Electronics and Computer Science

Faculty of Engineering and Physical Sciences

University of Southampton

Immersive Audio-Visual Virtual Reality Scene Reproduction

Team
Adam Gafar
Atharv Sankholkar
Joe Boutros
Shubh Harshad
Sourish Mukherjee

Project Supervisor
Dr Hansung Kim

Second Supervisor
Dr Andersen ANG

A Group Design Project Report submitted for the Award of

MEng Computer Science with Artificial Intelligence
MEng Computer Science
MEng Computer Science with Artificial Intelligence
MEng Electronic Engineering with Artificial Intelligence
MEng Electronic Engineering with Industrial Studies

February 7, 2024

Abstract

Pertaining to the longstanding oversight of realistic audio in Virtual Reality (VR) development, this project introduces an automated end-to-end pipeline that pioneers the generation of a VR scene with realistic acoustic properties. Drawing inspiration from Dr Hansung Kim's seminal work and leveraging pre-trained models, the pipeline efficiently generates a 3D environment from a single omnidirectional image. This immersive environment is enriched with spatial audio properties, significantly enhancing user immersion in VR experiences. The pipeline operates within a groundbreaking 15-minute timeframe.

With only a single omnidirectional image as input, the pipeline achieves comparable audio and mesh quality to traditional stereoscopic image inputs. Moreover, the sounds generated within a replicated 3D environment closely resemble those found in the corresponding real-world environment, highlighting the virtual environment's faithful representation of acoustic properties. This achievement underscores the efficacy of the approach in bridging the gap in VR development, offering a streamlined and efficient solution for realistic audio integration.

Keywords: virtual reality, realistic audio, end-to-end pipeline, audio-visual reconstruction.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

The implementation uses open-source models, namely 360MonoDepth, EdgeNet360, and DBAT.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Acknowledgements

We would like to extend our gratitude to our supervisor Dr Hansung Kim for his guidance and support throughout this project. Collaborating with him has been truly enriching, and we appreciate the knowledge gained under his mentorship. Working alongside Dr Kim has been an honour.

Special thanks to Dr Yihong Wu for kindly providing us with an early release of the SliceFormer model.

Many thanks to Dr Atiyeh Alinaghi for kindly calculating the Room Impulse Response (RIR) values, which have been crucial to demonstrating the effectiveness of our solution.

We would like to dedicate our work to our beloved families.

Contents

1 Introduction

1.1 Motivation

In virtual reality (VR) systems, enhancing personalised audio-visual experiences is a paramount challenge, crucial for elevating the sense of presence. Human perception heavily relies on audio and visual cues to comprehend and engage with the virtual environment. The core objective of this project is to forge ahead in developing an immersive audio-visual VR system, effectively transposing real-world environments into the virtual space.

Focusing on the visual and audio aspects in the progression of VR fidelity aims to address the oversight of the significance of audio in previous work, thereby bridging a crucial gap in simulated environments. On a broader scale, the research underscores the impact of audio on enhancing user engagement [1] and attention [2]. On a more granular level, realistic audio integration within virtual reality opens opportunities across various domains, such as entertainment and education.

The motivation behind this project is to pioneer the development of a comprehensive pipeline capable of generating a VR scene with realistic visual and acoustic properties from a single omnidirectional image on a single device, ultimately optimising the user experience.

1.2 Project Specification

This project builds on research conducted by Dr Kim, as described in his seminal 2021 paper [3]. Building upon this research, and with the adoption of suitable pre-trained models, the objective of this project is to implement an end-to-end pipeline that takes an omnidirectional image as input and reconstructs and renders a real-time 3D environment in Unity, with simulated acoustic properties derived from the materials detected therein. The audio solution is implemented utilising the spatial audio package "Steam Audio" by Valve. As Dr Kim's approach used a multi-stereo camera setup to capture an indoor scene and calculate a depth map manually using computer vision techniques during post-processing, a time and cost reduction element exists in that the output for this project is to instead be obtained from any given omnidirectional image (produced by a consumer-grade camera).

In terms of the final result, this project is a proof-of-concept system that runs on a supplied machine (Dell Inspiron 16 Plus with an NVIDIA GeForce RTX 4060 Graphics Card), and a supplied VR headset (HP Reverb G2).

1.3 Project Plan

1.3.1 Goals and Scope

The goals of this project are:

1. To build an end-to-end pipeline that takes an omnidirectional image and returns a VR scene in Unity;
2. To demonstrate the efficacy of utilising omnidirectional images in conjunction with suitable depth and material recognition models – instead of stereo images and manual material assignment – as a means to attain realistic visuals and audio within the VR scene.

The scope of this project is building a pipeline specifically for omnidirectional images taken in an indoor setting where:

1. The objects can be: ceiling, floor, wall, column, beam, window, door, table, chair, bookcase, sofa, and board. Other objects are marked as clutter.
2. The materials can be: plaster, asphalt, soil, ceramic, metal, foliage, stone, food, wood, fabric, concrete, rubber, plastic, paper, glass, and water.

1.3.2 Requirements

Limited team correspondence with the external client, ETRI labs, ensued due to temporal disparities and workload constraints. Therefore, liaison and ratification of core deliverables for the project were conducted between Dr Kim and the ETRI labs in advance.

The project's delivery requirements are the following:

- Revise, adapt, and rigorously test essential components, encompassing the evaluation of alternative approaches and models.
- Incorporate monocular depth estimation within the pipeline to produce a depth map for semantic scene completion.
- Employ material recognition to autonomously associate materials and their corresponding acoustic properties with objects within the VR scene.
- Automate the importation of 3D models with acoustic properties into the game engine for acoustic room modelling and simulation.
- Assemble pipeline components to process omnidirectional image input from a single camera and generate a VR scene with acoustic properties capable of simulating spatial audio effects.
- Implement the pipeline system with a straightforward graphical user interface.

These requirements are complemented by the stretch goal of enabling user customisation for selected aspects of the pipeline and scene, such as using other depth map generation methods in the pipeline or modifying the sound source location within the VR scene.

1.3.3 System Diagram

The following diagram highlights the overall pipeline of the project. The Graphical User Interface (GUI) is the entry point where the user interacts with the system and passes in the omnidirectional RGB image. This input file is passed into the Monocular Depth Estimation module which analyses the visual information in the scene and generates a depth map. The RGB image is also passed into the Material Recognition Unit which analyses the content of the image and produces a material map that highlights the different materials present in the scene.

The depth map and material map generated in the previous parts are passed into the Semantic Scene Completion part of the pipeline for which a mesh is generated with the correct materials assigned to it. This part is imported and rendered in Unity with added acoustic properties depending on the materials in the scene to reconstruct the 3D scene.

