# Spatial Reconstruction using Microsoft HoloLens™

Department of Computer Science
Faculty of Engineering
University of Hong Kong

COMP4801 - Final Year Project

Final Report

GUPTA, Aman
3035206885

SCHNIEDERS, Dirk
Supervisor

April 2018

**Abstract**

Advancements in hardware have led to the invention of the Microsoft HoloLens™, a mixed reality system that allows users to perform physical interactions with the digital world in real time. The device combines features of Virtual Reality (VR) with the physical world, creating Augmented Reality (AR). Nowadays, 3D indoor models can be constructed using construction blueprints and plans; however sometimes, the plans are unavailable or might not be a true representation. In such cases, the areas must be surveyed and reconstructed through a time consuming and costly process. This project aims to showcase the ease with which The Microsoft Hololens™ can be used to perform spatial reconstruction of indoor environments and to visualize them as holograms. The reconstructed, processed holograms will be complemented with interactive features. This report presents the technology being used, the team's methodology, and application design. The report explains the implementation of the various modules of the project and finally discusses the limitations and challenges the team faced during development.

# Contents

# List of Figures

## Acknowledgement

I am extremely grateful to Dr Dirk Schnieders for choosing to work with the team. Dr Dirk Schnieders supported the project through every stage. His guidance and supervision helped the team in realizing its vision of building the platform.

I am also grateful to Dr K.P. Chan, course coordinator for COMP 4801, for creating the the project schedule and deliverables. I also thank Ms Loretta Choi who accepted to be the second examiner of the project and assess the work of the team.

I also acknowledge the MSc Programme Office, the Department of Computer Science, the Faculty of Engineering, and the University of Hong Kong for providing the necessary hardware for this project.

I would also like to extend my gratitude to the creators of and contributors to Microsoft HoloLens, CGAL Library, Unity3DAzure™ Storage Service, and other Open Source libraries that form an integral part of this project.

Lastly, I would also like to thank my team mate, Waleed Zafar, for working with me and providing support throughout the development of the project.

# Abbreviations

**API** Application Programming Interface

**AR** Augmented Reality

**CGAL** Computational Geometry Algorithms Library

**HLA** HoloLens™ Application

**IDE** Integrated Development Environment

**I/O** Input/Output

**LiDAR** Light Detection and Ranging

**prefabs** prefabricated components

**PRPU** Post-Recording Processing Unit

**REST** Representational State Transfer

**SDK** Software Development Kit

**SPA** Single Page Application

**UWP** Universal Windows Platform

**UX** User Experience

**VCS** Version Control System

**VR** Virtual Reality

# 1 Introduction

## 1.1 Project Details

This project was conceptualized and implemented as a part of the course COMP 4801 - Final Year Project for the degree of Bachelor of Engineering in Computer Science at the University of Hong Kong. The supervisor for the project was Dr. Dirk Schnieders. The team consisted of two students - Aman Gupta and Waleed Zafar.

The team has built a HoloLens™ Application (HLA) to scan and visualize indoor environments that runs on the HoloLens™ hardware. The application is complemented by a processing server running on the Microsoft Azure™ platform as well as a web portal built using the React framework.

## 1.2 Motivation

Computer modeling software has allowed designers and architects to visualize 3-dimensional models on 2-dimensional screens. With advancements in technology and the advent of VR and holographic projections, it is now possible to visualize and interact with environments in unforeseen ways. The Microsoft HoloLens™ is the leading tool for holographic manipulation. It features some of the most advanced AR technology that is commercially available. Interested developers have already created some groundbreaking applications to bring the activities of designing and modeling to the AR space.

The market for applications that project holograms is booming; however, the development of these holograms usually requires using complex modeling software to mirror a real-world indoor environment. Capturing environments and objects to rebuild them as holograms is an unexplored realm in this nascent industry. Existing systems trying to achieve the same objective usually exhibit accuracy and quality issues owing to hardware limitations, or cost prohibitions. The Microsoft HoloLens™ provides a decent trade off between cost and hardware, and has not been previously used, in a public and commercial capacity, for indoor 3D scanning purposes.

The team sees an opportunity in using the HoloLens™ device to create a comprehensive solution to serve the needs of capturing and visualizing indoor environments. The approach of scanning indoor environments instead of de-

signing them on a software is analogous to what a photograph is to a painting - a user will be able to build more accurate 3D models at a much faster pace.

Engineers and Designers will be able to use the application in a variety of different ways. For example, model visualization and the accompanying statistics will save designers the arduous process of manually measuring and recreating indoor spaces. As our team's technology would improve the speed of modeling, the productivity of such users will improve. Moreover visualization of the models as holograms and VR environments will help users gain a new perspective of the environment that is missing in existing technologies (Section 2).

This report begins with a review of relevant works pertaining to spatial reconstruction. It then highlights the primary objectives for the project, followed by an overview of the technology utilized by the team. As I, Aman Gupta, worked on the HoloLens, web portal and Microsoft Azure™ integration, the report mainly focuses on the application design, architecture and its implementation. Moreover, the report briefly discusses the other component of the project, the PRPU. The report finally outlines the methodology employed by the team as well as a brief discussion of the challenges, limitations and future extensions for the project.

# 2 Related Works

There are several hardware and software solutions in the public domain that have attempted to solve the problem: the scanning and reconstruction of real world environments. However, they have been limited by hardware, sensors and processing power alike. Microsoft HoloLens™ is the first Augmented Reality device of its kind.

## 2.1 Microsoft Kinect™ Fusion [8][17]

Kinect Fusion 3D provides object scanning and model creation using a Microsoft Kinect™ sensor. Capabilities of the project include the ability to simultaneously see and interact with detailed 3D models scanned using the sensor. However, Kinect Fusion™ utilizes a single depth sensing camera for tracking, which results in poor performance while scanning flat surfaces. Furthermore, the Kinect™ sensor is not wireless, hindering mobility for the project in varying indoor environments.

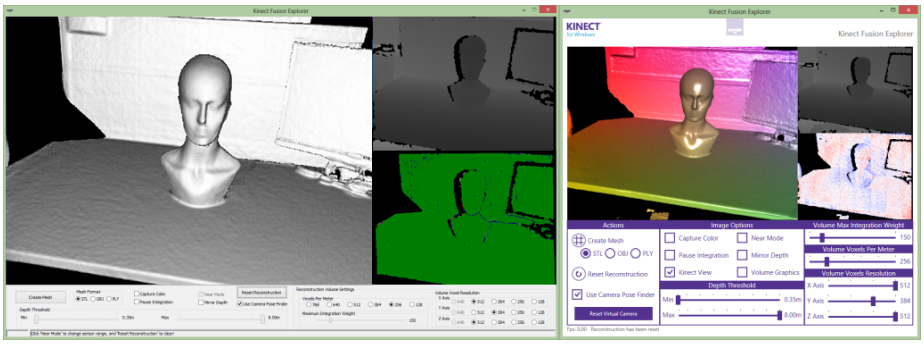

Figure 1: Kinect Fusion Scanning and Reconstruction [17]

## 2.2 Laser Scanning[16]

Figure 2: Leica ScanStation P40

Figure 3: Trimble LaserAce 1000 Rangefinder

Laser Scanning is one of the most popular methods of indoor surveying to generate 3D models, and is used in a number of commercial applications. One of the most common technologies under laser scanning utilizes Light Detection and Ranging (LiDAR) to gather 3D point cloud data. However, extracting actual building models from large data-sets generated via LiDAR is a costly and time-consuming endeavor and requires experienced technicians at work. Some commercial solutions utilizing laser scanning technology include the Trimble LaserAce 1000 Rangefinder[18] and the Leica ScanStation P40[13]. These solutions do exhibit problems similar to those faced by the LiDAR, is-

sues include shape distortion and angular inconsistencies while scanning indoor environments.

