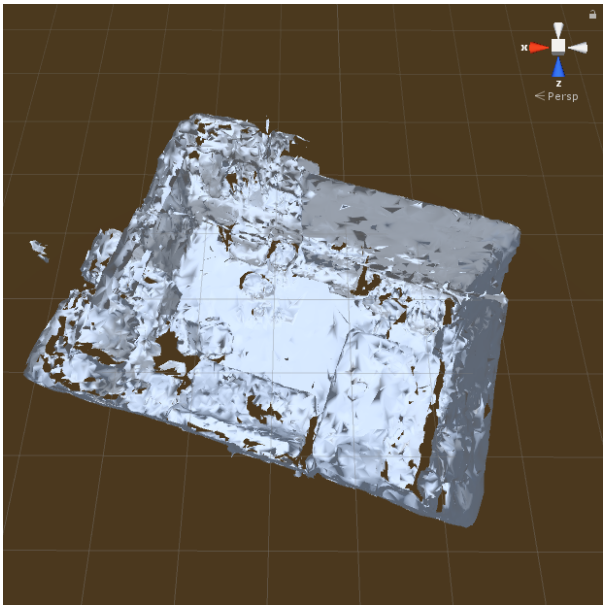
9. New Hamp Study Room





10. Dorm Room 2





8 References

1. Corke, Greg. "Virtual Reality for architecture: a beginner?s guide." Aecmag.com. 16 Feb. 2017. Web. 29 Apr. 2018. https://www.aecmag.com/59-features/1166-virtual-reality-for-architecture-a-beginner-s-guide

2. Akio Moriwaki. "How Traditional AEC Processes and BIM Level 2 Reinforce Silos." The BIM Hub. 28 Nov. 2014. Web. 30 Apr. 2018. https://thebimhub.com/2014/11/28/how-traditional-aec-processes-and-bim-level-2-rein/

3. Daniel Rubino. "These are the full hardware specifications of the Microsoft HoloLens." Windows Central. 29 Feb. 2016. Web. 30 Apr. 2018. https://www.windowscentral.com/hololens-hardware-specs

4. N.a. "Mortenson Creates First-of-Its-Kind Augmented Reality App for Construction Visualization." Mortenson.com. 19 Oct. 2017. Web. 30 Apr. 2018. http://www.mortenson.com/company/news-and-insights/2017/mortenson-creates-first-of-its-kind-augmented-reality-app-for-construction-visualization

5. Marc Stiles. "Ahead of the crane: Mortenson Construction augments reality on UW project (Video) - Puget Sound Business Journal." Puget Sound Business Journal. 22 Jun. 2017. Web. 30 Apr. 2018. https://www.bizjournals.com/seattle/news/2017/06/22/mortenson-construction-virtual-reality-design.html

6. Daqri. "Building the Worksite of the Future ' Augmenting Humanity ' Medium." Medium. 15 Dec. 2016. Web. 30 Apr. 2018. https://medium.com/daqri-blog/building-the-worksite-of-the-future-b6db32873e32

7. Sarah Perez. "Wayfair?s Android app now lets you shop for furniture using augmented reality." TechCrunch. 20 Mar. 2018. Web. 30 Apr. 2018. http://social.techcrunch.com/2018/03/20/wayfairs-android-app-now-lets-you-shop-for-furniture-using-augmented-reality/

8. N.a. "VuMark." Library.vuforia.com. n.d. Web. 30 Apr. 2018. https://library.vuforia.com/articles/Training/VuMark

9. Rafe Needleman. "Daqri connects QR codes to augmented reality." CNET. 23 Feb. 2011. Web. 30 Apr. 2018. https://www.cnet.com/news/daqri-connects-qr-codes-to-augmented-reality/

10. Techli. "Daqri: Using QR Codes To Create Experiences." Techli. 13 Jul. 2011. Web. 30 Apr. 2018. https://techli.com/2011/07/daqri-using-qr-codes-to-create-experiences/

11. Daqri. "DAQRI Smart Helmet: A Deep Dive ' DAQRI ' Medium." Medium. 15 Dec. 2016. Web. 1 May 2018. https://medium.com/@DAQRI_Media/daqri-smart-helmet-a-deep-dive-b384f5997537

12. Mortenson. "Mortenson Creates First-of-Its-Kind Augmented Reality App for Construction Visualization." Prnewswire.com. 18 Jul. 2017. Web. 1 May 2018. https://www.prnewswire.com/news-releases/mortenson-creates-first-of-its-kind-augmented-reality-app-for-construction-visualization-300490295.html

13. N.a. "Project — Metaverse Lab — Hackaday.io." Hackaday.io. n.d. Web. 1 May 2018. https://hackaday.io/project/5077/logs?sort=oldest&page=3

14. Foundry. "Read Full Article." Foundry. n.d. Web. 1 May 2018. https://www.foundry.com/trends/design-visualisation/AR-AEC

15. Skanect 3D Scanning Software By Occipital. "Skanect 3D Scanning Software By Occipital - The Easiest Way To 3D Scan With the Structure Sensor and Kinect-like 3D Sensors." Skanect 3D Scanning Software By Occipital. n.d. Web. 1 May 2018. http://skanect.occipital.com/