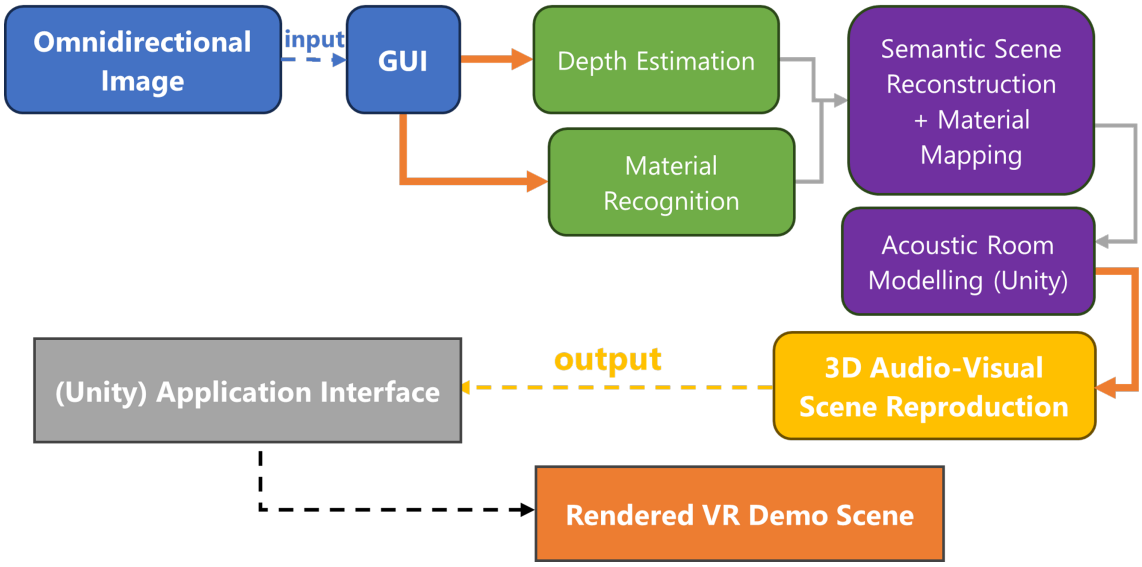


Figure 1.1: Diagram showing the flow of the project pipeline

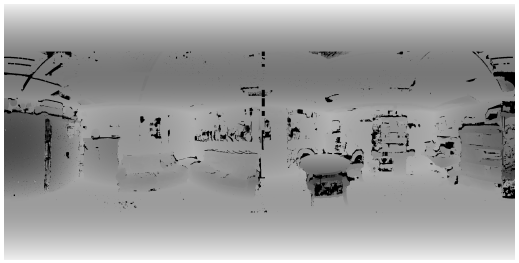
2 Background

2.1 Monocular Depth Estimation

2.1.1 Introduction

Monocular Depth Estimation (MDE) is a computer vision task that focuses on estimating the depth information of a scene using only a single monocular RGB image as the input. It does this by computing the depth value of every single pixel of an image. This method has significant importance in various applications, such as Augmented Reality, autonomous driving, and as in the case of this project, Virtual Reality.

(a) An indoor omnidirectional image reproduced from the EdgeNet test set, reproduced from [4].



(b) The indoor image's corresponding ground truth depth map, reproduced from [4].

Figure 2.1: Example of the expected input and output for Monocular Depth Estimation.

2.1.2 Mechanism

The principle behind Monocular Depth Estimation involves training models to learn the relationship between visual cues present in images and their corresponding depths. Convolutional Neural Networks have proven to be highly effective for this task. These networks are trained on large datasets of RGB images and their corresponding ground truth depth maps. During the inference phase, the trained model takes an RGB image as the input, produces a greyscale depth map, and then outputs that depth map represented by a colour map. The purpose of these colour maps is to visually convey the depth information encoded in the greyscale depth map by assigning different colours to the different depth levels. The colour map used for a Monocular Depth Estimation network varies based on purpose and from network to network.

360MonoDepth, serving as the core of this component, is used in the pipeline for the purposes of Depth Estimation. The output depth map generated by this is then passed in as the input to the next component of the pipeline for the purposes of 3D reconstruction.

2.1.3 Use Cases

Monocular Depth Estimation is used in scenarios where acquiring depth information using traditional methods, such as multiple stereo cameras, is impractical or challenging. In the context of this project, the aim is to be able to input any image and reconstruct it in VR, and as such this technique provides a more practical, cost-effective, and accessible solution, as it only requires a single RGB image and no other equipment.

2.1.4 Traditional Approach

The traditional approach for depth estimation uses two stereoscopic cameras to take images of the same space from a set distance apart, where a depth map of that space is computed afterwards using stereoscopic matching algorithms. While effective, this approach has limitations, including increased hardware complexity in addition to the need for precise calibration. The transition to Monocular Depth Estimation represents a potential paradigm shift in this project. The novel approach of Monocular Depth Estimation is thus particularly revolutionary because it increases the project's versatility by making it accessible to anyone with a device that can capture omnidirectional images.

2.1.5 Challenges

Within the framework of this project, a unique challenge emerges from the use of omnidirectional RGB images as the input to the pipeline instead of standard RGB images. Attempting to input omnidirectional images through a newer state-of-the-art depth estimation network such as MiDaS is not feasible. The disparities between the dataset used to train MiDaS, and the unique characteristics, such as the visual distortions, of 360 images introduce a misalignment of features, leading to inaccuracies in the depth estimation process. Consequently, addressing this challenge requires modifying and adapting depth estimation techniques in order to tailor them for omnidirectional images.

2.2 Semantic Scene Completion

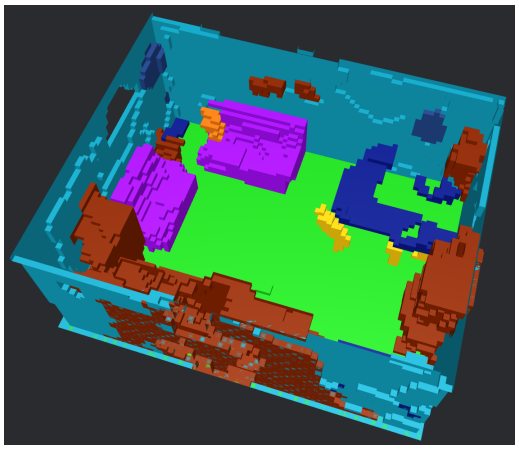
2.2.1 Introduction

Semantic Scene Completion (SSC) is an integral component in this project's pipeline. It serves as a crucial step following Monocular Depth Estimation, hence leading to the creation of a realistic 3D environment such as the one in the original input omnidirectional image. This section will explore the fundamental aspects of SSC, including its pipeline, its concept, and the role of EdgeNet.

2.2.2 The SSC Pipeline

The SSC pipeline involves taking in the original omnidirectional image of an indoor scene and in conjunction with the depth map generated using Monocular Depth Estimation, using this information to generate a 3D reconstruction of that scene. This part of the pipeline consists of two main components: Semantic Scene Reconstruction, and Semantic Scene Completion. Semantic Scene Reconstruction is the process of generating a 3D scene with assigned semantic labels, whereas Semantic Scene Completion focuses on predicting and filling in any anomalous missing parts of a scene or object while maintaining semantic consistency. Both of these components contribute to enhancing the user experience and interaction with the generated scene while maintaining as much of the original scene as possible.

(a) Indoor 360 image provided in the EdgeNet test set, reproduced from [4].



(b) Generated Voxel Grid using the RGB image in 2.2a and its depth map in 2.1b, reproduced from [4].

Figure 2.2: SSC Output with EdgeNet using RGB image and its ground truth depth map.

