Spatial Reconstruction using Microsoft HoloLens™

Department of Computer Science
Faculty of Engineering
The University of Hong Kong

Final Report

ZAFAR, Waleed
2013600952

April 2018

Abstract

The fields of Virtual Reality, Augmented Reality and Mixed Reality have advanced rapidly in the last few years, leading to the development of the Microsoft HoloLens™. Being the most advanced augmented reality device on the market today, the HoloLens™ provides state of the art hardware and software for mapping and understanding 3D spatial data. This paper showcases the use of the device to gather 3D scans and perform 3D spatial reconstruction of real world indoor environments, leading to the automated development of 3D models that can be re-purposed for various industrial applications. The solution focuses on the reconstruction of interior environments as 3D models which can be visualized and manipulated within the HoloLens™, as well as viewed on other modelling software and VR ready hardware. The solution developed by the team incorporates a HoloLens™ application, a web portal and a post processing unit that performs the 3D reconstruction. The end product developed by the team automatically generates 3D models with reasonable accuracy for various indoor environments. A number of algorithms have been tested and incorporated into the post processing unit as part of the spatial reconstruction process, details of which are available within this paper. This paper should be read in tandem with the paper authored by my team mate, Mr. Aman Gupta, which outlines details of the HoloLens™ application and the web portal. This paper presents the technology utilized, the development methodology, research and alternative solutions, alongside a detailed description of the final implementation of the post processing unit for spatial data.

Contents

List of Figures

List of Tables

Acknowledgements

The team acknowledges and extends its sincerest gratitude to Professor Dirk Schneiders for supervising and consulting with them on the project. The team also acknowledges the University of Hong Kong, the Faculty of Engineering and

the Department of Computer Science for providing the necessary framework and hardware to proceed with the project.

1 Introduction

3D modelling has been used across an array of industries such as construction and design over the past few decades. The technology utilized for such modelling has leaped recently with the advent of 3D printers and advanced scanning and visualization technologies. The Microsoft HoloLens™ provides cutting edge support for Augmented Reality applications, coupled with advanced visualization technology that can be leveraged for both scanning and visualizing objects in 3D space.

The HoloLens™ has drawn immense support from AR developers, with applications for various purposes such as education and gaming available commercially. The development of holograms for such applications remains a cumbersome process, requiring 3D modelling software such as Unity, AutoCAD and 3DS Max. The problem worsens for models of indoor environments, particularly for applications such as construction where a high degree of accuracy is mandatory. Our objective is to leverage the scanning technology provided by the HoloLens™ to scan indoor environments and generate accurate and reusable 3D models of those environments. While existing solutions utilizing different types of hardware are already in use in commercial applications, most solutions present problems either stemming from accuracy or set up and running costs. The HoloLens™ provides a decent cost-accuracy trade off, and has yet to be utilized for indoor scanning and modelling in a commercial capacity.

The proposed solution comprises of three main deliverables - an application for the HoloLens™, an accompanying web portal and a post recording processing unit. The different parts of the solution are linked together using Azure™ cloud services, which serves as the data sink between the HoloLens™ and the PRPU. The PRPU itself is housed on Azure as well, where it is controlled using a Representational State Transfer (REST) Application Programming Interface (API). While there are a number of improvements that can be performed in terms of fine tuning the final processed 3D models, the current state of the processed models is still a considerable improvement over the raw 3D spatial data captured by the HoloLens™.

1.1 Project Details

the project was developed as part of the course COMP4801 - Final Year Project under the Faculty of Computer Science at the University of Hong Kong. The supervisor for the project was Dr. Dirk Schnieders. The project team consisted of two final year undergraduates - Aman Gupta and Waleed Zafar.
The primary objective of the project was to create an Augmented Reality (AR) application for the HoloLens™ that is able to scan, reconstruct and model 3D indoor environments. The team has developed an HoloLens™ Application (HLA) and a PRPU to realize these objectives.

1.2 Existing Solutions

There are various hardware and software solutions available commercially that attempt to achieve the same objective as the team – scan and reconstruct indoor environments. Each of those solutions, however, exhibits some kind of problem, generally related to either the accuracy of the generated model, or running and set up costs associated with the solution. The HoloLens is the first AR device of its kind, both in terms of hardware provided as well as the internal software, and stands as a good contender to solve the problem at hand with regard to the efficiency and affordability of the solution.

1.2.1 Google Tango

The Google Tango project provides an API for the Android platform that allows applications to track the position and orientation of the mobile device and recognize known environments. Technically, an application similar to the one proposed can be built using Tango. However, the Tango project is still under development, and presents a number of quality and execution related issues, including limited processing power, lack of depth information for spatial data, and other hardware/software issues rising due to developmental bugs.

1.2.2 Microsoft Kinect Fusion

Kinect Fusion 3D provides object scanning and model creation capabilities using a Microsoft Kinect sensor. The Fusion provides capabilities to simultaneously see and interact with detailed 3D models previously scanned using the Kinect. However, the Microsoft Kinect camera utilized within the project provides a

single depth sensing camera for visual tracking, resulting in poor performance while scanning flat surfaces. Furthermore, the camera itself is not wireless, hindering mobility while scanning varying types of indoor environments.

1.2.3 Laser Scanning

Laser Scanning is among the most popular methods of indoor environment surveying to generate 3D models, and is regularly utilized in commercial applications such as construction. Among the most common of such laser scanning methods is the LiDAR (Light Detection and Ranging), which allows a user to gather 3D point cloud data. Extracting actual building models from large datasets generated via LiDAR is, however, a costly and time consuming process, and requires experienced technicians to work. Some commercial solutions utilizing such technology include the Trimble LaserAce 1000 RangeFinder and the Leica ScanStation P40. Such solutions, despite commercial use, exhibit similar problems as the aforementioned, alongside issues such as shape distortion and angular inconsistency in some indoor environments. On top of these, many commercial solutions do not use mesh correction algorithms before generating a final 3D model, which could result in holes, distortions and hallucinations in the generated models.

2 Proposed Solution

This section provides a high level overview of the features of the overall system the team has designed. The developed solution comprises of three three primary deliverables - the HLA, the PRPU and a web portal. High level details of each of these are outlined below, with the PRPU explored in depth later in the paper. Details regarding HLA and the web portal are finely explained in the paper presented by my teammate.

2.1 HoloLens™ Application

An application for the Microsoft HoloLens™ forms the first deliverable of the solution. The application itself has been designed for the Universal Windows Platform (UWP), using Unity 2017.2 and Microsoft Visual Studio 2017. The application provides two primary functions to the user - Recording a new envi-

ronment and visualizing existing models. The application communicates with Azure cloud storage, where it stores recorded environments and pulls processed models from for visualization. Another important feature incorporated within the HLA, though not directly visible to the user, is Spatial Understanding. Utilizing that, the HLA formulates the first step of the spatial reconstruction process by incorporating meta information about the captured space as well as performing partial smoothing and hole filling operations on the raw spatial data

2.1.1 Record a New Environment

This is the primary feature of the application, whereby a large chunk of the required spatial mapping data is generated using the HoloLens™. The HoloLens™ by default records spatial data as vectors describing the vertices of triangles that overlay any physical surface, and this triangular mesh overlay is visible to the user in real time while the user is recording an environment[16]. Once a user selects the 'Record a New Environment' option, the HoloLens™ overlays the user's surroundings with a triangular mesh overlay (see figure 1 below), indicating that the environment is being recorded. As the recording proceeds, the application continues the triangular mesh overlay, allowing the user to navigate to various areas of the room that still need to be recorded, minimizing holes in the scanned model resulting from the user's actions. Once a user is satisfied with the real-time mesh overlay, the user selects the 'Finalize' option, which calls the Spatial Understanding API within the HLA, and results in a regeneration of the mesh overlay after some geometric processing via spatial understanding. This model is then uploaded to Azure™, where the PRPU takes over the spatial reconstruction process and uploads a processed model back to Azure. The PRPU is necessary because the spatial data extracted from the HoloLens™ can exhibit a number of problems such as holes, hallucinations and bias in the model.

2.1.2 Visualization

The second purpose that the HoloLens™ application serves is to allow a user to visualize any model within their library as a hologram. Users have the option to view models in their library by selecting the 'My Library' option upon app start up. Once a user's library is open, the user can select any of the available models for visualization. The app supports 3 basic features - placement of holograms on flat surfaces, rotating models, and scaling models by size. The team will

Figure 1: Triangular mesh over an unidentified column and ceiling

Figure 2: Blue lines over a partially identified wall

explore development of advanced manipulation features such as model editing and furniture manipulation within the fine tuning stage of the project, alongside the development of a multi user platform whereby multiple users would be able to collaborate on the same model (see figure 3 below).

Figure 3: A team of designers looking at a holographic model. [12]

2.2 Web Portal

The team has also developed a web portal as part of the solution. The web portal serves two primary purposes - allowing a user to download models from their library, and performing custom spatial reconstruction processes utilizing the PRPU. As future extensions to the web portal, features such as 3D model conversion and public model sharing can be incorporated.

2.3 Technology

This section describes the technology used for building the team's proposed solution.