Apart from the specific limitations described above, most solutions do not use mesh correction algorithms to fill in the holes and remove bias from generated 3D models. The absence of a strong, coherent mesh correction process results in inaccurate 3D models which might not be suitable for applications that demand high precision, such as construction and industrial automation.

# 3 Objectives

The primary objective of the project is to create an AR application that is able to scan, reconstruct and model indoor spaces. These goals can be realized by implementing a HLA and a PRPU.

## 3.1 HoloLens

The underlying objective of building the HLA is to give the user ability to scan indoor spaces and model them in 3D spaces. The application must feature a clean and simplistic user interface that is controlled by voice commands and hand gestures.

### 3.1.1 Scanning

The team plans to use the depth-sensing camera and existing APIs of the Microsoft HoloLens™ to create 3D mappings of surrounding environments. The HoloLens™ provides hardware and software that is already capable of stitching together surface meshes as a user moves around in an indoor environment. The user simply presses the record button then walks around a room scanning its features.

### 3.1.2 Visualization

The proposed application will give the users the ability to project holographic representations of 3D models scanned via the HLA, and span, zoom, and rotate the holograms on the surface it is projected. Spatial information such as floor surface area, surface area of walls will also prove useful to the end user.

## 3.2 Spatial Reconstruction

The need for spatial reconstruction arises because the spatial data collected from the HoloLens™ may contain outliers, holes and noise. To overcome this issue, a special processing algorithm needed to be developed that runs automatically on each recorded mesh. The team also wants to give users the flexibility to adjust the parameters of the processing algorithms so that the resulting model is noiseless, accurate and water-tight.

### 3.2.1 PRPU

The HoloLens™ is the first device of its kind and has limited computation power to handle intricate processing algorithms. Therefore, a PRPU was developed to perform complex manipulations to the raw data collected from the HoloLens™ scanner.

### 3.2.2 Web Portal

Limited user controls on the HoloLens™ make it cumbersome for the user to customize the parameters for the implemented processing algorithms. Through a complementary web portal, the objective of spatial reconstruction will be further personalized for every recorded space.

# 4 Technology Used

The project incorporates a wide array of technologies, SDK and hardware. This section discusses the core technologies that were used to develop, debug and deploy the entire solution. Besides the technologies listed below, several libraries, tutorials, and code snippets were studied, adapted or referenced in the project are featured in the Appendix, Section 13.1.

## 4.1 Microsoft HoloLens™

It is an advanced head-mounted display with tinted visors that function as a see-through display, as shown in figure 4. The HoloLens™ has 4 environment sensing cameras and 1 depth sensing camera[5]. The cameras mounted on the

device are already optimized for spatial mapping and come coupled with complex algorithms that are able to create 3D triangular meshes for the spaces it captures.

Figure 4: HoloLens™ Developers Edition Kit [10]

The device supports voice commands as well as hand gestures as input. It uses the orientation of the head to identify the gaze vector of the user and point. Selections in the AR environment can be either made by a tap gesture by the index finger (Figure 5) or using the HoloLens™ clicker pictured in the center of Figure 4 [6].

Figure 5: Tap Gesture within the HoleLens' field of view [6]

## 4.2 HoloLens™ SDK

The HoloLens™ follows the Universal Windows Platform (UWP) development framework and therefore has no separate special SDK or toolkit. However, Microsoft has released an open-source collection of scripts and components to

accelerate the development of applications targeting Windows Mixed Reality [11]. Libraries such as HoloToolkit and Mixed Reality Design Lab are essential for development of the application. The development environment includes:

1. **Visual Studio 2017:** Integrated Development Environment (IDE) for building and deploying UWP applications to the HoloLens. Visual Studio was the premiere IDE used in the development of the project. It also has built in emulators for Azure™ Services for debugging.

2. **Unity 2017.2:** A Unity engine that comes with Mixed Reality Support [7] and is primarily used to develop applications in 3D space. Prefab components such as Spatial Understanding, Input Mapping, and HoloLens™ Camera are inserted into a Unity scene along with 3D objects to design interactions and behaviours.

3. **Mixed Reality Portal:** The Mixed Reality Portal is the latest emulator developed by Microsoft to test Mixed Reality applications without a HoloLens™ [12]. Although the HoloLens™ device is essential for testing capturing techniques, development of the User Experience (UX) work-flow can be done on an emulator.

## 4.3 Microsoft Azure™ [4]

Microsoft's cloud infrastructure, Azure™, provides developers with various storage and deployment tools that were extensively used to build the project. Azure™ was chosen as the go-to service for all back-end needs of the team due to various reasons. Firstly, Azure™ is very cost effective as it automatically scales the resources depending on the usage of the application in a *Pay As You Go* subscription plan. Secondly, as the HLA is built through Unity into a C# application using Microsoft Visual Studio, the team realized that Azure™ SDK will integrate into the system seamlessly. Lastly, using enterprise level infrastructure has the added of advantage of advanced security, global availability and continuous support. Services provided by Microsoft Azure™ used in the project include:

1. **Azure™ Blob Storage:** The service provided us with containers to store our recorded and processed meshes in Wavefront OBJ file format.

2. **Azure™ Function:** This service provides developers with the toolkit to develop serverless functions that run on time , event, or REST based event triggers. This service was used to start the mesh processing once a file is uploaded and also serve as a REST server for the web portal. Azure™ Functions integrate into the Visual Studio IDE which provides a one-click publish feature to the production environment.

3. **Azure™ App Service:** The service was used to host the Single Page Application (SPA) built on the React framework. The beta Continuous Deployment feature was also integrated into the service which is triggered automatically whenever a changeset is committed to a GitHub repository. Using this feature, a fresh build of the SPA is automatically built and deployed.

# 5 Application Design

The final system incorporates the web portal, PRPU and HLA. This system is able to achieve all the goals described in Section 3.

### Deliverables

The main deliverable of the project is the HLA. This application is assisted by a robust PRPU and an extension, the web portal.

I, Aman Gupta, completed the architectural design, implementation and software development of the HLA and the web portal as well as assisted in the deployment of the processing library to a PRPU.

Waleed Zafar completed the design, implementation, software development and testing of the underlying processing library. He is also responsible for the deployment of the PRPU to Microsoft Azure.

## 5.1 HLA

Through the HLA, a user can scan an indoor environment as well as view the processed holograms by accessing public collections stored on Microsoft Azure™ Blob Storage. Users are also be able to export the holograms as immersive environments for VR headsets through the web portal by downloading the files

in Wavefront OBJ file format [19]. The application is also designed to accommodate the user's needs for customized processing of the scanned holograms through a REST interface for the PRPU through the web portal.