2.2.3 SSC with EdgeNet360

At the heart of the SSC pipeline is EdgeNet360. It is a method that makes Semantic Scene Completion possible from a single omnidirectional image and its corresponding depth map. It employs a Deep Convolutional Neural Network to infer the semantic labels of the entire room. It has been trained on existing datasets and is suitable for dynamic applications. [4]

EdgeNet360, a 3D Convolutional Neural Network inspired by the U-Net Design, refines and enhances predictions for challenging objects in the scene by utilising a surface volume along with a volume generated from the edges in the RGB image, hence taking the depth image as an input in addition to the original RGB image that it does. The constructed voxel grid is divided into eight views, before being individually processed by EdgeNet. It then puts these eight views together. The extension to EdgeNet360 by Simon Lisowski generates a corresponding mesh for a given voxel scene reconstruction.

2.2.4 Reconstructing Occluded Areas

An image is a 2-dimensional projection of a 3-dimensional scene. Therefore, not all points within that scene can be captured as some will be occluded. Since the input image cannot capture the full 3D information of a scene, SSC becomes highly relevant in reconstructing the occluded areas by inferring the missing semantic details of objects in a scene and filling them in. These predictive capabilities of SSC contribute to a more seamless and visually pleasing VR experience.

2.2.5 Interdependence of MDE and SSC

Monocular Depth Estimation and Semantic Scene Completion are both tightly coupled aspects of the project's pipeline, working hand in hand to transform an omnidirectional image into an immersive 3D environment. Monocular Depth Estimation provides the initial depth map, providing the Semantic Scene Completion model (EdgeNet) with an initial understanding of the scene's geometry. SSC then takes this information and refines the generated construction. The success of SSC is inherently linked to the accuracy of the depth map generated by Monocular Depth Estimation. It serves as a crucial input for SSC and thus nailing depth estimation is imperative for SSC to function optimally and deliver a convincing and coherent 3D reconstruction.

2.3 Material Recognition

2.3.1 Introduction

The purpose of material recognition is to identify the material category (e.g. fabric, plaster, wood, etc) for each pixel in an input image. In the overall pipeline of the 3D VR scene reconstruction, the role played by the material recognition module is that, based on the materials recognised for the objects present in an image the corresponding acoustic properties can be assigned to those objects. This makes the final rendered scene much more realistic and immersive, as when a sound source is played in the rendered scene it would sound as though it is being played in actual scene due to the reverberations caused, creating a spatial audio effect. This enhances user experience and makes them much more involved/immersed in the scene. Without effective material recognition, the rendered scene would not mirror the actual acoustic properties of the captured scene diminishing the user experience.

2.3.2 Background

In the 2021 paper by Dr Kim [3] for immersive audio-visual scene reconstruction, no model was used for the task of material recognition. Instead, the output 3D model from EdgeNet360 was directly imported into Unity and the objects identified in the 3D model were assigned the most similar material type based on the acoustic properties available from the acoustic-materials list of Google Resonance or Steam audio from visual identification [3]. The method involved visual identification of the material present in the 3D model and manual assignment, this makes the final process of the 3D scene reconstruction in VR cumbersome and not free-flowing for an end user as there is human effort involved. This manual method would also be

prone to incorrect assignment of material labels due to human error/carelessness causing inaccurate scenes to be reproduced.

Due to the above issues highlighted a model was required for the automation of identifying the materials present in an image and then automatic assignment of materials labels to the 3D model accurately. The material assignment to the 3D model is discussed in the material mapping section.

2.3.3 Dynamic Backward Attention Transformer

For the task of Material Recognition in this project, the Dynamic Backward Attention Transformer (DBAT) model proposed by [5] is used. Research was conducted to find if there were alternative models available that outperformed the DBAT model provided. However, there was a scarcity of specialised models documented in the literature for the task of material recognition. Among the very few models that were found, DBAT stood out as the most accurate and robust option with an accuracy of 86.5% on the LMD dataset, achieving the best performance among state-of-the-art models. The model was also developed by Dr Kim and his PhD student, Heng, therefore the authors were easily accessible for contact regarding issues with the model. Given the limited availability of reliable alternatives and the state-of-the-art performance metrics of DBAT, it was decided to use the provided DBAT model for material recognition. It is noted that DBAT does not take omnidirectional images as input, therefore requiring pre-processing and post-processing to adapt it to the project's use case.

The model cumulates cross-resolution patch features for material recognition. "DBAT takes cropped image patches as input and gradually increases the patch resolution by merging adjacent patches at each transformer stage, instead of fixing the patch resolution during training [5]". After which the "intermediate features extracted from cross-resolution patches are explicitly gathered and merged dynamically with predicted attention masks [5]". Using dynamic patch resolution instead of fixed patch resolution the model can capture a wide range of characteristics from the whole image and cross-resolution feature aggregation from feature maps helps extract features from both shallow and deep layers.

The backbone encoder, the dynamic backward attention module, and the feature merging module make up DBAT. Patch merging and window-based attention for retrieving cross-resolution feature maps are handled by the encoder. The Dynamic Backward Attention (DBA) module aggregates the cross-resolution feature maps retrieved by the encoder by predicting per-pixel attention masks. The DBA module extracts features that complement the final stage encoder output, which contains a global view of the image, under the direction of the feature merging module. To generate the material predictions, the combined features which include improved global and cross-resolution features are then fed into a segmentation decoder [5].

Upon feeding an input image, the DBAT model outputs a corresponding image with different colours (RGB) corresponding to the materials identified for each pixel. Each pixel in the image is assigned a specific colour from the colour plate shown in figure 2.3 based on the material identified for that particular pixel.

Figure 2.3: DBAT's colour plate. Reproduced from [5].

2.3.4 Cube Map Projection

When a flat image is projected onto a curved surface, or conversely, a curved image is projected onto a flat surface, this is known as an image projection. Several projection techniques can be used to map an omnidirectional (equirectangular) image to 2D images/surfaces.

A cube map projection takes an omnidirectional image/3D scene and projects it onto the six faces of a cube that surrounds the object or viewpoint. Each face of the cube can be unfolded into a 2D square, making up the cube map. Faces represent six directions: up, down, left, right, front, and back.

A cylindrical projection projects a sphere onto a cylinder that is wrapped around the sphere [6]. It introduces distortion into the projections as you move further away from the equator and towards the poles. The distortion is so prominent that it is generally used for understanding purposes rather than in practice such as practical maps [7].

Equirectangular projection maps the latitudes and longitudes of a sphere onto the coordinates of a grid. The projection results in "horizontal stretching increasing further from the poles, with the north and south poles being stretched across the entire upper and lower edges of the flattened grid [8]". Thus, this technique introduces distortion as well.

Distortion is not desired in the projected image as the DBAT model expects a normal 2D image as input. A distorted input image would yield poor material identification output. [9] compared 360-degree image projection techniques based on distortion caused by the techniques using objective quality assessment. It compared equirectangular projection, cube map projection and its variants, and cylindrical projection and its variants. They found equiangular cube map projection (a variant/modification of cube map projection) to have the best for 360-degree format conversions [9]. Based on this and the issues with distortion in the above-discussed techniques, simple cube map projection was chosen as the projection technique. The cube map projection is for splitting the equirectangular image into projected 2D images so that they can be fed to the model.