2.3.1 Microsoft HoloLens™ [13]

The Microsoft HoloLens™ is one of the most advanced Augment Reality headsets available commercially for the mass market. It is a pair of smart glasses with a tinted visor and a Heads Up Display (HUD) built into each lens which allows for projection of images/holograms while leaving the user's actual view of his/her surroundings unencumbered. The HoloLens™ provides 4 environment sensing cameras and 1 depth sensing camera[9]. The HoloLens™ comes packed with software that optimally utilize the available cameras to generate spatial mapping data in the form of a 3D triangular mesh. This allows the team to extract a coarse 3D model using the HoloLens™ API, which then undergoes post processing to generate the final 3D model. The device can be controlled by voice commands as well as various gestures, alongside the availability of a Bluetooth™ enabled Clicker to replace the select gesture. The HoloLens™ utilizes a cursor visible at the center of a user's field of view that can be controlled by the user's head's orientation. This is coupled with the Tap and Bloom gestures (see figures 4 and 5 below) which allow a user to select something (analogous to a mouse click) or to open up the main menu respectively[10].

2.3.2 Development Environment

2.3.2.1 Microsoft HoloLens™ The HoloLens™ itself runs the Windows Mixed Reality platform under the Windows 10 operating system. App development for the device thus only requires the Universal Windows Platform development framework[9]. Apart from that, Microsoft has released an open-source collection of scripts and libraries to aid the process of developing applications for the Microsoft HoloLens™ [14]. The team also utilizes the aforementioned contributions from Microsoft, including the Mixed Reality Toolkit and the Mixed Reality Design Lab. The rest of the development environment for the HoloLens™ application includes:

1. **Microsoft Visual Studio 2017:** An Integrated Development Environment (IDE) developed by Microsoft which allows the development of

Figure 4: Tap Gesture within the HoloLens' field of view [10]

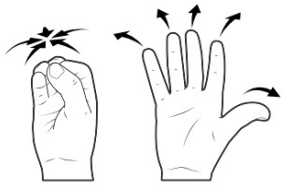


Figure 5: Bloom Gesture

Universal Windows Platform (UWP) applications.

2. **Unity 2017.1:** The Unity engine features Mixed Reality Support [11] and allows for the development of applications featuring 3D graphics, most notably games. Components specific to the HoloLens™ such as Spatial Mapping can be inserted into a Unity scene, and specific scripts to manipulate such components can be modified using a text editor before building a Unity solution and deploying it to the HoloLens™ using Visual Studio 2017.

3. **Mixed Reality Portal:** The Mixed Reality Portal is an emulator for the HoloLens™ developed by Microsoft, to allow for testing and development of applications for the device [17]. The Mixed Reality Portal allows for the development of workflows and the UI for the application, while performance and hardware input testing still require the use of an actual HoloLens™ device.

2.3.2.2 Post Recording Processing Unit The PRPU, which is responsible for the bulk of the spatial reconstruction features of the designed solution,

has been developed using C++ on the macOS High Sierra operating system. This has been done so as to interface conveniently with open source graphics processing libraries, a large number of which are usually written in C++. The development environment for the PRPU includes:

1. **XCode 9:** An Integrated Development Environment (IDE) developed by Apple which allows the development of cross platform C++ applications, among many others.

2. **Microsoft Visual Studio 2017:** The Microsoft Visual Studio 2017, running on Microsoft Windows 10, has been utilized to interface with the Azure cloud platform and directly publish the PRPU to Azure from within the Integrated Development Environment (IDE).

3. **CMake:** CMake is a cross platform and open source software for managing the build processes of software using a compiler-independent method. CMake has been utilized for out-of-source compilation of the PRPU to allow OS and IDE independent development of the PRPU by the team, utilizing different machines and development environments.

4. **Apple LLVM 9.0:** The Apple LLVM 9.0 compiler has been used to compile the C++ project on the macOS.

2.3.3 Web Server and Portal

The team assessed a number of development technologies for the development of the web portal, and finalized the utility of Azure™ services to provide data storage for both the raw and processed 3D models. Azure™ has also been utilized to house the PRPU allowing access to it from the HoloLens™ via a REST API. The front end for the web portal has been designed using ReactJS. Figure **??** showcases the current outlook of the web portal, presenting provisions for browsing models, downloading models and performing custom processing on the models via the PRPU.

2.3.4 Meshlab [5]

Meshlab is an open source 3D mesh processing system, allowing manipulation of 3D models using an array of methods. Meshlab presents a GUI allowing a user to view the effects of a number of different functions on models under testing.

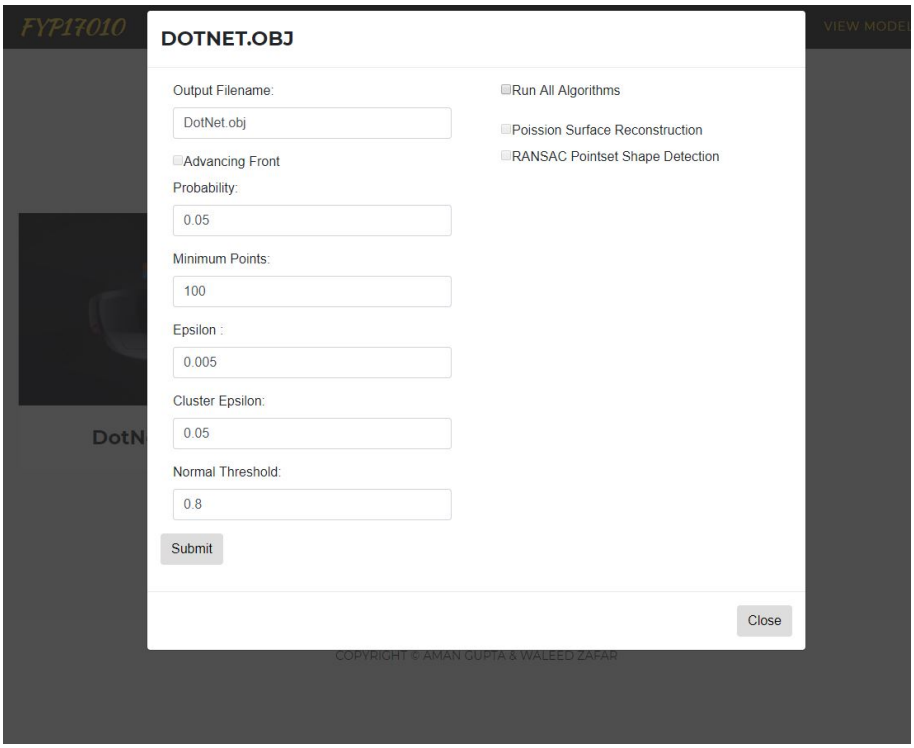


Figure 6: Web Portal View

Furthermore, the source code, available in C++ over Github, also provides the team with a better understanding of the requirements and intricacies of mesh processing algorithms. The team has conducted considerable research via Meshlab by trying various mesh processing algorithms on 3D models fetched from the HoloLens™. Such operations include refinement, smoothing, outlier and noise removal, convex hull constructions, hole filling operations and alignment and stitching operations. Usage of Meshlab has been invaluable during the development of the PRPU as a tool to visualize and manipulate 3D models and observe and analyze various models generated during the processing steps.

2.3.5 PCL [30] and VCG

The Point Cloud Library (PCL) is a large scale open source project for 2D and 3D image and point cloud processing. The library exposes an Application Programming Interface (API) in C++, alongside numerous point cloud processing

algorithms which the team can leverage. The team identified a number of classes and modules within PCL to perform spatial reconstruction operations. These include filtering, downsizing using the voxelGrid filter, plane detection using the RANSAC algorithm, concave hull constructions, among others. While the results generated via PCL had been positive, they were qualitatively superseded by the usage of another library, CGAL, for such processing. While the PCL implementation is still available within the PRPU, it is not actually used for spatial data processing in the final version of the PRPU. A key benefit of utilizing PCL earlier had been its Sample Consensus module, which allowed plane detection leading to the construction of an independent outer hull for any model. The major issue with PCL had been, as the name suggests, its utilization of only point clouds for spatial reconstruction purposes, without using any information about facets in 3D space. An alternative the team explored early on was the inclusion of the VCG library, although, as with PCL, the VCG implementation is not actively included in the PRPU anymore.

2.3.6 CGAL

CGAL is a software project that provides access to efficient and reliable geometric algorithms exposed via a C++ API. The library provides numerous packages for spatial data processing, including modules to perform computations on spatial data for n dimensions. CGAL follows an exact computation paradigm, that is to say that it produces geometrically accurate results, whereby intermediate round off errors are handled correctly and numbers of arbitrary precision are used to generate accurate results. The PRPU relies, for all of the currently implemented algorithms, on CGAL. This has lead to a number of constraints during the development as well, which will be discussed in the Spatial Reconstruction section later.

3 Methodology

This section outlines the methodology that has been followed throughout the development of this project.