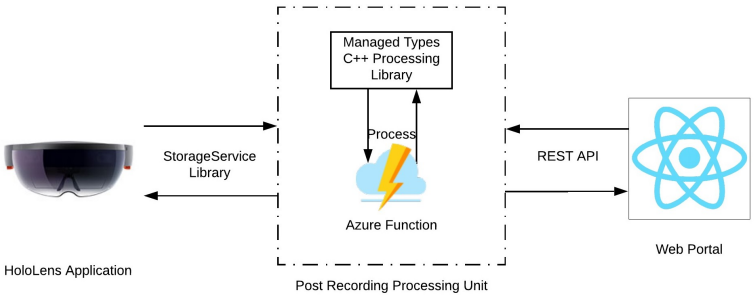


Figure 6: Overview of the Application Design: Azure™ Function serves as the single API for both the web portal and HLA

Once deployed on and installed through the Windows Store, the HLA can be accessed through a HoloLens™ device's *bloom* menu. On initialization the application will display a holographic menu allowing a user to either view their collection or begin recording a new environment.

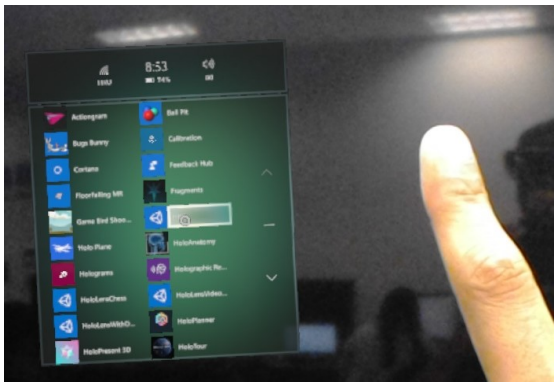

Figure 7: Bloom Menu - The *Start* menu of the HoloLens

### 5.1.1 Recording Indoor Environments

This is the core feature of the project. The heavy weight of spatial mapping is already taken care of by the device which records a 3D mesh of triangles to

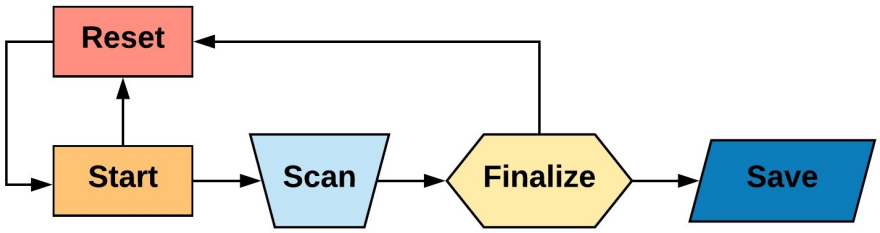


Figure 8: Recording Process Flow

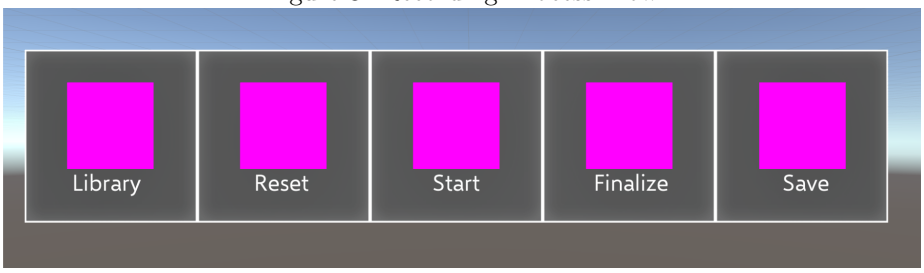


Figure 9: Recording View in Unity Editor. Buttons from left: Library, Reset, Record, Finalize and Save

identify planes in its vicinity. However, the existing software is limited by the buffer of mesh it can record and its inability to fill gaps left by unidentified planes.

On selection of the *Start* button, the solution will begin spatial mapping and offer the ability to *Reset* the recording. Walls and planes that are being scanned by the HoloLens™ will be overlaid with a mesh material (Figure **??**). Upon satisfactory recording, the user will select *Finalize* which would trigger a request to finish the scanning process. Subsequently, clicking the *Save* button will serialize the meshes, store them in a temporary file on the HoloLenseventually use a REST PUT request to add the file to the Azure™ Blob Storage. The process flow is depicted in the Figure 8.

### 5.1.2 Visualization

The user will be able to revisit their recorded environments using the *Library* button in the main scene or the *My Library* voice command. On selecting an

environment, the user will have the ability to place the environment's hologram on a flat surface such as table and rotate it along the horizontal or vertical axis. Information relating to the surface area of walls, floor and ceiling will be visible with the hologram.

Figure 10: A team of designers looking at a holographic model. [10]

The team plans to design a multi-user platform (Figure 10) for visualization so that businesses can creatively use the application for their needs collectively as a team. API for the aforementioned feature is already available in the UWP SDK however, due to the unavailability of multiple HoloLens™ devices, the team will be unable to implement or test this feature.

## 5.2 PRPU

The need for a processing unit arises because the spatial data collected from the HoloLens™ contains outliers, holes and noise. Once the user stops their recording, the recorded spatial data will be delivered to the PRPU server. The PRPU will be responsible for applying filters and spatial reconstruction algorithms to create a useful model which will then be saved on the server as well as the HoloLens™ device.

Using Azure™ Functions, the processing algorithm will be triggered once a blob of filetype OBJ is uploaded to the input container. As the Computational Geometry Algorithms Library (CGAL) API is written in *C++*, a managed types[9] C++/CLI interface will to be created to interact with the Function written in C# depicted in Figure 13.

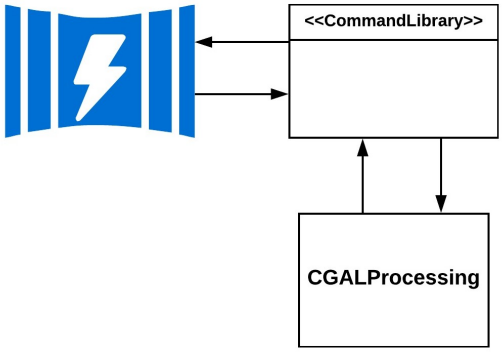


Figure 11: Design of the PRPU: The Azure™ Function uses a an interface to interact with the Processing Library

### 5.3 Web Portal

The web portal will serve two primary needs of the project. Firstly, it will serve as the source of all user documentation and project information. Secondly, it will be used to access the meshes so that a user can download them as OBJ files or perform additional mesh processing. *React* framework will be used to build the SPA which will use REST requests to interact with the Azure™ Function. The Azure™ Function will have the additional responsibility of acting as the server hosting the PRPU library.

## 6 Methodology

The project involved design and implementation of three softwares: HLA, PRPU and the web portal. This chapter presents the methodology that was employed throughout the development of the project.

### 6.1 Hardware Setup and SDK Installations

Microsoft provides a detailed instruction set to install the SDK[7]. To benefit from the on-going developments in the field of Mixed Reality, Unity Engine 2017.2 and the latest iteration of Visual Studio were used to implement the project. Furthermore, the team has also cloned repositories from GitHub with

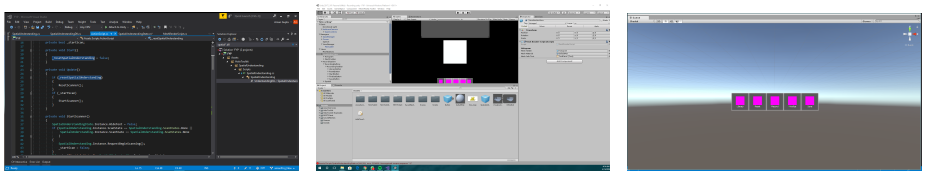

Figure 12: From Left: Visual Studio 2017, Unity Editor, Unity Scene Manager

a library of components and scripts designed for HoloLens™ development[11].