16. N.a. "." Microsoft.com. 4 Jan. 2018. Web. 1 May 2018. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/ismar2011.pdf

17. Mattzmsft. "HoloLens hardware details - Mixed Reality." Docs.microsoft.com. 21 Mar. 2018. Web. 1 May 2018. https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details

18. Thetuvix. "Spatial anchors - Mixed Reality." Docs.microsoft.com. 21 Mar. 2018. Web. 1 May 2018. https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-anchors

19. N.a. "." Donar.messe.de. 9 Mar. 2017. Web. 2 May 2018. http://donar.messe.de/exhibitor/hannovermesse/2017/U208979/dsh-data-sheet-eng-495327.pdf

20. Daqri. "Building the Worksite of the Future ' Augmenting Humanity ' Medium." Medium. 15 Dec. 2016. Web. 2 May 2018. https://medium.com/daqri-blog/building-the-worksite-of-the-future-b6db32873e32

21. N.a. "(5) The Daqri Smart Helmet ? A closer look — LinkedIn." Linkedin.com. n.d. Web. 2 May 2018. https://www.linkedin.com/pulse/daqri-smart-helmet-closer-look-nathan-gaydhani/

22. Skarredghost. "Sketchup Viewer for HoloLens review - The Ghost Howls." The Ghost Howls. 20 Mar. 2017. Web. 2 May 2018. https://skarredghost.com/2017/03/20/sketchup-viewer-for-hololens-review/

23. Efficient variants of the ICP algorithm - IEEE Conference Publication Efficient variants of the ICP algorithm - IEEE Conference Publication. (2018). Ieeexplore.ieee.org. Retrieved 11 May 2018, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=924423

24. Unity Unity. (2018). Unity. Retrieved 11 May 2018, from https://unity3d.com/

25. Academy - Mixed Reality Academy - Mixed Reality. (2018). Docs.microsoft.com. Retrieved 11 May 2018, from https://docs.microsoft.com/en-us/windows/mixed-reality/academy

26. Microsoft/MixedRealityToolkit-Unity Microsoft/MixedRealityToolkit-Unity. (2018). GitHub. Retrieved 11 May 2018, from https://github.com/Microsoft/MixedRealityToolkit-Unity

27. Using the Windows Device Portal - Mixed Reality Using the Windows Device Portal - Mixed Reality. (2018). Docs.microsoft.com. Retrieved 12 May 2018, from https://docs.microsoft.com/en-us/windows/mixed-reality/using-the-windows-device-portal

28. Technologies, U. Technologies, U. (2018). Unity - Manual: Importing Objects From SketchUp. Docs.unity3d.com. Retrieved 12 May 2018, from https://docs.unity3d.com/560/Documentation/Manual/HOWTO-ImportObjectSketchUp.html

29. Technologies, U. (2018). Unity - Scripting API: Physics.Raycast. Docs.unity3d.com. Retrieved 13 May 2018, from https://docs.unity3d.com/ScriptReference/Physics.Raycast.html

30. Technologies, U. Technologies, U. (2018). Unity - Scripting API: Bounds. Docs.unity3d.com. Retrieved 13 May 2018, from https://docs.unity3d.com/ScriptReference/Bounds.html

31. Convex Hull — Set 1 (Jarvis's Algorithm or Wrapping) - GeeksforGeeks Convex Hull — Set 1 (Jarvis's Algorithm or Wrapping) - GeeksforGeeks. (2013). GeeksforGeeks. Retrieved 13 May 2018, from https://www.geeksforgeeks.org/convex-hull-set-1-jarviss-algorithm-or-wrapping/

32. Liagson/ConcaveHullGenerator Liagson/ConcaveHullGenerator. (2018). GitHub. Retrieved 13 May 2018, from https://github.com/Liagson/ConcaveHull

33. Introduction to Octrees Introduction to Octrees. (2018). GameDev.net. Retrieved 13 May 2018, from https://www.gamedev.net/articles/programming/general-and-gameplay-programming/introduction-to-octrees-r3529/

34. Case study - Expanding the spatial mapping capabilities of HoloLens - Mixed Reality Case study - Expanding the spatial mapping capabilities of HoloLens - Mixed Reality. (2018). Docs.microsoft.com. Retrieved 27 May 2018, from https://docs.microsoft.com/en-us/windows/mixed-reality/case-study-expanding-the-spatial-mapping-capabilities-of-hololens

35. Firman, Michael. 'RGBD Datasets: Past, Present and Future.' 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2016): 661-673.

36. Magic in the Making Magic in the Making. (2018). Magicleap.com. Retrieved 28 May 2018, from https://www.magicleap.com/

37. Company, M. Company, M. (2018). Meta — Augmented Reality. Metavision.com. Retrieved 28 May 2018, from http://www.metavision.com/

38. Magic Leap — Crunchbase Magic Leap — Crunchbase. (2018). Crunchbase. Retrieved 28 May 2018, from https://www.crunchbase.com/organization/magic-leap

39. Armeni et al., "3D Semantic Parsing of Large-Scale Indoor Spaces," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 1534-1543.
from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7780539&isnumber=7780329