3.1 Feasibility Assessment

The team has already carried out a feasibility assessment for the project. The assessment included API research on the HoloLens™ to understand the information that can be extracted from the device, as well as the exploration of various tutorials by Microsoft Academy [7] [8] and the developer community regarding various HoloLens™ applications. The purpose of such study was to understand the structure of a HoloLens™ application and to understand the purpose of the various components of such an application. Results of the feasibility assessment seemed favorable to the team, allowing them to continue development of the project using the HoloLens™.

3.2 Hardware and Software Setup

The hardware setup, primarily setting up the HoloLens™ for development was fairly trivial, with the device supporting remote deployment of apps using Microsoft Visual Studio 2017. The same IDE has also been used to publish the PRPU C++ project to Azure remotely. Apple's XCode served as the primary IDE for the development of PRPU. Microsoft has provided detailed instructions on setting up the Software Development Kit (SDK) for app development for the universal Windows Platform [11]. The team had set up the latest version s of Unity and Microsoft Visual Studio, alongside XCode for development. These were aided by the Mixed Reality Toolkit for Unity [15] [14] for development of the HLA and the setup of CGAL, PCL and VCG. The development process was aided by Meshlab as well, which the team utilized throughout for research and visualization purposes. All of the hardware and software setup was completed by October of 2017. The only addition later on was the setup of the Azure cloud environment alongside the development environment for the web portal, which were completed by March 2018.

3.3 HoloLens™ Application Development

The HoloLens™ application was the primary deliverable which the team had to develop, because that serves as both the source and sink for the 3D spatial data that needs to be processed. Development details for the HLA are outlined in greater detail in the paper by my teammate.

3.4 3D Spatial Reconstruction

The 3D spatial reconstruction phase, tied to the PRPU deliverable, has been developed in C++. The first necessity for 3D spatial reconstruction, trivially, was the development of the 'Record a New Environment' feature for the HLA. That allowed sample data from the HoloLens™ to be extracted and utilized throughout the development of the PRPU. Following the extraction of sample models from the HLA, the team was able to work in mostly distinct streams with regard to the development of the HLA and the PRPU respectively. The entire development process, for both the HLA and the PRPU was version controlled and housed on shared Github repositories, providing the team with a convenient collaboration tool. The spatial reconstruction phase utilizes three open source C++ libraries - PCL, VCG and CGAL. Following extensive trial and testing using the three libraries, the final version of the PRPU only includes functions and modules from CGAL, because that library provides the greatest amount of flexibility during development and delivers the best qualitative results among the three. The choice to use CGAL within the PRPU has also been aided by the availability of mostly thorough documentation and tutorials alongside the library. Details of the spatial reconstruction phase and the final implemented design of the PRPU are all outlined in the Spatial Reconstruction section.

3.5 Model Visualization

The model visualization feature was developed following the development of the spatial data extraction feature of the HLA i.e. the 'Record a New Environment' feature. The HLA currently provides visualization support for 3D models in OBJ file format, alongside the ability to move and place models on flat surfaces within the environment. This is augmented by the ability to rotate models along the X and Y axes to view varying perspectives. The application also provides meta information alongside models, including statistics such as the floor area, the surface area of walls, the ceiling area, among others. Future modifications and enhancements to the model visualization features can include 3D model manipulation techniques and even real time visualization of spatial reconstruction algorithms within the HoloLens™.

3.6 Web Portal

The web portal was developed as the last deliverable of the project. This provides a user with a convenient interface to download 3D models and to perform spatial reconstruction operations on models using the PRPU. Details of the web portal development and utility are provided in the report by my teammate.

4 Spatial Reconstruction

Spatial reconstruction has been the most challenging aspect of designing the proposed solution. As mentioned earlier, the spatial reconstruction phase is loosely split into two parts. The first attempt at spatial reconstruction is carried out as part of the HLA using the Spatial Understanding API. The second phase is carried out by the PRPU. This section outlines the challenges to begin with, the objectives of the spatial reconstruction phase, and possible approaches and algorithms that can be utilized for spatial reconstruction.

4.1 Challenges and Objectives

There are a number of problems exhibited by the models extracted from the HoloLens™. The qualitative issues with any given model generated by the HLA can be broadly categorized as one of three issues [16]:

1. **Holes:** Real-world surfaces are missing from the spatial mapping data. This becomes a problem when the HoloLens™ cannot scan certain surfaces within an environment or portions of surfaces, either because of user movement, the lighting conditions or the surface color and texture. The HoloLens™ has a limitation of scanning data only up to 3.1 meters away from the HoloLens™, so surfaces that are farther away during the scan may not show up.

2. **Hallucinations:** This is the appearance of surfaces and objects within the spatial mapping data that are not present in the real world environment.

3. **Bias:** The HoloLens™ does experience bias, whereby surfaces in the spatial mapping data are imperfectly aligned with their real-world counterparts. Surfaces thus appear either pulled in or pushed out.

Although a number of factors affect the spatial data retrieved from the HoloLens™, the following are the most common ones:

1. **User Motion:** The speed and smoothness of a user's motion while scanning can greatly affect the spatial data output. The camera has a 70-degree field of view and a range of 0.8-3.1 meters while scanning. If a user never approaches a real-world surface closer than 3.1 meters, then that surface may not be scanned. Similarly, a user rushing past an object might not provide the HoloLens™ with sufficient time to scan the object and incorporate that in the spatial mapping data. Furthermore, the user should be careful to scan all objects within a given environment - objects or parts of objects that are occluded from the line of sight of the HoloLens™ should be scanned from an alternative angle such that they may be recorded by the HoloLens™.

2. **Surface Materials:** The HoloLens™ also relies on its infrared cameras for scanning spatial surfaces. The texture/material of real-world surfaces can greatly influence such scans, whereby dark surfaces that reflect tiny amounts of visible and infrared waves may not be properly scanned, particularly from afar. Shiny, reflective surfaces can introduce problems within a model as well as they create illusory reflections of real world surfaces.

3. **Scene Motion:** The HoloLens™ constantly adjusts to its environment as it moves around any place. If there are rapid changes within a given scene, the HoloLens™ may not be able to update the spatial scan quickly enough. Dynamic elements within an environment can introduce further hallucinations, as moving objects, such as people may not need to be included in a spatial scan.

4. **Lighting Interference:** Ambient infrared light in a scene while scanning spatial data may adversely affect the quality of the scan. Very shiny surfaces or light from outside a room (e.g. sunlight), as well as light reflected directly into the HoloLens™ camera may cause issues such as floating mid-air hallucinations and holes.

The primary issues with the HoloLens™ generated spatial data translate to the primary objectives for the spatial reconstruction phase. These include

1. Hole Filling

2. Hallucination Removal

3. Bias Removal

There are other minor issues with the HoloLens™ spatial mapping data, such as noise and outliers, which form secondary objectives for the spatial reconstruction phase.

4.2 Spatial Reconstruction Algorithms

This subsection outlines some possible algorithms and approaches to solving the spatial reconstruction problem. None of the algorithms individually solve all the required objectives, so a combination of these have been utilized in the final version of the PRPU, whose design is detailed in the next section.

4.2.1 Convex Hull [4] and Concave Hull [32]

The convex hull algorithm essentially generates a closed convex hull around a set of points such that the hull encapsulates all the points. Mathematically, this means the generation of the smallest super set of points forming a closed hull such that the original set of points is a subset of the super set. This makes the algorithm quite useful for tasks such as hole filling in flat surfaces, but it deteriorates in performance given noisy data, which is a problem that plagues the data extracted from the HoloLens™. There are two common algorithmic implementations of convex hull: Chan's algorithm, which is suitable for calculation of the convex hull for sets of points in 2 and 3 dimensions, and the Quick hull algorithm[1], which is suitable for higher dimensions. The figure below demonstrates the result of the convex hull, where the black lines represent a mesh that incorporates the entire model (in blue).

The concave hull algorithm generates a closed concave hull around a set of points. This means the computation of an envelope for a set of points in 2D or 3D space. The concave hull for a set of points as compared to the convex hull provides a tighter bound with minimal area. The figure below vividly showcases the difference between the outputs of the two algorithms.

4.2.2 RANSAC Algorithm [33]

The RANSAC algorithm is primarily an algorithm for outlier detection and removal. This is analogous to drawing a line of best fit for a set of points in 2D.