A Version Control System (VCS), git, was used to manage the code for all the software. Unique private repositories on GitHub - a popular repository hosting service for git - were initialized to manage the code by the team. GitHub makes it easier to collaborate within the team, track changes through unique commits, and integrate libraries as submodules and services, it was ideal for development of this scale. Besides, keeping the code on GitHub leaves the team with the flexibility of eventually making the project open-source in the future.

## 6.2 Feasibility Study

The main aim of this task was to understand the structure and components of the SDK as well as the functions of the prefabs. Therefore, the team carried out API research on the HoloLens™ and explored tutorials created by Microsoft Academy to understand the development process for a HLA.

It was essential that the team understand the capabilities of the device in the early stages of the project. Knowing these limitations helped the team decide and design the objectives and scale of the project.

Through the feasibility study, the team became confident that a solution could be developed within the time range using HoloLens™ and Unity. The results of the study proved beyond reasonable doubt that the solution would have the ability of observing and recording spatial data accurately. Moreover, the team realized that the SDK provides the developers with the ability to easily build Gaze-Gesture-Voice (GGV) input as controlling methods[10].

This phase helped the team discover software such as MeshLab that can be used to visualize meshes or point-cloud data. MeshLab also became essential for debugging the PRPU as it implements several surface mesh libraries.

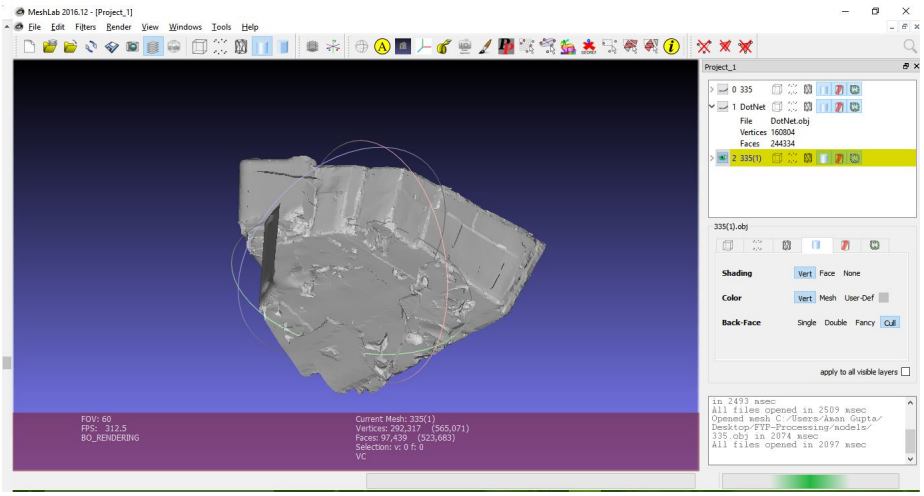


Figure 13: An instance of mesh lab with a loaded model. MeshLab was used as to understand mesh transformations and for mesh visualization.

## 6.3 Spatial Reconstruction

After understanding the HoloLens™ device, the focus was on developing the spatial reconstruction part of the project. This part consisted of the HLA and the PRPU.

Spatial reconstruction for the raw mesh generated by the HoloLens™ was performed in two distinct phases. The first phase utilizes the Spatial Understanding API available via the HoloToolkit for Unity. The mesh generation feature performs the recording and partial reconstruction within the HLA as described in Section 5.1.1. The partially reconstructed mesh is then uploaded to the application's PRPU for the second phase of spatial reconstruction which automatically runs the processing task and stores the resultant mesh.

## 6.4 Visualization and User Collection

Upon successfully completing the tougher task of recording the indoor environments, time was devoted on the visualization of the spaces. First software to support model placement was developed. Thereafter, manipulations such as rotations, pick, and drop were incorporated.

## 6.5 Environment Export and Web Portal

Once the application was able to perform the two aforementioned functions to an acceptable degree of satisfaction, the team worked on making the service web-friendly. Nodejs libraries including *react*, *react-bootstrap* and their dependencies were installed and a mock application (pictured 14) was created. Once the mock GUI was complete, REST requests to fetch the mesh lists from the storage were incorporated. As the processed OBJ files are stored in in a view-only Azure™ Storage, capabilities for exporting the models in OBJ format (compatible format across various Virtual Reality devices) was added next and thereafter, the ability to perform spatial processing tasks with custom parameters on-demand.

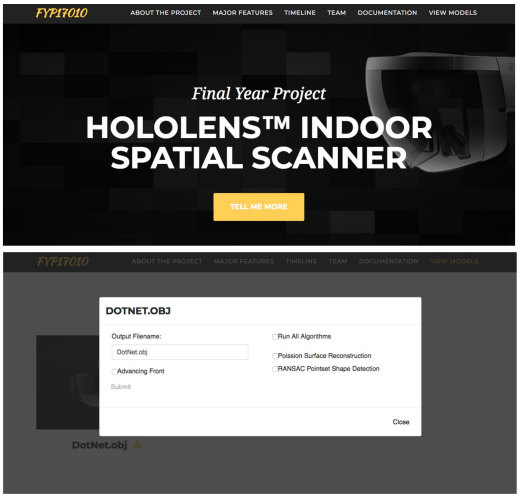


Figure 14: A mockup of the proposed web portal. Top: Landing page. Bottom: Open Modal to perform mesh processing.

## 6.6 Mesh Processing

In parallel to the development of the aforementioned phases, Waleed Zafar continued research and implementation of spatial mesh processing algorithms using CGAL library.

# 7 HLA

The architecture of the HoloLens™ Application is comprised of four separate modules: Library, Recording, Visualization and OBJ File I/O. This allowed greater control over the design of each feature and boosted the development efficiency. The views were designed and implemented using the Unity engine with the help of prefabricated components (prefabs) from the Holo Toolkit for Unity. To control the view activation, a *View Manager* singleton was implemented. Other global prefabs include the *Storage Manager* script, *Speech Input* prefabs and basic camera and cursor prefabs. Each view features a 3D toolbar that uses the *Tag Along* script to follow the gaze of the user. The following subsections discuss the implementation of each of the views.

## 7.1 Library Module

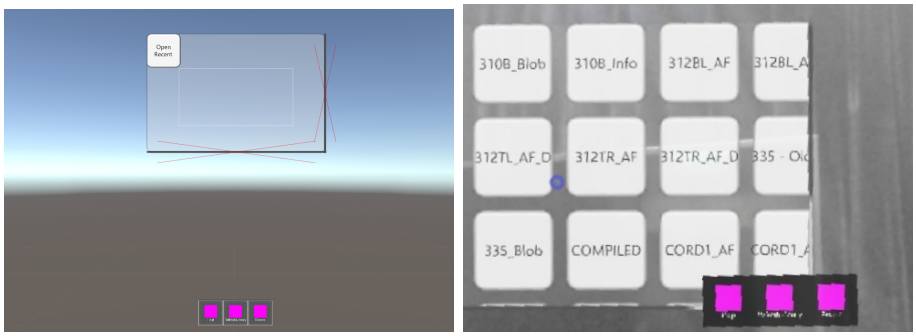


Figure 15: Library View in HoloLens™. Buttons from left: Help, Refresh Library, Record

This is the landing view for the application. The view features a panel populated with buttons to download and visualize the meshes along with a toolbar to navigate to the recording view. To populate the view, the application queries the Azure™ Storage service using the Storage Manager script. A refresh button complements this method in case of network errors on initialization of application.