2.4 Automatic Material Mapping

2.4.1 Introduction

Kim et al.'s [3] implementation required the user to manually detect and assign material properties to the scene. This was a time-consuming process that required expertise in Unity whilst also being prone to human errors. Therefore, to simplify the process of generating the scene on unity, a major milestone of this project was

to automate the material assignment onto the scene. This followed 2 steps; material recognition and material mapping. Material recognition entailed segmenting the image into regions based on the material present in that area and was performed using the DBAT model. The automatic material mapping merged the material recognition pipeline with the Semantic Scene Completion pipeline. Material mapping automatically assigns material labels to each object in the recreated scene, hence helping assign the corresponding acoustic properties, making the recreated scene more realistic. The automation would reduce the arduous task of assigning material properties to the unity scene, and so improve the overall user experience, compared to the previous implementation.

The output of the DBAT contains the image with RGB values assigned to each pixel based on the material identified for it. The task of material mapping entails projecting the material map onto the 3D voxel scene. Therefore, this requires translating the information from the 2D image onto a 3D scene. There were 2 methods investigated to project the material information which are discussed in the next sections.

2D to 3D Image Projection

2D-to-3D projection of the material image onto the voxel scene generated was the first concept that was explored. Paper [1] discussed the process of converting 2D coordinates to reference world coordinates, before converting to voxel coordinates. This method provided inspiration to a similar method, where the projection was broken down into the following steps:

1. Converting the pixel coordinates into respective spherical coordinates. This would involve converting the Cartesian pixel co-ordinates (x,y,z) of the produced depth image into Polar co-ordinates(θ, ϕ).The depth image would initially be used because the pixel coordinates would be the same as the material image, and it contained depth data, required for the alignment of the new spherical map.

2. This spherical map would encompass the camera's location in the scene. Rays would be emitted from the camera to the voxel scene. These rays would be used to measure the distance between the voxel and the camera, as well as find the intersection between a ray and the spherical map.

3. The error between the calculated distance between the camera and voxel surface, and the depth value from the spherical map is measured. This error would be reduced by rotating the alignment of the spherical depth image until a minimum was found.

4. Using the conversion of spherical coordinate to voxel coordinate, the material value, represented by the RGB value of the pixel would be used to assign to the corresponding voxel.

3D to 2D Image Projection

Song et al. [10] approached volumetric encoding of the depth map data using Flipped- Truncated Signed Distance Function (TSDF). TSDF encodes each voxel with a distance value to its closest surface and whether the voxel is in free space

or occluded. Kim et al. [3] also implemented the Flipped-TSDF function within EdgeNet to construct the voxel scene.

Edgenet360 initially creates an empty voxel grid with the correct dimensions as the input scene. It then splits the empty voxel grid into 8 views. Each view is then fed into Flipped-TSDF implementation where it converts the voxel coordinates to relative world coordinates, based on the view that was fed in, before converting to pixel coordinates. This voxel-to-pixel mapping is used to retrieve the depth value, based on pixel coordinate, for that corresponding voxel. Using world coordinates and the depth value, a TSDF value is calculated and assigned to each voxel, indicating the distance from the surface and whether the voxel was inside an object, as seen in 2.4. Flags are allocated to each voxel, indicating whether it was out of the field of view, occluded, space or occupied (visible surface).

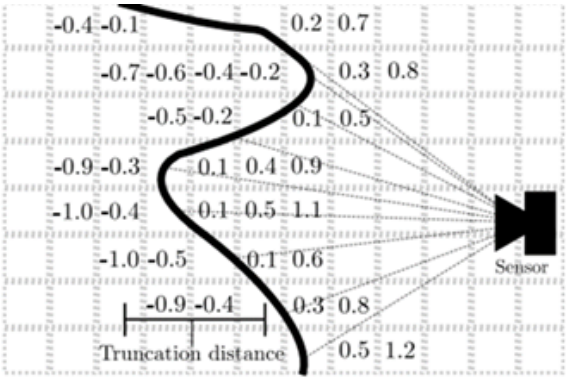


Figure 2.4: 2D depiction on how the TSDF function works, reproduced from [11].

For material mapping, the use of TSDF would be to create and initialise an empty voxel grid then iterate through every voxel in the grid and assign it a colour value based on the material identified in the material map. This method would exploit the code available from the EdgeNet360 repository, building on it to add this functionality.

Conclusion

The 3D to 2D projection method was chosen as the method for implementation due to various reasons. One of the main reasons why the 2D to 3D projection method was not suitable, was that it would require the pixel-to-voxel coordinate mapping to be done from the ground up, whereas the 3D-to-2D reconstruction had voxel-to-pixel mapping already implemented in EdgeNet360 and already created a voxel grid with objects based on a depth map. Therefore, it would only require adapting the EdgeNet360 model code to accept a material map image and then appropriately changing the code to map the material data to the voxel grid. Hence, this method would be less time-consuming, relying on adapting a similar technique with existing code available to build and exploit, making it a more viable option than implementing something from scratch. Especially for this project with a short time frame. Additionally, the latter method would be more efficient, as the prior method would be more time-consuming during the post-processing of the voxelated scene. 3D-to-2D image reconstruction was done using CUDA, which allowed for data parallelisation on the GPU, expediting the total run time of the 3D scene creation.

2.5 Acoustic Room Modelling

2.5.1 Technologies

Achieving accurate environmental acoustic modelling could be considered the overarching goal for the pipeline – the preceding modules' resultant 3D meshes with constituent recognised materials are the baseline from which the reproduction of realistic audio for a given scene can be calculated.

In the previous work performed by our supervisor, Dr Hansung Kim [3], both Google Resonance [12] and Steam Audio [13] were evaluated as open-source plugins to achieve this. Given the former's focus on providing spatialised audio for mobile AR environments, and Valve's commitment to VR, it was decided that Steam Audio was most suited for our project's purpose.

Steam Audio provides plug-ins for various game development environments, such as Unity and Unreal, that greatly abstract the setup and calculation of a scene's acoustics: objects can be tagged with specified acoustic properties to emulate real-world materials' dynamics, and sound sources can be customised with inbuilt features such as their implementation of the Head Related Transfer Function (HRTF) to better emulate spatialisation.

2.5.2 Evaluation Metrics

Evaluation of the accuracy of any acoustic modelling will typically compare the simulated Room Impulse Response (RIR) of an environment with its real-world source, which necessitates the configuration of extensive measuring equipment that was outside of the scope of this project. Thankfully, Dr Kim has performed experiments to measure the RIR data of a scene from which an omnidirectional image was provided as input to our pipeline – enabling quantitative evaluation of our project as detailed in the Results section.

The evaluation will focus on the replication of two RIR metrics:

- **RT60 (Reverberation Time, T_{60}):** A measure of the duration taken for a sound's pressure level to decay by 60 decibels, giving a general indication of the reverberation characteristics of an environment. A higher value corresponds to greater reverberation, with values over 2 seconds typically indicating an "echoic" room [14].
- **EDT (Early Decay Time):** A measure extrapolated from the early decay of a sound's pressure level by 10 decibels. Since early reverberations are much more impacted by position, the EDT is much more affected by listener position than RT60 values; lower values will correspond to a greater "clarity" of sound, despite a room potentially having high reverberation [15].