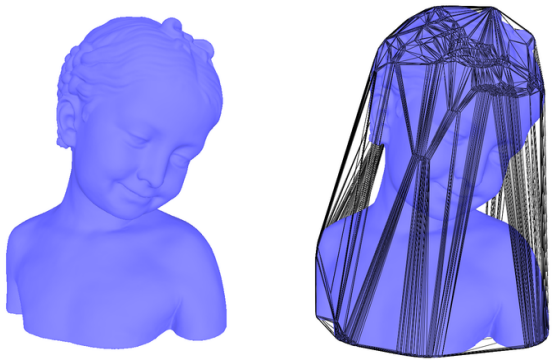
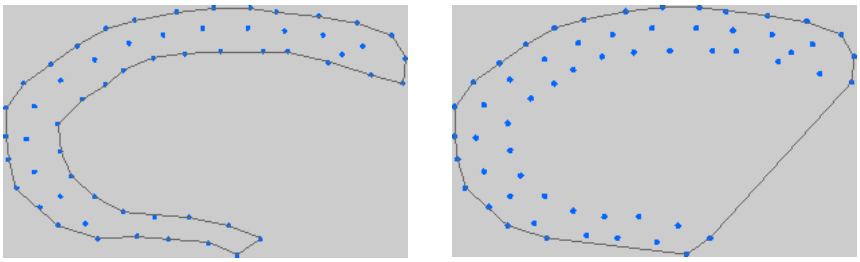


Figure 7: 3D Model (blue) with the convex hull wire frame (black)



(a) Concave Hull (b) Convex Hull

Figure 8: Concave and Convex Hulls

The RANSAC algorithm is the most widely used algorithm for plane detection. It effectively finds the best fit plane given a set of 3D points, a function that is incredibly useful within the post processing steps as a part of cleaning the given dataset. RANSAC is an iterative algorithm, showing qualitatively better results as more iterations are performed. The primary use within the PRPU is for plane detection. The current project is optimized for indoor environments with planar boundaries. That is to say that it focuses on the detection and reconstruction of planes, which results in comparatively smoothed surfaces for the outer hull of a model, alongside serving hole filling purposes as each plane can be individually reconstructed. The PRPU utilizes the RANSAC algorithm for shape detection via two primary functions - Point Set Detection and the Advancing Front algorithm. Point set detection solely detects planes utilizing an input set of 3D vertices and vertex normals. It then allows the output of the detected planar shapes either as individual files or as a single file combining

all the detected shapes. The advancing front function implemented within the PRPU performs plane detection using RANSAC as a first step. It then proceeds to reconstruct the surface facets for the model, details of which are outlined in the subsection on Advancing Front 4.2.5. RANSAC has been implemented within the PRPU using both PCL and CGAL. However, the results produced by CGAL are considerably better than the ones generated by PCL, alongside the added benefit of CGAL being able to handle facet information as well.

4.2.3 Poisson Surface Reconstruction [23] [27]

Poisson surface reconstruction is a novel method for surface reconstruction that considers the entire point cloud dataset under consideration as opposed to sampling. This makes the algorithm highly resilient to noise within the data, and hence a suitable choice to use for smoothing operations within the post processing steps. The algorithm introduces a novel way of approaching the problem of surface reconstruction by expressing the problem as a Poisson equation. This involves an implicit function framework whereby an indicator function is computed, classifying points as either being inside (a value of 1) or outside (a value of 0) the model. The algorithm then reconstructs the surface by extracting an appropriate isosurface from the processed data. The figure below showcases the steps involved in Poisson surface reconstruction in 2D space.

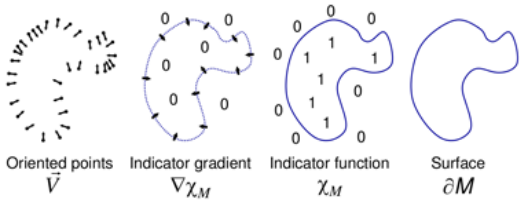


Figure 9: Poisson Surface Extraction in 2D space

Poisson surface reconstruction has also been implemented within the PRPU. The current CGAL based implementation computes the implicit Poisson function for the input point and normal set. This is carried out by a conjugate gradient solver that is represented as a piece wise linear function stored on a 3D Delaunay mesh. The algorithm then meshes the function based on user defined parameters and finally outputs the result in a polygon mesh. Details of the CGAL data structures, the parameters for PRPU algorithms and results are all

detailed under the PRPU section 5.

4.2.4 VoxelGrid Filter [30]

The VoxelGrid Filter is a function provided by the PCL library, which serves to downsize a dataset. This is particularly useful because the data obtained from the HoloLens™ can be quite noisy. The algorithm essentially works by dividing the given 3D space into tiny cuboids, or voxels, and approximating all points lying within a voxel to the voxel's centroid. Apart from reducing noise in the data sample, the downsizing also benefits the post processing steps by allowing for enhanced performance. Figure 10 below showcases the filter in action: the image on the left, a point cloud of a few hundred thousand points, is downsized by a factor of almost 10 in the image on the right, while still maintaining a suitable approximation of the original data.

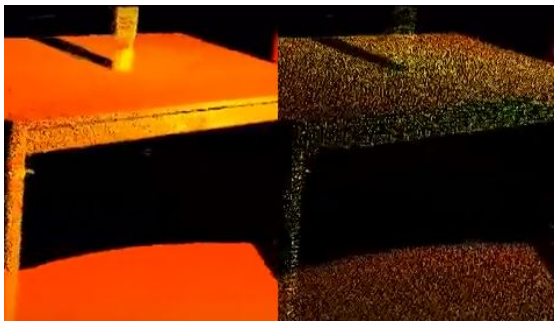


Figure 10: Poisson Surface Extraction in 2D space

4.2.5 Advancing Front Algorithms and Delaunay Triangulation [6]

Advancing front algorithms are used for piecewise surface reconstruction. The cited paper introduces an algorithm that ensures a smooth variation of elements and results in a triangulation that constitutes the Delaunay triangulation of the points as well. The proposed greedy algorithm creates an unstructured mesh by stitching triangles based on their utility to fill holes. The algorithm combines the two approaches of advancing fronts and Delaunay triangulation, and leverages their complementary nature to perform surface reconstruction. Delaunay triangulation attempts to enforce a particular connectivity associated with a set of points that leads to desirable results, while an advancing front

algorithm introduces a point placement strategy while maintaining an order for element generation. Each new triangle constructed to be stitched to the existing mesh needs to share an edge with the current, advancing edge of the mesh. This leads to a number of different possibilities the algorithm encounters while generating the surface, some of which are showcased in figure 11 below.

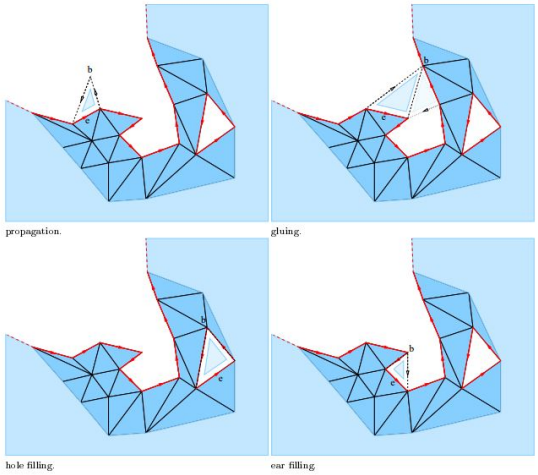


Figure 11: Possibilities encountered while stitching the next triangle

The advancing front algorithm, in tandem with RANSAC for plane detection, has been implemented within the PRPU, and has produced high quality results compared to other algorithms. These details are laid out within the PRPU section 5.

5 PRPU

This section outlines the design of the PRPU, alongside details of the implemented surface reconstruction algorithms as well as results produced by the PRPU algorithm for real world models scanned via the developed HLA.

5.1 System Design

The PRPU has been designed in two parts. The core processing unit has been developed in C++, which incorporates the spatial reconstruction algorithms necessary. The C++ project handles command line arguments as input for

processing any spatial data file. This project has been wrapped in a Managed C++ class, which finally gets compiled as a DLL, alongside all the necessary dependencies [19] which include CGAL, STL, Boost, GMP and MPFR and Qt. The DLL is then used, via the command line arguments, by a C project that runs on Azure as an App Service. The C project implements a REST API to communicate with the HoloLens™ while the HLA is running. Azure is also utilized for its blob storage, where the HLA pushes and pulls recorded and processed models respectively.

5.1.1 Project Structure

The current implementation of the C++ project for processing is split over 8 files:

1. **CommandInterface.hpp** - Houses the class and function declarations for the **ProcessXYZ** class. Also includes the relevant header files that the class depends on.

2. **CommandInterface.cpp** - Houses the class and function definitions corresponding to the declarations in the CommandInterface.hpp file.

3. **ModelBuilder.hpp** - Houses the class and function declarations for the **ModelBuilder** class. Also includes the relevant header files that the class depends on, as well as type definitions to be used within the implementation of the ModelBuilder class.

4. **ModelBuilder.cpp** - Contains the class and function definitions for the ModelBuilder class declared in ModelBuilder.hpp.

5. **Definitions.hpp** - Contains the required CGAL includes as well as type definitions that can be used during the implementation of any of the classes used within the project.

6. **polyhedronBuilders.h** - Defines two classes - **polyhedron_builder** and **mesh_polyhedron_builder**.

7. **cgalProcessing.h** - Contains the class declarations for the **CGALProcessing** class.

8. **cgalProcessing.cpp** - Contains the implementation details for the CGALProcessing class declared in the cgalProcessing.h header file.

5.1.2 Project Dependencies

The processing unit utilizes a number of external libraries for processing. These include:

1. **CGAL [3]** - CGAL is a software project that provides access to efficient and reliable geometric algorithms packaged as a C++ library. The library offers data structures and algorithms like triangulations, Voronoi diagrams, Point Set processing, Surface and Volume Mesh Generation, Geometry Processing, Convex Hull algorithms, amongst others. CGAL is heavily relied on by the PRPU for spatial reconstruction purposes.