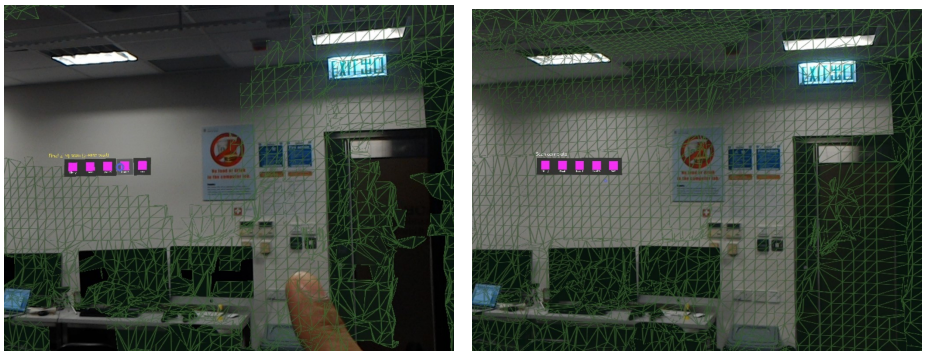

Figure 16: Recording View (HW310B, HKU). Left: Before the user has clicked on the Finalize button. Right: Once the mesh has been finalized by the SU library

## 7.2 Recording Module

The recording view was the first scene implemented in Unity. The view features 5 buttons as shown in figure 9 for user input as well voice inputs for *Record Mesh*, *Finalize Mesh*, *Reset Mesh*, and *Save Mesh*. The SU library prefab and its support library are used extensively in this module and perform bulk of the heavy lifting for the mesh data collection. Using the library's *RequestBeginScan()* function, the application triggers the SU singleton to start the scanning process. Along with it, a request is also sent to the library to show the mesh being generated and the library is queried every Unity frame (each second) for spatial data. This information is displayed in conjunction with helpful directions for the user once the user has scanned at least $5m^2$. A minimum threshold during the scanning is enforced before the user is able to choose the *Finalize* button. The threshold has two criteria both of which use the statics from the SU library:

1. Total Surface Area: Findings by a research group at The Chinese University of Hong Kong [14] state that an average living area per capita of Hong Kong's subdivided flats is $4.44m^2$ this led the team to set a minimum threshold of $3m^3$ or total surface area of $18m^2$.

2. Cell Quality: Inside the SU DLL library, the mesh is subdivided into 8cm sized voxel cubes [3] and information about the number of visible cells and the number of good quality cells is also available. At least 50% of

the visible cells must have a good quality in the system. The reason for a low threshold at the scanning stage is twofold: the input process becomes excessively long and arduous with a higher threshold, and the finalization stage generates a reasonably adequate input mesh for the PRPU.

To provide user feedback about when the scan meets the criteria, the primary information text changes from color red to color green and displays *Tap to finalize*.

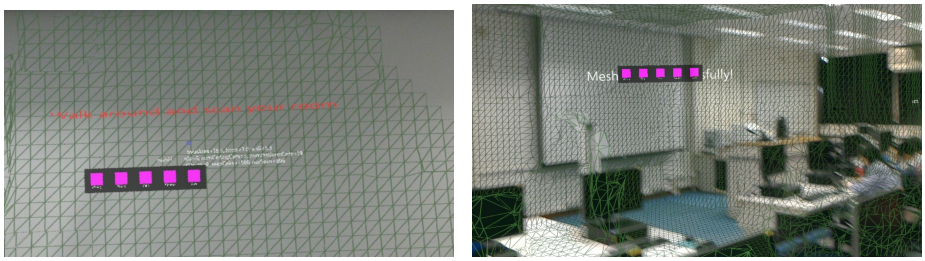


Figure 17: Recording View (HW312, HKU). Left: during the recording stage: as the data has not met the minimum criteria, the instructions appear in red along with spatial statistics. Right: after mesh has been finalized and saved sucessfully to the Azure™ Storage

Once the user clicks finalize, the software requests the SU library to finish the scan. This process uses the proprietary DLL library provided by Microsoft to generate a playspace for the user. On finalization the primary information text displays the *Mesh Finalized* message to alert the user that it is safe to save the mesh. After the finalization process is complete, the user is also able to walk around and assess the overlaid mesh to determine if it is satisfactory.

If the user remains unsatisfied and wants to scan the mesh again, they have the ability to reset the mesh and restart the scanning process using the reset button or voice input. Clicking on the save button transfers control to the OBJ File I/O module described later in the subsequent section (Section 7.4.1). The spatial information about the total, wall, ceiling and floor area are also stored with the data as OBJ file comments in the file.

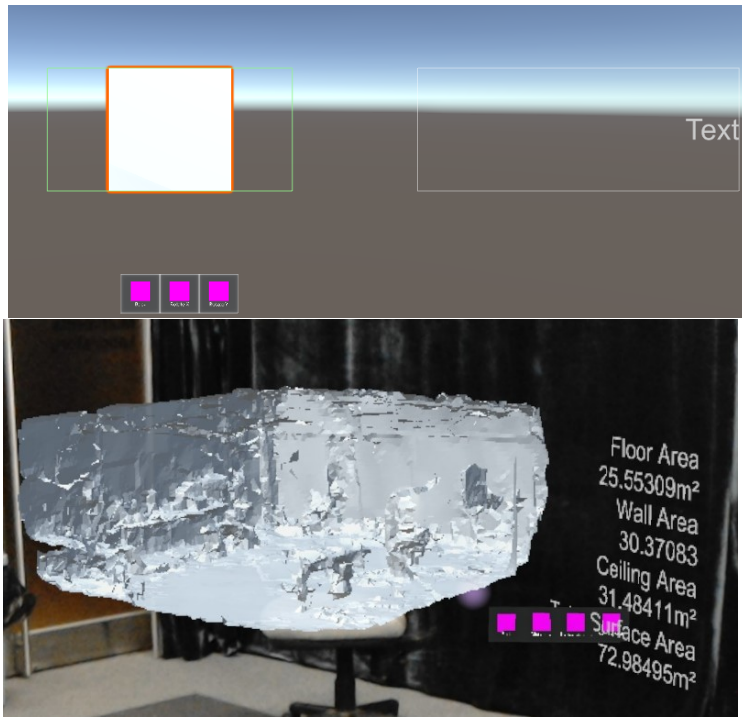


Figure 18: Visualization View in Unity Editor and on HoloLens™. Cuboid placeholder for the mesh and Spatial Information Text Placeholder. Buttons from left: Back, Rotate X, Rotate Y.

## 7.3 Visualization Module

Upon selecting a mesh from the Library View, the View Manager toggles the Visualization View where the Storage Manager singleton initiates a GET request to the blob storage and retrieves the entire OBJ file as a string buffer. The string buffer is processes using the OBJ I/O module (Section 7.4.2) deserialises this information and returns submeshes along with the extra spatial statistics that were stored during the recording stage. This statistical information is displayed in the view adjacent to the mesh as shown in figure 18.