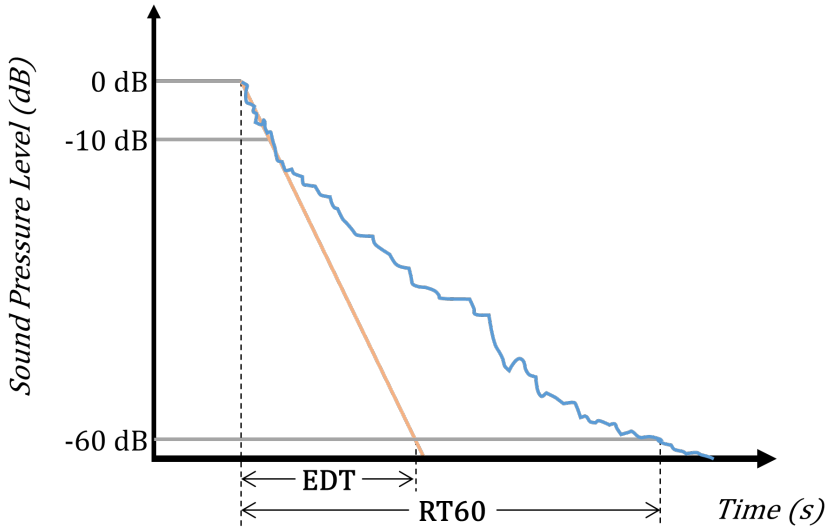


Figure 2.5: Graph showing the EDT and RT60 of a sound recording, adapted from [16].

2.6 3D Audio-Visual Scene Reproduction

With the metaverse taking centre stage in the development of novel immersive technologies, the Electronics and Telecommunications Research Institute of Korea's so-named Hyper-Reality Metaverse Research Laboratory has been engaged in numerous collaborative efforts to help achieve what they claim to be the "centre of [...] future industry" [17]. One avenue of this collaboration is between the Lab's head, Dr. Sung-Uk Jung, and our supervisor, Dr. Hansung Kim; this project was commissioned at his behest for further research in this area.

A fundamental component of the appeal of the metaverse and other such augmented or virtual reality environments is that of immersion – that is, conveying a realistic sense of presence for the user. Typically, development has been focused on greater visual fidelity; this project's focus, however, is on the audio component – convincingly real environments must expand to encompass all possible senses to achieve true immersion.

The university-provided VR headset for the duration of this project, the HP Reverb G2, lends itself well to Steam Audio (the software chosen to achieve the audio component), given that its audio solution was created in tandem with Valve Software, the developers of Steam Audio itself. Unfortunately, the Source 2 engine to which Steam Audio is best interfaced has proprietary development tools [18], and as such we opted to use Unity [19] as our development environment – the same used by our supervisor in his previous work – for which the package is available as a plugin.

Thus, the final stage of the pipeline will involve importing the outputted 3D model of the previous modules to a Unity environment with automatic assignment of acoustic properties from its recognised materials. The scene will then be rendered as a VR environment, with user functionality to experience and evaluate the audiovisual results.

3 Methodology

3.1 MDE and SSC

3.1.1 Design

The 360MonoDepth framework is a state-of-the-art solution designed to estimate depth maps from high-resolution omnidirectional images. Its methodology involves a series of advanced steps, starting with the projection of the omnidirectional image input into overlapping images that resemble the faces of an icosahedron [20]. Following this, the cutting-edge monocular depth estimation deep learning model, MiDaS, is applied to predict the depths associated with each "tangent image" on the icosahedron. Subsequently, once this is applied to all the tangent images, the individual depth maps are aligned through global optimisation[20] the use of multi-scale spatially-varying deformation fields [20]. The final step incorporates gradient-based blending to merge these aligned maps into a single detailed, coherent high-resolution depth map. This approach enables consistent depth maps, eliminating any artefacts that would otherwise be present. This approach also allows for the generation of high-resolution depth maps which are impossible to achieve with CNN-based approaches due to the high GPU memory demands.

360MonoDepth emerges as the best alternative for Monocular Depth Estimation due to its ability to be able to handle and produce high-resolution images and their depth maps effectively. The comprehensive approach used by the framework ensures that the produced depth maps are closer to the ground truth depth maps for a given image in a dataset compared to other similar networks.

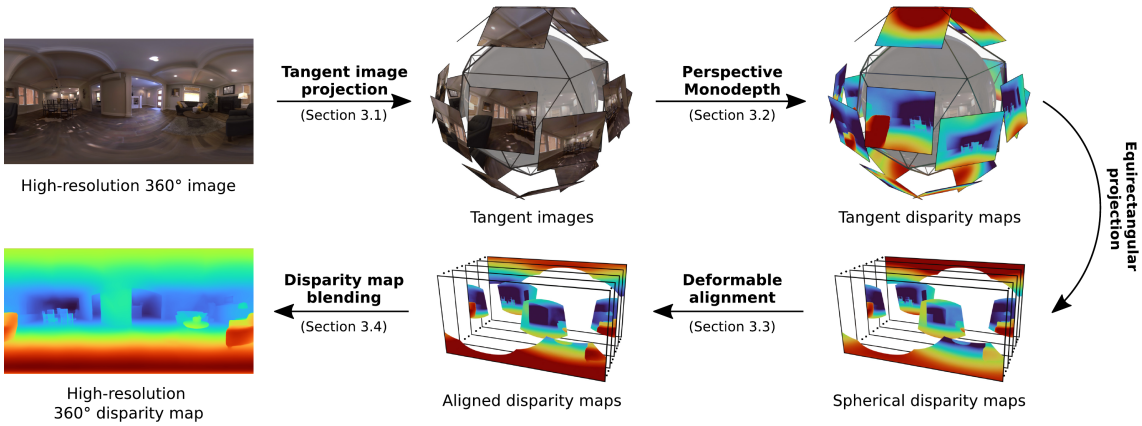
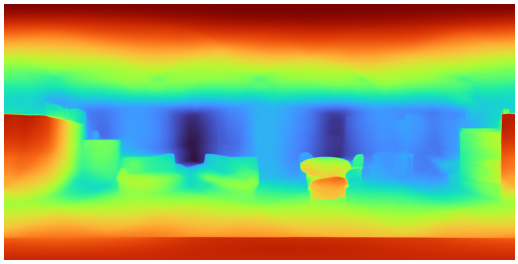


Figure 3.1: 360MonoDepth image decomposition, depth estimation, and reassembly, reproduced from [20].

3.1.2 Implementation

One of the notable challenges with using 360MonoDepth is the fact that it uses a non-linear "turbo" colour scheme to represent the final depths. Ensuring compatibility with EdgeNet360 requires a grayscale depth map represented on a linear scale. Simply converting the final rendered image is not productive as the represented greys are still on a nonlinear scale. To overcome this, it was possible to change the colour scheme of the depth map to a linear greyscale one, utilising darker shades of grey for objects closer in the scene and lighter shades for those further away. Further challenges were encountered when feeding the generated depth map into EdgeNet.