2. **Boost [2]** - Boost provides free, open source and portable C++ libraries. The library functions are relied on both by the PRPU and by CGAL.

3. **GMP [20]** - GMP is a free library for arbitrary precision arithmetic operating on signed integers, rational numbers and floating point numbers.

4. **MPFR [21]** - The MPFR library is a C library for multiple-precision floating point computations with correct rounding. Both GMP and MPFR are used within CGAL for geometric and arithmetic operations, as part of its Exact Computation Paradigm.

5. **Qt [28]** - Qt is a cross plaform application development and UI framework. It is relied upon by CGAL for a number of modules and demos.

5.1.3 Class Design

The core processing unit of the PRPU comprises of three main classes. Diagram 12 shows the relationship between these classes and the class declarations.

5.1.3.1 ProcessXYZ The main control class for the processing unit is the ProcessXYZ class. The class maintains global variables to control the algorithms to run and for file IO. The main function of the class parses command line arguments via a 'Flags.hh' file that provides argument parsing functionality. The main workflow for a given model is controlled by the following functions:

1. **setInput()** - sets the input file name and path for the model to be processed.

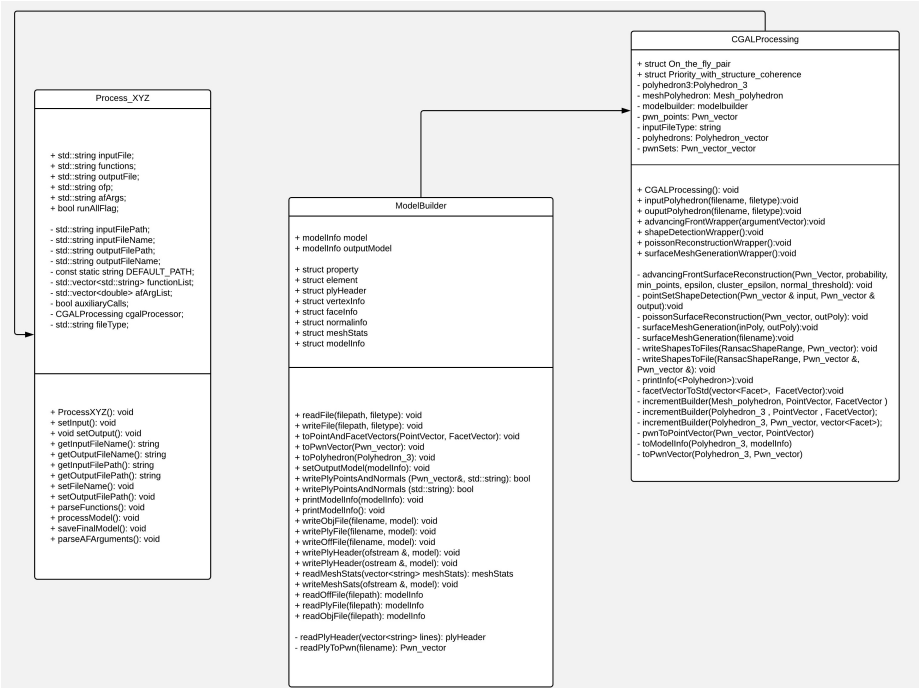


Figure 12: Class Diagram for the processing unit

2. **setOutput()** - sets the output parameters for the model to be processed.

3. **parseFunctions()** - parses the function names provided as part of the command line arguments.

4. **parseAFArguments()** - parses the algorithm parameters for the Advancing Front algorithm.

5. **processModel()** - runs all required algorithms on the input model according to the function list provided. Currently this includes Advancing Front (AFR), Point Set Shape Detection (PSD) and Poisson Surface Reconstruction (PSR).

6. **saveFinalModel()** - saves the final processed model to disk.

5.1.3.2 ModelBuilder The ModelBuilder class serves two primary purposes - it acts as a custom data structure to store spatial mapping data, allowing conversion to and from all other spatial data structures used within the PRPU,

and provides I/O functions for OBJ, PLY and OFF file formats. Section 5.1.5 later details the input and output processes for the PRPU. The ModelBuilder class defines a struct **modelInfo** to store spatial mapping data, including vertices, vertex normals, faces and meshStats. meshStats is a struct defined to handle meta information generated by the Spatial Understanding API within the HLA. The class provides conversions to PointVector, FacetVector, Pwn_vector and Polyhedron_3 types for use with the CGAL library.

5.1.3.3 CGALProcessing The CGALProcessing class defines the actual algorithms to be run for spatial reconstruction purposes. The class mainly deals with two data structures that store spatial mapping data. These are **Pwn_vector** and **Polyhedron_3**. The Pwn_vector is a vector of pairs of points and vectors i.e. vertices and vertex normals. Polyhedron_3 is a data structure created by CGAL to store polyhedral surface meshes. Details of the structures, algorithms and constraints on the CGAL library are outlined in section 5.1.4 later.
The CGALProcessing class defines three algorithm wrappers for spatial reconstruction:

1. **Advancing Front Reconstruction [18]** - The advancing front reconstruction wrapper implemented within the CGALProcessing algorithm accepts as input a Pwn_vector. It performs plane detection using the RANSAC algorithm, based on the following parameters [**efficientRansacParameters**]:

   (a) **probability:** The probability to control search endurance. The default value is set as 0.05.

   (b) **min_points:** The minimum number of points of a shape. This controls the termination of the algorithm. The default value is dynamically calculated to equal 1% of the total number of input points.

   (c) **epsilon:** The is the maximum tolerance of the Euclidean distance from a point to a shape. The default value is equal to 1% of the length of the bounding box diagonal for the shape.

   (d) **cluster_epsilon:** The maximum distance between points for them to be considered connected. The default value is equal to 1% o the length of the bounding box diagonal for the shape.

(e) **normal_threshold:** The maximum tolerance normal deviation from a point's normal to the normal on a shape at a projected point. The default value is 0.9 (around 25 degrees).

Following the plane detection phase, the algorithm generates a structured point set, and runs the advancing front algorithm on the structured point set. The structured point set generation includes resampling the planar sections and edges and corners included as part of the plane detection phase. The current implementation of the algorithm includes the utility of a Priority_with_structure_coherence struct, which acts as a priority functor to the algorithm that favors structurally coherent facets, making the advancing front algorithm robust to sharp features. Results on multiple test sets with varying parameters are showcased in section 5.1.6 later.

2. **Point Set Shape Detection [cgalPointSetManual]** - The point set shape detection wrapper within the CGALProcessing class implements the RANSAC algorithm for plane detection, similar to the advancing front wrapper. This algorithm however does not attempt to reconstruct surface facets from the input Pwn_vector. The wrapper allows the output of the detected planes individually as point sets, or as a combined point set. This function is useful for the detection of planar surfaces, possibly to reconstruct a model's outer hull. Outer hull construction can be performed by the conversion of the Pwn_vector to CGAL's Polyhedron data structure, a feature which is already implemented within CGALProcessing. Section 5.1.6 below showcases some examples of point set shape detection.

3. **Poisson Surface Reconstruction: [27]** - The Poisson surface reconstruction method accepts as input point sets with oriented normals. It then computes an implicit Poisson function. The Poisson surface reconstruction function implemented within CGAL primarily relies on the size parameter **spacing**. A reasonable argument for the size is the average spacing of the input point set. Other parameters for the poisson reconstruction include:

   (a) **spacing** - the size parameter.

   (b) **sm_angle** - bound for the minimum facet angle in degrees.

(c) **sm_radius** - bound for the radius of the surface Dalaunay balls (relatively to the average spacing).

(d) **sm_distance** - bound for the center to center distances relative to the average spacing.

A primary problem with Poisson surface reconstruction, however, is that it cannot handle sharp and 1 dimensional features. That results in loss of proper representation of edges and corners in 3D models, which is undesirable for indoor environment models. The poisson surface reconstruction method can be utilized for smaller models with less sharp features, such as small connected components inside a 3D model.

5.1.4 CGAL

This section outlines some of the packages offered by CGAL that can or have been utilized for spatial reconstruction purposes. Such packages include, but are not limited to:

1. **3D Convex Hulls [22]** - provides algorithms for convex hull generation in 3D space.

2. **3D Polyhedral Surface [24]** - the package introduces a data structure to store polyhedral surfaces that are composed of vertices, edges, facets and an incidence relationship on them. The underlying organization is a halfedge data structure, which restricts the class of representable surfaces to orientable 2-manifolds. Such surfaces that are closed are called Polyhedrons. The package introduces the Polyhedron_type, which is extensively used by the PRPU to store spatial data during processing.

3. **Surface Mesh [26]** - the package introduces the Surface_mesh class as an implementation of a halfedge data structure that can be used to represent a polyhedral surface. This serves as an alternative to the 3D Polyhedral Surface package, with the main difference being that this is index based as opposed to being pointer based.