The green outline around the mock model in the figure represents the box collider, a basic cube-shaped collision primitive, for the mesh that ensures that other components such as the toolbar do not *collide* with the mesh. The same collider helps to drag the mesh around a room and place it on a surface once the user gazes at the mesh and the cursor identifies the collider. The *Rotate X*

and *Rotate Y* buttons as their names suggest are used to rotate the mesh about its X and Y axis smoothly. Toggling one action deactivates the other and each can be disabled by pressing the button again.

## 7.4 OBJ File I/O Module

The OBJ file I/O module consists of the *Model Loader* and *Room Saver* class. Due to the memory intensive nature of I/O, the tasks are performed asynchronously however, the current hardware still suffers from a brief lag during these tasks. The underlying implementation of the classes are as follows:

### 7.4.1 Room Saver

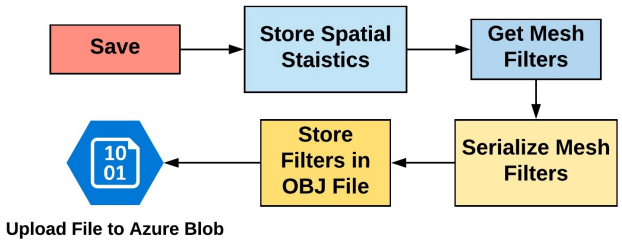


Figure 19: Process flow of Room Saver Sub-module: The get mesh filters process uses the SU class to retrieve the spatial data of the current scan.

Once a mesh is finalized and the user initiates the saving procedure, the system first converts the statistical information into a human readable string and passes along this information to an asynchronous *Save Room* method. This method again uses the SU library to *Get Mesh Filters* - the sub meshes from the visible grid as a list of Unity *Mesh Filter*. As no working method to serialize mesh filters into OBJ file was implemented in the toolkit, a new one was adapted and implemented, the method was added to the existing *MeshSaver* class (and a pull request to the feature commit was generated for the open source community).

Under the hood, each submesh filter is serialized into its vertices, vertex normals and face indices as discussed in Waleed Zafar's report. The serialized submeshes are collected and stored as an OBJ file in the temporary file system

of the HoloLens. The process returns the file path to the temporary file and triggers the Storage Manager singleton to *PutObjectToBlob* method.

### 7.4.2 Model Loader

Upon receiving the entire mesh as a string buffer, this module first extracts spatial information from the file. Thereafter, it splits the mesh back into submeshes for sterilization. Each submesh passes through an asynchronous *Process Submesh* method which returns a Unity *Mesh* component.

The general algorithm for the *Process Model* was adapted[1] outlined on the following page.

**Algorithm 1** Model Loader Algorithm

```
1:  function ProcessModel(meshstring)
2:      vertices, normals, triangles and faceData ← List as Vector3
3:      mesh ← Unity.Mesh0
4:      while  dostringi ← meshstring
5:          if i.BeginsWith("v") then
6:              temp = i as Vector3
7:              vertices.Add(temp)
8:          if i.BeginsWith("vn") then
9:              temp = i as Vector3
10:             normals.Add(temp)
11:         if i.BeginsWith("f") then
12:                                          ▷ fff/fff/fff fff/fff/fff fff/fff/fff
13:             temp ← Parse index for vertix and normals
14:             faceData.Add(temp)
15:             triangle ← Parse triangle info
16:             triangles.Add(triangle)
17: Reindex normal and vertices
18: mesh ← normals, vertices, triangles                  ▷ Unity Mesh Created
19: return mesh
```

For each submesh Unity *Mesh Renderer* and *Mesh Filters* components are initialized and the generated *Mesh* is attached to the new Mesh Filters components. With the now sequence complete, the Mesh Loader relinquishes control

and the application transitions to the Visualization module.

# 8 Web Portal

Using *create-react-app* npm library, a client-side SPA was created and *EcmaScript6™* scripting-language specification was used to program the web portal. It was decided to port the original Final Year Project website to the react framework as well. This ensured all the documentation of the project is easily accessible to the user. The implementation required designing several modular *React Components* to build dynamic cards, modals and, forms for the portal. The *react-bootstrap* helped implement a responsive, mobile friendly SPA as shown in Figure 8.

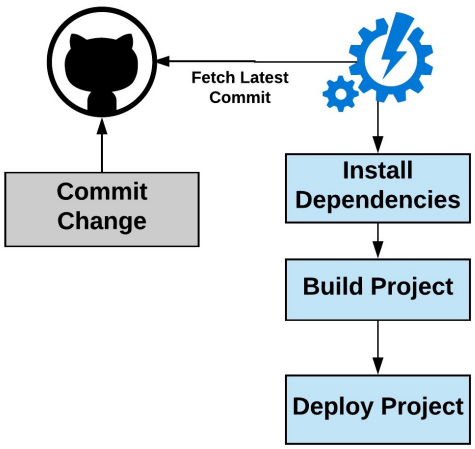


Figure 20: Process flow of Azure™ Continuous Delivery: Azure™ automatically detects when a new commit has been made and builds and deploys the web portal to the web app service.

Since Azure™ Services have the ability to connect GitHub and automatically deploy new builds to the web app, the service was integrated with the SPA. The *Continuous Delivery* system workflow is shown in Figure 20 and it improves production build deployment speed as well as eliminates chances of human error.

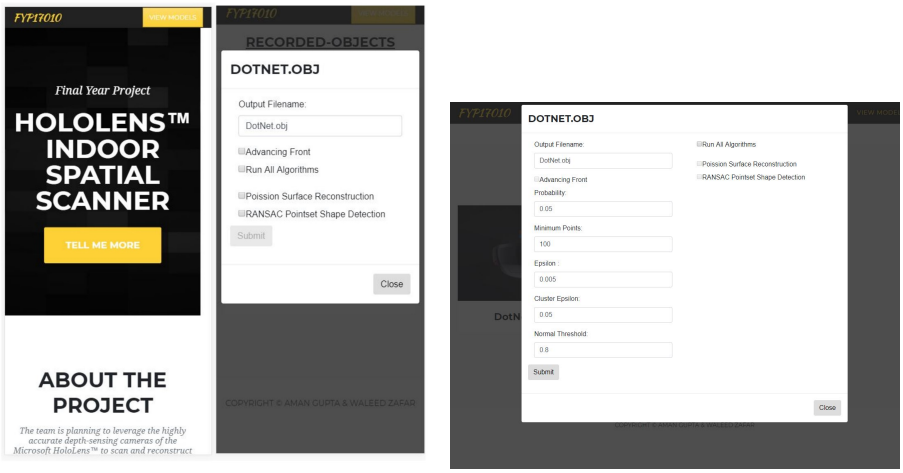


Figure 21: Right: Landing Page and Processing Modal (mobile). Left: The processing modal can be used to trigger a mesh to be reprocessed using custom parameters for the functions.

# 9 PRPU

The responsibility of researching, designing, implementing and deploying fell on Waleed Zafar. The methodology, application design and implementation for the same are detailed in his report. However, as I, Aman Gupta, assisted in the deploying stage and made the decision of using Azure™ Functions for our server-less back-end, a brief discussion on Azure™ Functions is included.