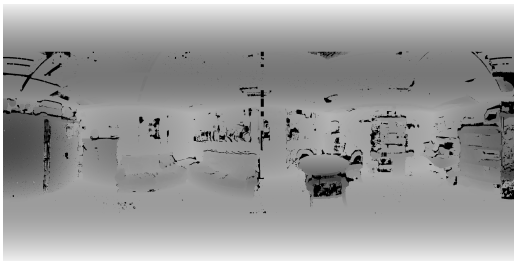
(a) Indoor 360 image reproduced from the EdgeNet test set.



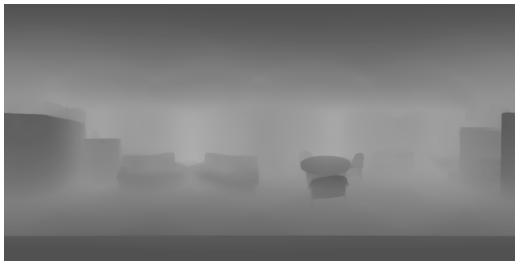
(b) Depth Map prediction generated by stock 360MonoDepth.

Figure 3.2: The RGB image input and its predicted Depth Map using unmodified 360MonoDepth.

When EdgeNet is initialised with an RGB image and its corresponding depth map, it outputs the predicted room dimensions for its reconstruction. When the depth map generated by 360MonoDepth was fed into EdgeNet, there were two main problems. The first one was that the built-in pre-processing element of EdgeNet that enhances depth maps leads to information loss in the enhanced map as it relates to object recognition; most objects in the scene from the generated depth map ceased to exist in the enhanced one. It was because of this that EdgeNet pre-processing (an optional step) was scrapped and instead, the output of 360MonoDepth is fed into EdgeNet inference straight away. Issues concerning Docker and limitations of EdgeNet itself, specifically regarding the failure of the Marching Cube algorithm on depth ranges utilising the new black and white colour scheme added another layer of complexity.



(a) The ground truth DWRC depth map reproduced from EdgeNet360.



(b) Normalised Depth Map produced with modified 360MonoDepth.

Figure 3.3: Example of the expected input and output for Depth Estimation.

In order to overcome this, normalisation of the depth ranges was performed using a meticulous approach of trial and error. The normalisation of the adapted colour scheme involves moving away from the deep blacks and the bright whites in the

colour scheme. It was possible to define a custom linear colour scheme where the colour values were constantly adjusted such that the predicted room dimensions by EdgeNet's interface were as close to those displayed when the ground truth depth map was fed in.

3.2 Material Recognition

Initially, the DBAT model was run and tested on normal 2D images to evaluate its output and verify if the model worked as expected. Figure 3.4a shows the Input images and their corresponding output images with material recognition done using DBAT. From figure 3.4a and 3.4b, it was observed that outputs from the model seemed to be accurate for images where there was sharp edge separation between different materials. It is also observed that in areas where many different materials are present together, the model seemed to overlap the material covering the larger area with the other materials. In some instances, the model incorrectly classified materials label assigning objects labels of the neighbouring materials or partial areas of some objects having some incorrectly classified material label or neighbouring material label. However, overall from figures 3.4a and 3.5 it can be seen that the model assigns material correctly in the majority of the cases and is accurate in most instances. All the input images in figures 3.4a and 3.4b are from the LMD dataset [21] which consists of low-resolution images.

(a) (b)

Figure 3.4: DBAT output with material recognition done on normal 2D images, with the input images on the left and its corresponding output image produced from DBAT on the right, reproduced from [5].

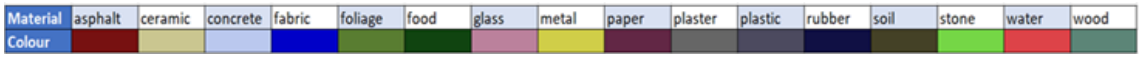


Figure 3.5: Colours assigned for the materials.

The system uses an omnidirectional (360-degree) camera to capture the images of the scenes and hence the input is a 360-degree image. However, DBAT is trained only on normal 2D images and not on omnidirectional images therefore only normal

2D images can be provided as input to the DBAT model. To bridge the gap between the input image type and DBAT's input capabilities, the technique employed was to project the omnidirectional images into a series of normal 2D images. The projected 2D images were then fed to DBAT for material recognition. Once material recognition is done on the projected 2D images they are combined to reconstruct the 360-degree image, now consisting of the material recognition data. This approach allowed to utilise the capabilities of the DBAT model without the need for retraining on omnidirectional images, also circumventing the challenge of finding an additional omnidirectional dataset with indoor scenes. The implementation for this is detailed in section 3.2.1.

3.2.1 Cube Map projection Implementation

To implement the cube map projection, `split_img.py` python script was written which takes in an omnidirectional image and splits it into 6 cube face images, and outputs them. These 6 images can be fed to DBAT for material recognition. The output material recognised cube face images are then projected and combined again using another python script `combine_img.py`.

The `split_img.py` python script implements cube-map projection to split an input 360-degree image in the following way:

`Orient_face` function: The function takes in a cube face name (e.g., 'front', 'back', etc.), an x and y coordinate on that face, and returns a 3D direction (out vector) for that point. This vector essentially points from the centre of the cube outwards.

Rendering Each Face: The `face_rendering` function then uses this `orient_face` function to convert each (x, y) pixel coordinate on a cube face to a direction in 3D space. For each direction:

- It computes the spherical coordinates (longitude and latitude).
- Maps these spherical coordinates back to the 2D coordinates of the omnidirectional image.
- Copies the colour value from the omnidirectional image to the cube face using the nearest-neighbour interpolation.

Generating All Faces: The `generate_cube_faces` function reads an omnidirectional image, sets the desired cube face size to 512x512, and calls `face_rendering` function for each of the six cube faces ("right", "left", "top", "bottom", "front", "back"). It then saves each rendered face as an individual image. The cube map for the output images can be visualised as shown in figure 3.6.

The `combine_img.py` python script combined 2D cube map images to create a 360-degree image. The working of the file can be broken down as:

- The 6 face images are loaded: the top and bottom face images are rotated by 90 degrees and the top, bottom, front, and back face images are flipped in the horizontal direction to ensure their orientation will match the expected output omnidirectional image layout. Then these processed cube faces are passed to the `cubemap_to_omnidirectional` function.

- `Cubemap_to_omnidirectional` function: This does the main conversion from the cube faces to the omnidirectional image. It iterates over each pixel in the output omnidirectional image, converts the pixel's coordinates to 3D spherical coordinates, determines which cube face and position this point corresponds to, and uses the nearest neighbour interpolation to get pixel value from the cube map image and assigns it to actual pixel in the omnidirectional image.
- `Orientation_to_face` function: finds out which cube face corresponds to the current pixel for which the 3d coordinates are calculated and calculates the normalized 2d coordinates on that face.