4. **3D Surface Mesh Generation [29]** - the package provides a function template to compute a triangular mesh approximating a surface. The meshing algorithm is based on the notion of the restricted Delaunay triangulation. The algorithm basically computes a set of sample points on

the surface, and extracts an interpolating surface mesh from the 3D triangulation of the sample points.

5. **3D Mesh Generation [cgal3DMeshGeneration]** - the package allows the generation of isotropic simplical meshes over 3D domains. The domain has to be a region of 3D space and be bounded. The region may contain a single or multiple connected components. The mesh generation process outputs a 3D triangulation, including subcomplexes that approximate input domain features such as subdomain, boundary surface patches or input domain features with 0 or 1 dimension.

6. **Poisson Surface Reconstruction [27]** - this package provides functions to reconstruct a surface following the computation of an implicit Poisson function for an input point set.

7. **Advancing Front Surface Reconstruction [18]** - the package provides functions to reconstruct surfaces using the advancing front algorithm.

8. **Polygon Mesh Processing [31]** - the package implements a collection of methods and classes for polygon mesh processing. The package outlines functions that can be used with any class model of the concept FaceGraph. This includes both Polyhedron_3 and Surface_mesh. Available processing algorithms include algorithms for meshing, coreinement and boolean operations, hole uilling, normal computation, connected components, among others.

9. **Point Set Shape Detection [cgalPointSetManual]** - the package provides algorithms (including a RANSAC implementation) for detection of various shapes in input point sets, such as planes, cylinders, tori and spheres.

All the aforementioned packages provide algorithms and data structures that can be used for some aspect of spatial reconstruction. The most important data structure introduced by CGAL that is used in the PRPU is the Polyhedron_3 [25]. Polyhedron_3 representation consists of vertices V, edges E, facets F and an incidence relation on them. The underlying data structure is a halfedge data structure, where each edge is represented by two halfedges with opposite directions.

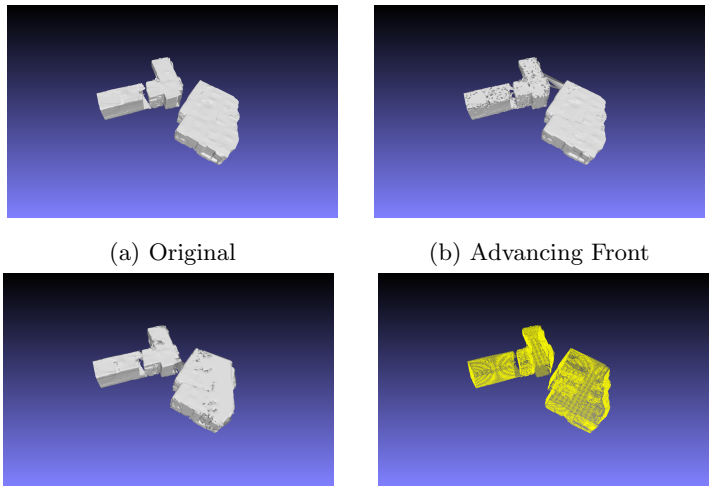
5.1.5 Input and Output

The primary input and output format that is followed within the PRPU and HLA is the OBJ file format.
The main I/O format used within CGAL is the OFF file format. The Model-Builder class implements file readers and writers for the OBJ, PLY and OFF file formats, alongside converters to and from the Polyhedron_3 class.
The OBJ file is the final output generated via the PRPU, which is then pulled from Azure by the HoloLens™ for visualization. Meta information about the model is included by the HLA, which is preserved even after the processing by the PRPU. This information includes:

1. Total Surface Area
2. Horizontal Surface Area
3. Up Facing Surface Area (Floor)
4. Down Facing Surface Area (Ceiling)
5. Wall Surface Area
6. Virtual Wall Surface Area
7. Virtual Ceiling Surface Area

The actual spatial information from an OBJ file that the PRPU or HLA uses includes:

1. **Vertex information** - This includes each vertex represented by 3 real values representing the Cartesian coordinates of a vertex in 3D Euclidean space.
2. **Vertex Normal information** - This includes a vector with 3 real values showcasing the direction of the normal corresponding to a given vertex.
3. **Facet information** - This includes a list of vertex indices that are incident to each given facet.



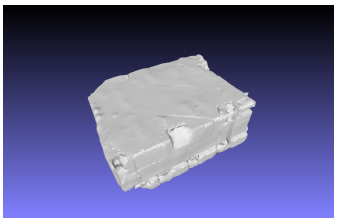
(a) Original (b) Advancing Front

(c) Advancing Front (d) Point Set Shape Detection

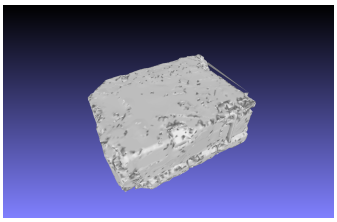
Figure 13: 312BL

5.1.6 Spatial Reconstruction Results

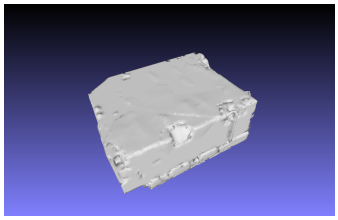
This section showcases the results of the Point Set Shape Detection wrapper, as well as the Advancing Front Surface Reconstruction wrapper. Four images are presented for each case - subfigure A represents the original mesh extracted from the HoloLens™. Subfigure B represents the model processed via the Advancing Front wrapper using the following parameters for the RANSAC algorithm - Probability (0.05), Minimum points (100), Epsilon (0.005), Cluster Epsilon (0.05), Normal Threshold (0.8). Subfigure C represents the model processed via the Advancing Front wrapper using the following parameters for the RANSAC algorithm - Probability (0.05), Minimum points (1% of total number of points), Epsilon (1% of length of bounding box diagonal), Cluster epsilon (1% of length of bounding box diagonal), Normal Threshold (0.9). Subfigure D represents the model processed using the point set shape detection wrapper, showcasing the planar shapes detected from the input point cloud.