## 9.1 Azure™ Function

The Azure™ service is running two separate functions: *REST Function* and *Processing Function*. As discussed in Section 5.2, the function's C# code and processor's C++ code made it necessary to develop a *managed C++/CLI* interface[9]. Alternate methods of interaction such as PInvoke[2] and a command line interface were also considered were quickly abandoned. Use of PInvoke to build an unmanaged library and calling from C code greatly increased the dependency of correct system architecture resulting in complications with the server deployment. The built executable and its complementary command line interface for the mesh processor were adopted to perform quick, extensive tests on the underlying processing algorithms.

The *Processing Function* is triggered when a blob of file type OBJ is added to an input storage container by the HoloLens. Thereafter the function copies the blob from the storage into its temporary memory, calls the processor, and uploads the processed mesh to an output storage container. Figure 22 shows a detailed diagram while figure 23 shows a locally running instance of the Azure™ Function.

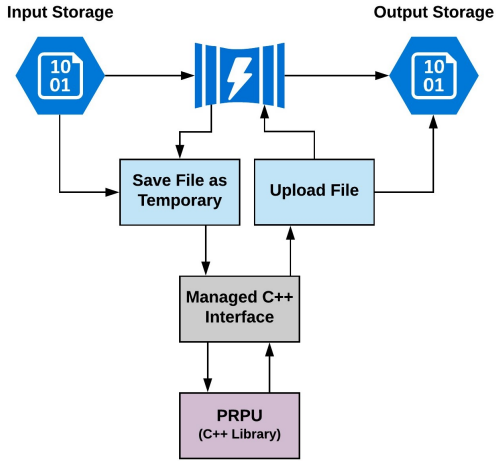


Figure 22: Process diagram of Azure™ Function: The function fetches the file from input storage, saves as temporary file, uses the C++ interface to call the PRPU and uploads the processed mesh to an output storage.

The Azure™ service was also chosen to act as the REST API. The endpoints for the service are listed along with their respective request and response headers in the Appendix.

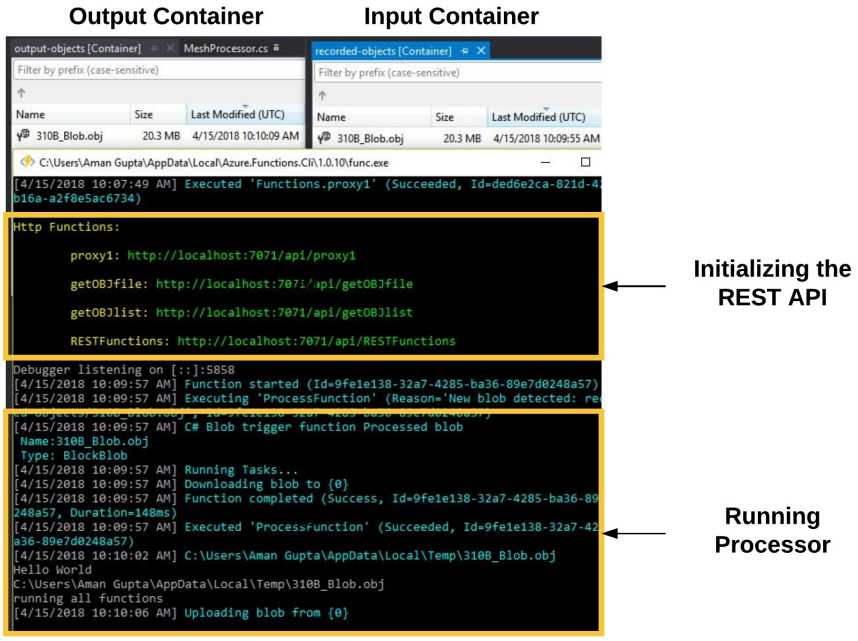


Figure 23: A locally running instance of the Azure™ Function

# 10 Limitations and Difficulties

The team has limited the current development of the project to handle only flat surfaces that are reasonably distant from one another, as the project put a greater stress on identifying boundary planes, walls, ceiling and floors. Furthermore, the depth sensing capabilities of the device are limited to a spherical radius of only 3.1 meters. This reduces the scope of spatial data scanning as it become difficult to scan indoor areas with high ceilings. However, this sensor radius should still be sufficient for general use cases.

Another limitation of the project is that extensive movements while wearing the device can lead to disorientation and lossy surface stitching. There is also a problem with the available SDK because of which the tracking state of the world anchor on the device are lost. This leads the spatial mesh data, model visualization view and other components to misalign with the user's gaze. These issues are being addressed by Unity and will be incorporated into future releases of the Unity Editor.

Besides the aforementioned limitations on the device, the fact that the playspace recording is limited to only $16m^2$ at one time was a major setback to the team. However, the team's findings show that the device is capable of performing satisfactory spatial reconstruction of indoor environments. Future iterations of the device would do away with these restrictions and possibly extend the API further. On-going and future developments are expected to greatly benefit the application.

As the team worked with complex libraries, in varied languages, written by a diverse set of developers and with limited documentation, development of the PRPU was significantly delayed. The team managed to run the processing functions on a local machine independently and through the Azure™ Function emulator however it failed to deploy the application on the production environment. Discussions with Microsoft's Azure™ support team are on-going and the team is confident they will be able to overcome this final hurdle.

Implementation of user authentication for both the HLA and web portal were considered; however, as the PRPU development stayed on the critical path beyond the initial estimate, they were abandoned to work on the core components. These features were deemed non-essential and relegated to a future stage of development.

Despite these hardships, the team fulfilled its objective of building a proof of concept for the final year project to their satisfaction.

# 11 Future Works

During the feasibility study, the team realized the great potential of an all-in-one 3D scanner and visualizer. Although the team achieved the objectives as planned, the scope of the project can be extended and revisited in future.

There is scope for training a machine learning algorithm to learn and understand various objects (pieces of furniture) in a scanned room. The algorithm would help the processing algorithms to filter out unnecessary objects, help label and divide the mesh components, and improve the overall reliability of the processing software.

Once the HoloLensbecomes capable of handling the rigorous process software, it would be in the best interest of the project to incorporate the processing task into the device itself. Coupled with interactive menus to customize the pro-

cessing algorithm, this extension would make the software a stand-alone service.

On the visualization front, the team would have liked to extend the capabilities of the device even further by incorporating varied colors for different kinds of surfaces. As the application aims to improve modeling and visualization techniques for engineers and designers, having the ability to collaboratively view spatial data through multiple devices is also deemed necessary to bring the product to market.

Lastly, the team also brainstormed the idea of giving the user a 3D grid of dots that the user can join to make models. The user could construct rectangular or triangular planes and incrementally join them to create a boxed model. This extension opens the recording module to a wide array of applications. Besides having greater control of the scanning stage, the extension could be used to create free hand 3D models that can rival those of Google Tilt Brush [15]. However, early findings suggested that the HoloLensunable to maintain world anchor positions of a large number of meshes; therefore, the team focused on developing the automatic scanning solution.

# 12 Conclusion

The team has successfully developed a proof of concept for a new technique of mapping and reconstructing indoor spaces through a futuristic device, the Microsoft HoloLens™. The application has immense benefits for designers and engineers as it gives them a convenient new way to scan and visualize environments as holograms. By realizing the two main objectives of creating a HLA and a PRPU early on, the team was able to work with great synchronization. Using modern technologies for development, code collaboration and deployment, the large code stack remained manageable especially during the last sprint to complete the application.