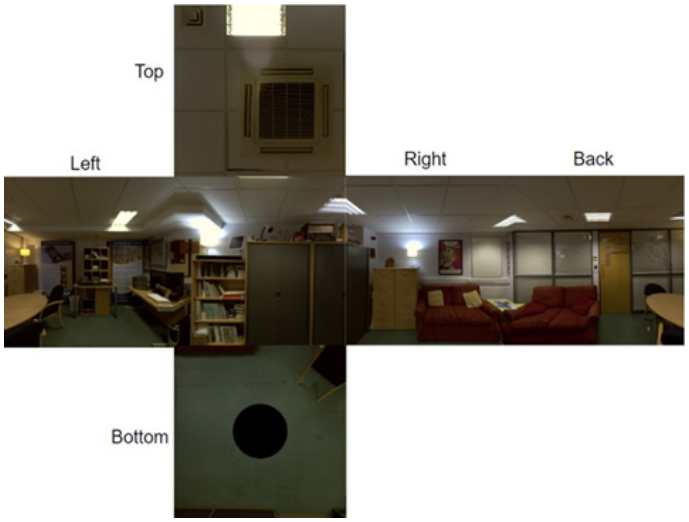


Figure 3.6: Cube map created from the output projected 2D images after cube map projection.

3.3 Automatic Material Mapping

Currently, the EdgeNet360 model uses the 3D-to-2D projection to reproduce the 3D scene based on the depth map. Therefore, the design for material mapping exploited this functionality to implement the projection of materials onto the scene. Figure 3.7 provides the changes made to the EdgeNet360 pipeline. The red arrows show the original EdgeNet360 pipeline. The first two modules were altered to include volumetric encoding of the material labels. The blue arrows indicate the new modules implemented for the assignment of material to each object in the scene. Therefore, the implementation has the following features:

- The volumetric encoding of the materials, and the material assignment to objects.
- Splitting the objects into separate classes, such that each object has its label.

3.3.1 Merging Material Recognition to the Pipeline

Material recognition was initially linked with the rest of the pipeline by loading the material labeled map into EdgeNet360, and its corresponding colour plate, which was a lookup table defining the RGB values corresponding to the material label. The DBAT model defined 2 dictionaries: a `color_plate` dictionary in which each

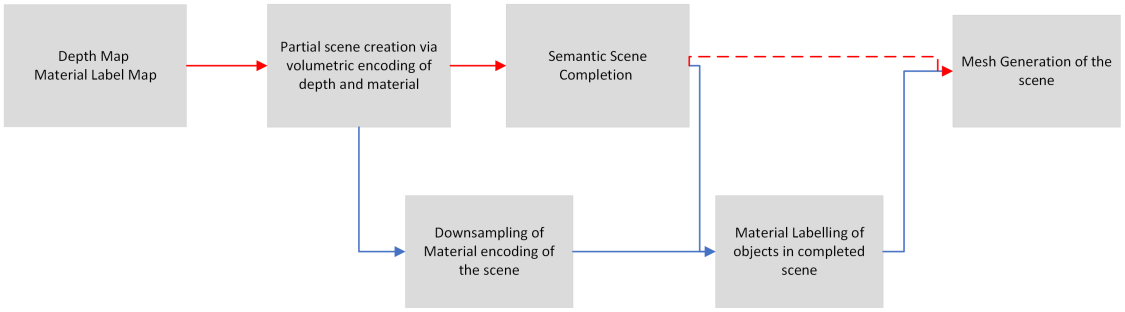


Figure 3.7: Pipeline for material mapping using EdgeNet360.

RGB colour value corresponded to a key from 0 to 15, and another dictionary which mapped these keys to each of the 16 different material labels, as seen in 3.8.

The RGB value keys corresponded to the material types and their keys. However, when the material map was loaded in EdgeNet360, a `grayscale_plate` was defined which consisted of the converted grayscale values for each RGB value from the `color_plate`. The grayscale values corresponded to the same keys and hence the same material for their RGB counterparts. This conversion was done, such that the space complexity of the mapping operation would be reduced.

```
material_labels = {
    0: "asphalt", 1: "ceramic", 2: "concrete", 3: "fabric", 4: "foliage",
    5: "food", 6: "glass", 7: "metal", 8: "paper", 9: "plaster", 10: "plastic",
    11: "rubber", 12: "soil", 13: "stone", 14: "water", 15: "wood"
}

color_plate = {
    0: [119, 17, 17], 1: [202, 198, 144], 2: [186, 200, 238], 3: [0, 0, 200], 4: [89, 125, 49],
    5: [16, 68, 16], 6: [187, 129, 156], 7: [208, 206, 72], 8: [98, 39, 69], 9: [102, 102, 102],
    10: [76, 74, 95], 11: [16, 16, 68], 12: [68, 65, 38], 13: [117, 214, 70], 14: [221, 67, 72],
    15: [92, 133, 119]
}

grayscale_plate = {0: 47, 1: 193, 2: 200, 3: 23, 4: 106, 5: 41, 6: 149, 7: 191, 8: 60, 9: 102,
                   10: 77, 11: 22, 12: 63, 13: 169, 14: 114, 15: 119}
```

Figure 3.8: Diagram showing the different dictionaries imported to EdgeNet-360 from the DBAT model.

3.3.2 Volumetric Encoding of Materials

Originally, the EdgeNet360 pipeline created an empty voxel grid and loaded both the grid and depth image into the flipped-TSDF function. This method was altered to integrate material mapping. In essence, another empty voxel grid with the same dimensions (240x144x240) as the voxel grid for semantic scene completion (SSC) was created and initialised. This empty voxel grid was passed into flipped-TSDF, alongside the material image. The function utilised the voxel coordinates to measure relative world coordinates, from which using geometry, the corresponding pixel coordinates were calculated. Since the depth map and the material label map have the same dimensions and pixel resolution, this mapping could be used to retrieve the grayscale value of that pixel, to assign to the corresponding voxel. The material value was only assigned if the voxel represented a surface, i.e., voxels that represented non-visible parts of the scene in the image were not assigned any labels. This

was done to avoid wrongly assigning values to occluded regions. Assigning incorrect material to occluded regions would have a cascading impact on the accuracy of material assignment on objects.

During semantic scene completion, the original incomplete voxel scene is inputted into a convolutional neural network (CNN), which outputs the complete scene. The completed scene has dimensions 0.25x of the input scene. Therefore, to be compatible with the completed scene, the material-encoded voxel grid had to be downsampled by the same ratio, where 4x4x4 of the original voxels would be mapped to 1x1x1 of the new voxel grid. Mode-down sampling was the chosen technique since the set of material values was discrete, and to avoid random assignment to voxels. Mode-down sampling worked by measuring the most common material value between those 64 adjacent voxels and assigning that value to the new voxel, fundamentally implementing a majority voting mechanism. If all the original voxels or the majority of those 64 voxels had no material values assigned to them, the new voxel would be assigned 0.

Object Material Assignment

EdgeNet360 itself employed object detection to identify objects in the scene. Using the depth estimated input image and the original RGB image of the scene. Each voxel was assigned an object label such as a sofa, furniture, floor, wall, object, etc.

Majority Voting Mechanism

The output of the CNN (depth-based voxel grid with objects identified) and the material-encoded voxel grid (voxel grid created from the material map) were then utilised to assign a material to each object in the scene, via majority voting. For each object, all the voxels making up that object were identified, and their coordinates were used to retrieve all the material values present. Using these voxel coordinates, the material values stored in the material encoded voxel grid were retrieved, and the frequency of each possible material was calculated per object. The most frequent material was assigned to that object. This data was saved in a HashMap, to be used for testing and validation.