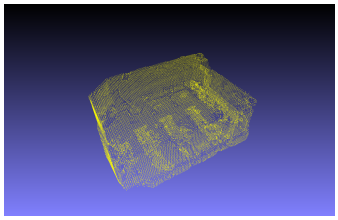
(a) Original



(b) Advancing Front

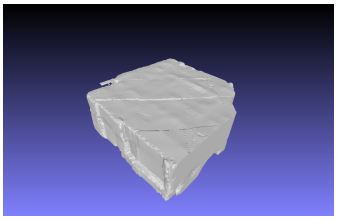


(c) Advancing Front

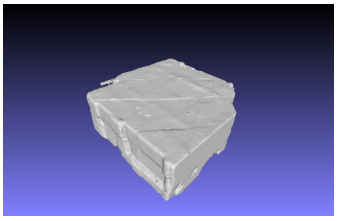


(d) Point Set Shape Detection

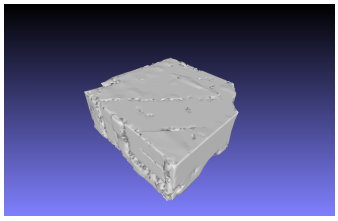
Figure 14: 312BR



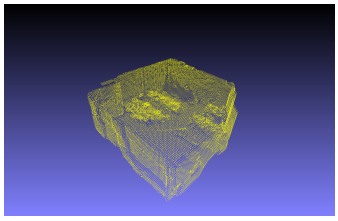
(a) Original



(b) Advancing Front

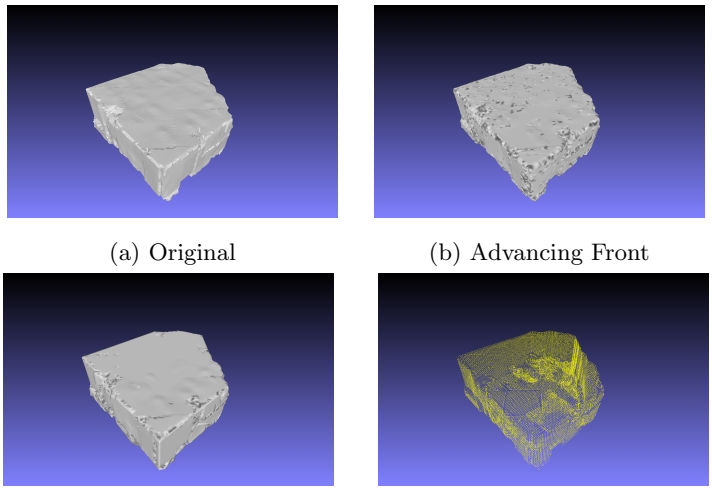


(c) Advancing Front



(d) Point Set Shape Detection

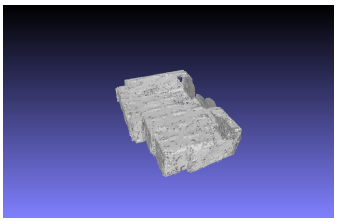
Figure 15: 312TL



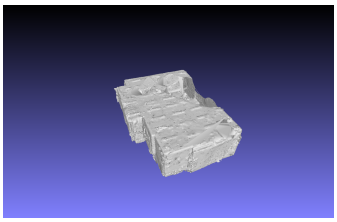
(a) Original (b) Advancing Front

(c) Advancing Front (d) Point Set Shape Detection

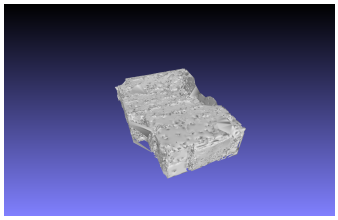
Figure 16: 312TR



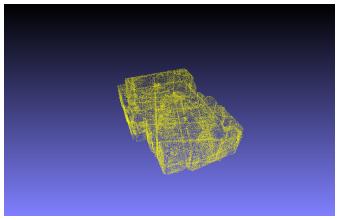
(a) Original



(b) Advancing Front

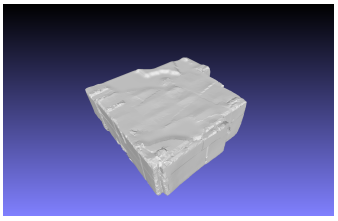


(c) Advancing Front

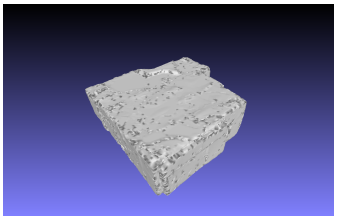


(d) Point Set Shape Detection

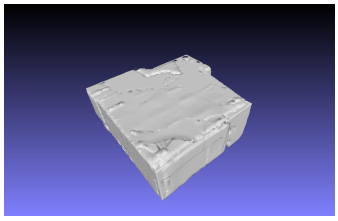
Figure 17: 335 - Old



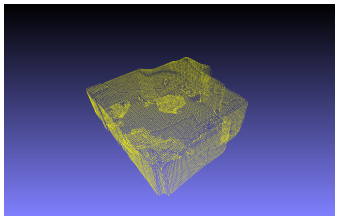
(a) Original



(b) Advancing Front

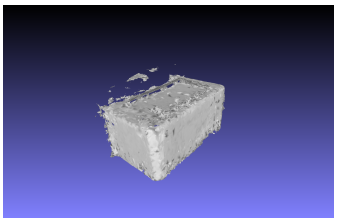


(c) Advancing Front

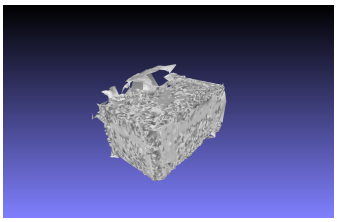


(d) Point Set Shape Detection

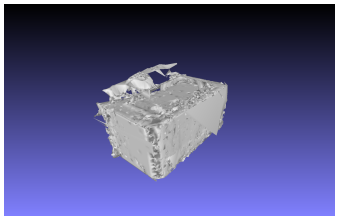
Figure 18: Haking Wong Building - Room 335



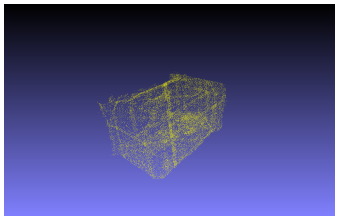
(a) Original



(b) Advancing Front

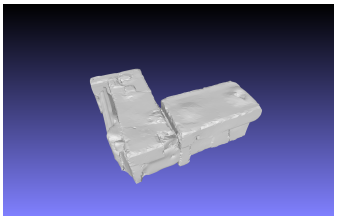


(c) Advancing Front

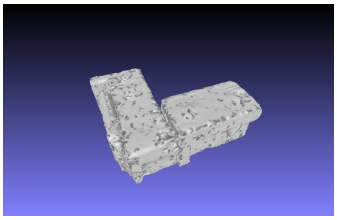


(d) Point Set Shape Detection

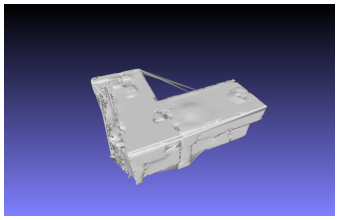
Figure 19: Chi Wah Room 11



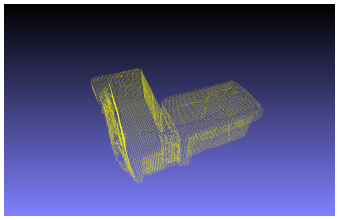
(a) Original



(b) Advancing Front

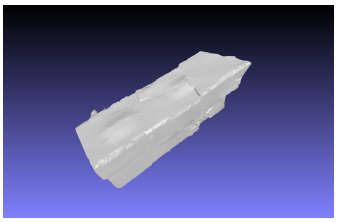


(c) Advancing Front

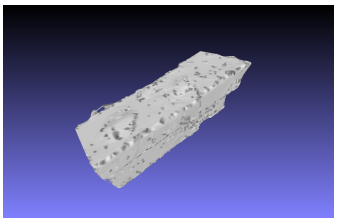


(d) Point Set Shape Detection

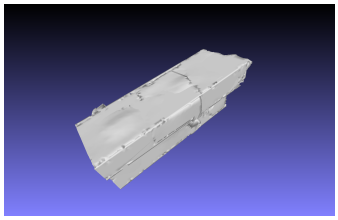
Figure 20: Corridor L1



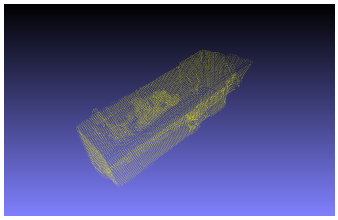
(a) Original



(b) Advancing Front

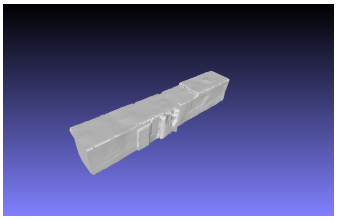


(c) Advancing Front

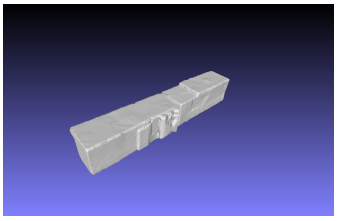


(d) Point Set Shape Detection

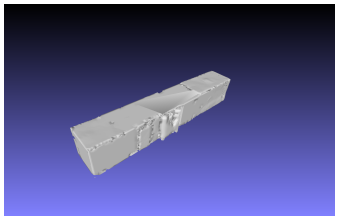
Figure 21: Corridor L2



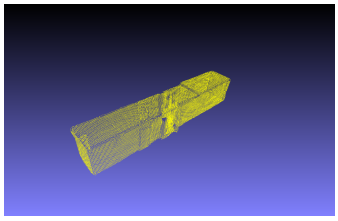
(a) Original



(b) Advancing Front

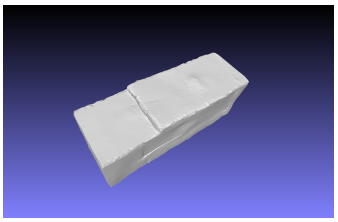


(c) Advancing Front

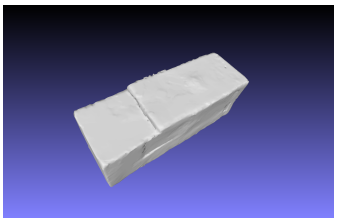


(d) Point Set Shape Detection

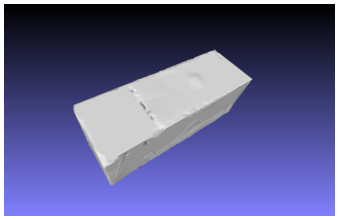
Figure 22: Corridor R1R2



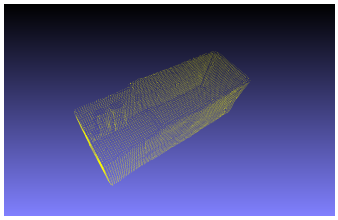
(a) Original



(b) Advancing Front

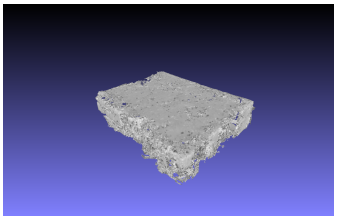


(c) Advancing Front

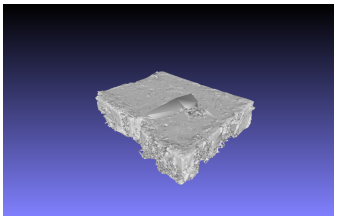


(d) Point Set Shape Detection

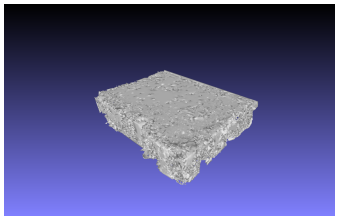
Figure 23: Corridor R1



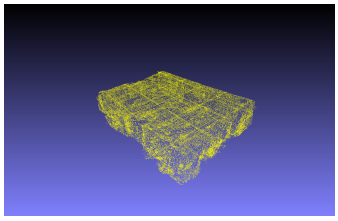
(a) Original



(b) Advancing Front



(c) Advancing Front



(d) Point Set Shape Detection

Figure 24: Dot Net Laboratory

6 Limitations and Difficulties

The HoloLens™ is one of the most advanced AR devices available commercially, but still suffers owing to the quality of its spatial mapping data. The granularity of the triangular meshes that the device can capture is rather coarse, which results in the misrepresentation of rough and curved surfaces. This bias adds to the complexity of the post processing steps, which regardless have to handle other issues such as holes and hallucinations. The team has focused the current development of the project to handle flat surfaces well, incorporating plane detection and point set structuring to aid the process.