This report describes the implementations of the Spatial Reconstruction Solution: a HLA coupled with a PRPU and a web application that will be used to create holographic models of existing indoor environments. The HLA ticked all the check boxes in terms of Recording, Saving, and Visualizing the mesh. Furthermore, the front-end of the web portal was completed ahead of schedule and that allowed the team to spend more time to debug the issues with the PRPU. Although the processing algorithm meets all the requirements of hole

filling, surface reconstruction, and plane segmentation at the time of writing the report, the team is still trying to debug issues in the production environment with assistance from Microsoft's Azure™ support team.

Although it was disappointing to learn about the limitation on the playspace area for spatial scanning, the team is sure that the issue as well as the other limitations of the Developer Edition HoloLens™ device will be eventually overcome.

In the meantime, the team is focusing on finding a solution to the deployment issue. Alternatives to Azure™ Functions such as virtual machines or an Apache Tomcat Server may be considered.

Our results suggest that there is a great scope for further development of the application and the choice of hardware is fitting for the team's use case. The HoloLens™ adequately identifies walls, floors, tables, and other surfaces in real time, paving the way for the team to perform spatial reconstruction.

The HoloLens™ device was announced only three years ago and it has already opened a brand new realm of Mixed Reality. With this project we have proven that it has applications beyond the trivial sphere of games and the like and the ability to make an impact in workplace. These ground breaking results prove that the legacy of the device and this project will inspire new avenues of technological discovery.

# 13 Appendix

## 13.1 Other libraries, code snippets, and tutorials referenced from the internet:

1. The Western Town Tutorial by Cameron Vetter
2. How to Save Understanding Mesh - Issue on MixedRealityToolkit-Unity
3. Introduction to Managed C++ by Nish Nishant
4. Save Mesh as OBJ - Pull request by Aman Gupta on MixedRealityToolkit-Unity

## 13.2 Extra Figures

| End Point | Function | Description | Response |
|---|---|---|---|
| / | proxy1 | GET Redirect to Web Portal | https://fyp-webportal.azurewebsites.net |
| /blobs/{container-name}/{blob-name} | getOBJfile | GET OBJ file from storage | https://hkufyp17010.blob.core.windows.net/{container-name}/{blob-name} |
| /blobs/{container-name} | getOBJlist | GET list of OBJ files in container | {<br>meshes: [<br>{<br>name: string<br>metadata: object<br>}<br>]<br>} |
| /process | RESTFunctions | PUT a request for processing models | N/a |

Figure 24: Endpoints for the REST API

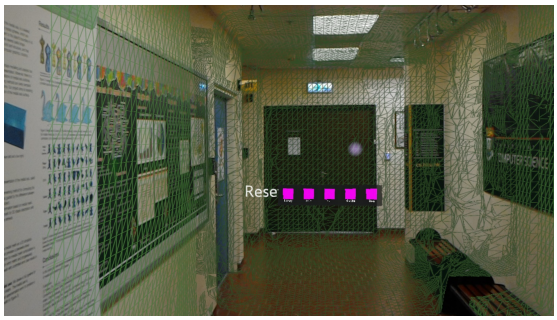

Figure 25: Finalized Mesh of a corridor, 3F Haking Wong, HKU

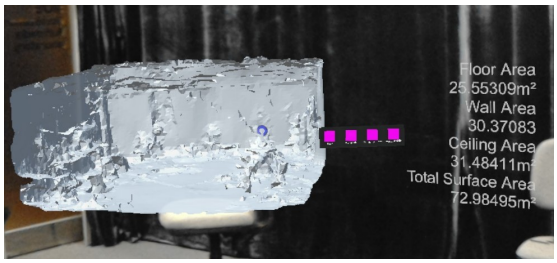


Figure 26: Visualization Result (without the mesh processing)

# References

[1] El Anónimo. *ObjImporter: A utility class for importing .obj files into a Unity mesh at runtime.* URL: `http://wiki.unity3d.com/index.php?title=ObjImporter/`.

[2] Microsoft Corporation. *Call Native DLLs from Managed Code Using PInvoke.* URL: `https://docs.microsoft.com/en-gb/cpp/dotnet/how-to-call-native-dlls-from-managed-code-using-pinvoke`.

[3] Microsoft Corporation. *Case study - Expanding the spatial mapping capabilities of HoloLens.* URL: `https://docs.microsoft.com/en-us/windows/mixed-reality/case-study-expanding-the-spatial-mapping-capabilities-of-hololens`.

[4] Microsoft Corporation. *Directory of Azure Services.* URL: `https://azure.microsoft.com/en-us/services/`.

[5] Microsoft Corporation. *HoloLens hardware details.* URL: `https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details`.

[6] Microsoft Corporation. *HoloLens Use Gestures.* URL: `https://support.microsoft.com/en-us/help/12644/hololens-use-gestures`.

[7] Microsoft Corporation. *Installation checklist for HoloLens.* URL: `https://developer.microsoft.com/en-us/windows/mixed-reality/install_the_tools`.

[8] Microsoft Corporation. *KinectFusion Project Page.* URL: `https://www.microsoft.com/en-us/research/project/kinectfusion-project-page/`.

[9] Microsoft Corporation. *Managed Types (C++/CLI)*. URL: https://docs.microsoft.com/en-gb/cpp/dotnet/managed-types-cpp-cli.

[10] Microsoft Corporation. *Microsoft HoloLens*. URL: https://www.microsoft.com/en-us/hololens.

[11] Microsoft Corporation. *MixedRealityToolkit - Unity*. URL: https://github.com/Microsoft/MixedRealityToolkit-Unity.

[12] Microsoft Corporation. *Using the Windows Mixed Reality simulator*. URL: https://developer.microsoft.com/en-us/windows/mixed-reality/using_the_windows_mixed_reality_simulator.

[13] Leica Geosystems. *Leica ScanStation P40 / P30 - High Definition 3D Laser Scanning Solution*. URL: http://leica-geosystems.com/products/laser-scanners/scanners/leica-scanstation-p40--p30.

[14] Alfred Ho. *The unlivable dwellings in Hong Kong and the minimum living space*. URL: https://www.hongkongfp.com/2015/07/27/the-unlivable-dwellings-in-hong-kong-and-the-minimum-living-space/.

[15] Google Inc. *Tilt Brush by Google*. URL: https://www.tiltbrush.com/.

[16] A. Jamali et al. "3D Indoor Building Environment Reconstruction using Least Square Adjustment, Polynomial Kernel, Interval Analysis and Homotopy Continuation". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W1 (2016), pp. 103–113. DOI: 10.5194/isprs-archives-XLII-2-W1-103-2016. URL: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W1/103/2016/.

[17] Microsoft Corporation Developers Network. *Kinect Fusion*. URL: https://msdn.microsoft.com/en-us/library/dn188670.aspx.

[18] Trimble. *LaserAce 1000 Rangefinder*. URL: http://www.trimble.com/support_trl.aspx?Nav=Collection-77546&pt=LaserAce%201000%20Rangefinder.

[19] The University of Utah. *Object Files (.obj)*. URL: http://www.cs.utah.edu/~boulos/cs3505/obj_spec.pdf.