Desipte being highly advanced, algorithms to stitch and continuously update sections of any environment are not perfect. This imperfection gets compounded if the primary user's movements are erratic or fast. While visual feedback will be provided to the user indicating the parts of an environment that have been captured, there is still room for error on the user's end as well as during the mesh processing stage of the whole process.

The HoloLens™ also presents a limitation to the scanning distance, supporting distances between 0.8m and 3.5m. This could present problems for a user wishing to scan a large indoor environment with high ceilings. the HoloLens™ also limits the area that it can record in one instance to $16m^2$. This limitation was a major setback to the team, but the team has explored the possibility of stitching multiple models together as a temporary solution to the problem. This process can be automated to completely eliminate the issue.

Owing to the complex nature of the problem the team attempted to solve, alongside the reliance on complex libraries such as CGAL, the team experienced delays and setbacks during the development stage as well. Additional spatial reconstruction algorithms as well as UI/UX enhancements to the HLA and Web Portal had to be pushed back to work on the core processing components.

Despite numerous challenges, the team managed to fulfill its project objectives by the successful development of an HLA accompanied by a PRPU. The team is confident though that the solution under development can be modified at a later stage following the release of the next iteration of the HoloLens™ by Microsoft, which should introduce improvements to both the hardware and software of the device, resulting in higher quality spatial data captured via the HoloLens™.

7 Future Works

The problem of 3D spatial reconstruction is a highly complex one, and there are numerous enhancements that can be built on top of the current solution that the team has developed. Currently, the 3D models generated only include geometric information about the vertices, vertex normals and facets. The final model can be enhanced by the inclusion of color captured via the HoloLens™, as well as surface textures. Such information may need to be extracted manually via images captured using the HoloLens™, because the device's spatial mapping capabilities do not include such information as yet. There is also scope to train a machine learning algorithm to understand and allow the manipulation of various indoor objects within 3D models. Such objects generally include pieces of furniture such as tables and chairs. Semantic analysis can be performed on indoor objects, allowing for the labelling and categorizing of the data. A key enhancement that can be made to the HLA is the inclusion of the spatial data processing algorithms within the application itself. This would allow the user to visualize, in real-time, the effects of running the various algorithms. Further enhancements to the HLA include advanced model manipulation capabilities, as well as the option to collaborate on a single model using multiple HoloLens™ devices.

8 Conclusion

The team has successfully developed a proof of concept for 3D spatial reconstruction using the Microsoft HoloLens™, a method that has not been realized elsewhere in any commercial capacity. The application has numerous benefits for engineers and designers, providing a convenient method to visualize 3D models. Future enhancements to the HLA and PRPU would only serve to enhance such convenience. The three main deliverables of the project all fulfill the relevant project objectives, as have been described in this report. The HLA provides capabilities to Record, Save and Visualize 3D models, while the PRPU fulfills the spatial reconstruction objectives of hole filling and surface reconstruction. The PRPU does exhibit some issues within the deployment environment using Azure™, and the team is communicating with Azure's support team to debug such issues. The HoloLens™ itself imposes some restraints on the spatial data

that can be captured, related both to the area that can be captured in one instance as well as the $3.1m$ distance restriction for capturing spatial data. The team is confident, however, that future iterations of the HoloLens™ would address these issues and the current project could be repurposed to utilize the spatial data from the new devices.

The proposed solution holds immense potential for industrial and commercial applications such as construction and interior design. The HoloLens™ itself allows for enhanced productivity for such applications by allowing a user to interact with an accurate hologram that can be superimposed with detailed design data, as well as support for multiple users to interact with the same model and manipulate models while discussing possible designs.

This report details development and implementation details of a solution to construct 3D models within a very short time span using the HoloLens™, coupled with mesh processing algorithms and a web portal to enhance user experience and the possible use cases of the solution.

Quality of the processed models remains an issue, though the input geometry (i.e. quality of captured spatial data) greatly affects the quality of the processed output model. Future iterations of the device, providing spatial data with greater resolution and accuracy, should automatically help to curb the issue. Various noise-removal and filtering algorithms can be applied to the input models to remove outliers and minimize unexpected behavior by the algorithm, but such steps can far more reliably be conducted by a user via the web portal so as to be able to visualize the algorithm's output immediately.

The team's results showcase great promise for the further development of the application, and the team agrees that the HoloLens™ serves as a sufficiently reliable and accurate means of capturing 3D spatial data.

References

[1] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. *The quickhull algorithm for convex hull*. Tech. rep. Technical Report GCG53, The Geometry Center, MN, 1993.

[2] Boost. *Boost C++ Libraries*. URL: https://www.boost.org/.

[3] CGAL. *The Computational Geometry Algorithms Library*. URL: https://www.cgal.org/.

[4] Bernard Chazelle. "An Optimal Convex Hull Algorithm in Any Fixed Dimension". In: (). URL: https://www.cs.princeton.edu/~chazelle/pubs/ConvexHullAlgorithm.pdf.

[5] CNR. *MeshLab*. URL: http://www.meshlab.net/.

[6] David Cohen-Steiner and Frank Da. *A Greedy Delaunay Based Surface Reconstruction Algorithm*. Tech. rep. RR-4564. INRIA, Sept. 2002. URL: https://hal.inria.fr/inria-00072024.

[7] Microsoft Corporation. *Holograms 101*. URL: https://developer.microsoft.com/en-us/windows/mixed-reality/holograms_101.

[8] Microsoft Corporation. *Holograms 230*. URL: https://developer.microsoft.com/en-us/windows/mixed-reality/holograms_230.

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

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

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

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

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

[14] Microsoft Corporation. *MixedRealityToolkit*. URL: https://github.com/Microsoft/MixedRealityToolkit.

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

[16] Microsoft Corporation. *Spatial Mapping Design*. URL: https://developer.microsoft.com/en-us/windows/mixed-reality/spatial_mapping_design.

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

[18] Tran Kai Frank Da and David Cohen$_S$*teiner*. *Advancing Front Surface Reconstruction*. URL: https://doc.cgal.org/latest/Advancing_front_surface_reconstruction/index.html#Chapter_Advancing_Front_Surface_Reconstruction.

[19] Joachim Reichel Eric Berberich and Fernando Cacciola. *CGAL Manual*. URL: https://doc.cgal.org/latest/Manual/installation.html.

[20] GMP. *The GNU Multiple Precision Arithmetic Library*. URL: https://gmplib.org/.

[21] GNU. *The GNU MPFR Library*. URL: http://www.mpfr.org/.

[22] Susan Hert and Stefan Schirra. *3D Convex Hulls*. URL: https://doc.cgal.org/latest/Convex_hull_3/index.html#Chapter_3D_Convex_Hulls.

[23] Michael Kazhdan and Hugues Hoppe. "Screened poisson surface reconstruction". In: *ACM Transactions on Graphics (TOG)* 32.3 (2013), p. 29.

[24] Lutz Kettner. *3D Polyhedral Surface*. URL: https://doc.cgal.org/latest/Polyhedron/index.html#Chapter_3D_Polyhedral_Surfaces.

[25] Lutz Kettner. *3D Polyhedral Surface CGAL::Polyhedron-3¡Traits¿ Class Template Reference*. URL: https://doc.cgal.org/latest/Polyhedron/classCGAL_1_1Polyhedron__3.html.

[26] Philipp Moeller Mario Botsch Daniel Sieger and Andreas Fabri. *Surface Mesh*. URL: https://doc.cgal.org/latest/Surface_mesh/index.html#Chapter_3D_Surface_mesh.

[27] Gaël Guennebaud Pierre Alliez Laurent Saboret. *Poisson Surface Reconstruction*. URL: https://doc.cgal.org/latest/Poisson_surface_reconstruction_3/index.html.

[28] QT. *Qt*. URL: https://www.qt.io/.

[29] Laurent Rineau and Mariette Yvinec. *3D Surface Mesh Generation*. URL: https://doc.cgal.org/latest/Surface_mesher/index.html#Chapter_3D_Surface_Mesh_Generation.

[30] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 2011.

[31] Ilker O. Yaz Sébastien Loriot Jane Tournois. *Polygon Mesh Processing*. URL: https://doc.cgal.org/latest/Polygon_mesh_processing/index.html#Chapter_PolygonMeshProcessing.

[32] UBICOMP. *Concave Hull*. URL: http://ubicomp.algoritmi.uminho.pt/local/concavehull.html.

[33] Michael Ying Yang and Wolfgang Förstner. "Plane detection in point cloud data". In: 2010.