After creating a voxel grid with the appropriate material labels assigned to each voxel the next step was to generate a mesh from this grid. A mesh would have a smoother surface and texture which would better present the scene visually than a voxel grid consisting of cube-shaped elements. However, the mesh generated was deemed incompatible with the assignment of material properties on Unity due to the fact it was a single mesh and there were no object labels present in the scene. Therefore, assigning material properties using object names was not possible on Unity.

The solution derived for this problem compromised of utilising the object-material HashMap to segregate the mesh based on materials assigned to the objects. The idea was to iterate through the mesh and assign different colours to it based on the materials for those objects. Since the mesh was generated by applying the marching cubes algorithm on the voxel grid, retrieving the object label for each generated vertex and face was possible. Therefore, the face and vertices were assigned colours

based on the material assigned to them. The RGB colour value was assigned based on the colour plate defined in DBAT. After the mesh file was generated, the mesh was split based on the colours present on the scene, providing material-based sub-meshes.

Object Splitting

EdgeNet-360 returns a set of objects under the same label. For example, multiple chairs within the scene are classified under the same 'chair' label. However, since different chairs in the scene may be composed of different materials, these chairs need to be separated under different labels. Otherwise, the material assignment would assign all chairs the most prevalent material.

Therefore, to further improve upon the resolution of the acoustic properties of the scene, the object labelling was enhanced such that each separated object had a label. This would improve the acoustic properties as the material labelling of each object would be closer aligned to what the DBAT model outputted. The majority voting would be done on a single object rather than the group and hence would be more precise to that object.

Separating object labels entailed checking for connectivity of the voxel blocks. Voxel blocks that were completely disconnected, with empty space between the blocks, would be split into different object labels. Therefore, as seen in figure 3.9, 26-connected was preferred as it matched with what was required.

The implementation of splitting based on connectivity was done via `skimage.measure.label` [22], which returned a voxel array, containing the different labels based on connectivity.

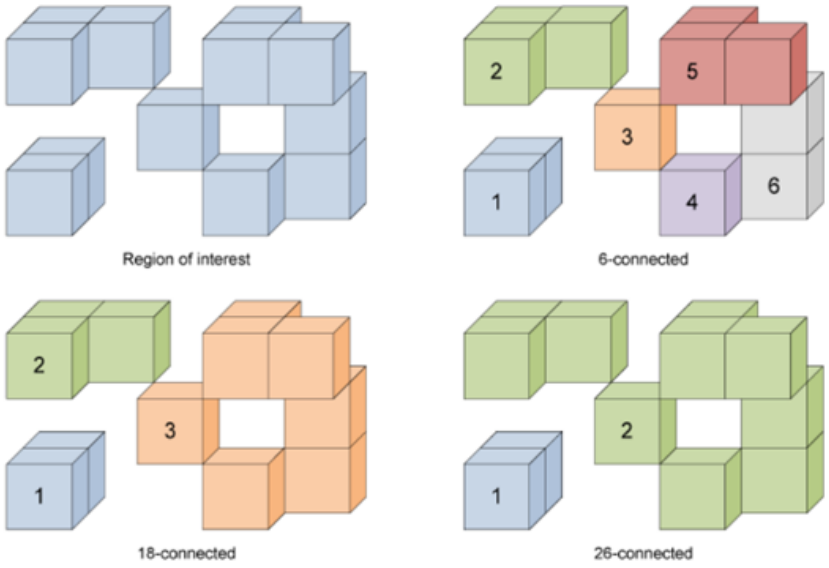
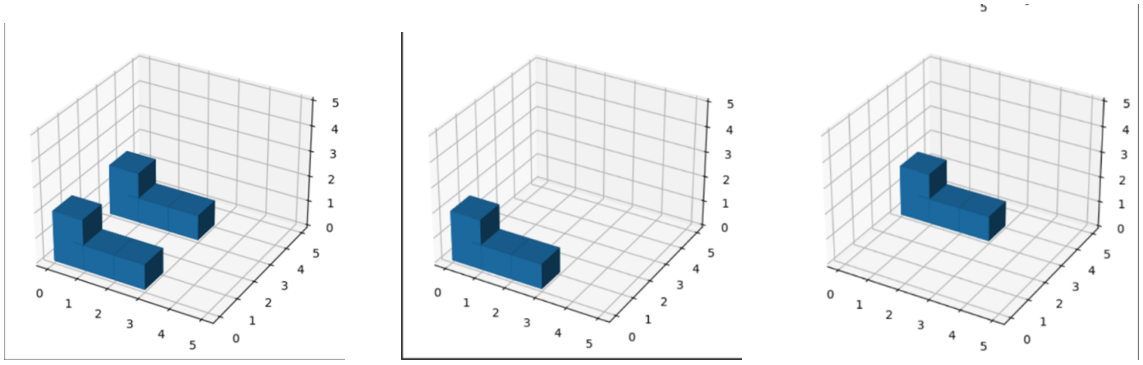


Figure 3.9: Diagram showing the types of voxels of connectivity, reproduced from [23].

Before applying it for EdgeNet 360, the functionality was tested on a simulated voxel grid, containing two completely separate voxel blocks, classified under a single object label, as seen in figure 3.10a. The function applied provided 3 unique labels

to the voxel scene, which corresponded to [0: corresponding to empty space, 1: corresponding to the closest voxel blocks, 2: corresponding to the furthest voxel blocks]. Hence, it was possible to retrieve the voxels based on the specific label, as seen in Figures 3.10b and 3.10c. This proof of concept provided strong grounds to implement the function into EdgeNet.



(a) The initial testing grid which contains 2 objects grouped under the same "object" label.

(b) The separated object block after splitting under a new object label.

(c) The object furthest under another object label after splitting.

Figure 3.10: Testing grids.

The implementation on EdgeNet followed the steps:

1. Iterate through object groups: For each existing object class, the voxels making up the specified object were copied into an empty voxel grid which was then passed into the function. This outputted the array contained label for each voxel. This was used to identify the number of separate objects within the group.
2. Create object labels: Upon calculating the number of separate objects, new labels were created for the new classes. E.g. for 2 separate sofas in the scene, the classes sofa0 and sofa1 were created.
3. Update voxel grid with new labels: The corresponding voxels in the original voxel grid were updated with the new object classes.

3.4 Acoustic Room Modelling

3.4.1 Steam Audio

Acoustic Modelling for the generated scenery was accomplished with the Steam Audio [13] solution provided by Valve Software, pioneers in the Virtual Reality space; it was determined that this would pair well with the provided HP Reverb G2 headset, given that its spatial speaker hardware was developed in collaboration between HP and Valve [24].

Steam Audio is a package with plugins developed for common game engines such as Unreal and Unity (the chosen engine for this project) that abstracts complex calculations for producing realistic spatial audio within 3D environments. Given the presence of 3D scenery, Steam Audio provides numerous modules that were

utilised in modelling the acoustics for our generated scenes, as referenced in their documentation [25].

Steam Audio Source

This allows for the spatialisation and reverb of the reference audio emission to be customised through the provided parameters of the module: the chosen settings, derived through experimentation as well as quantitative and qualitative evaluation (refer to Results section) were:

- **HRTF (Head-Related Transfer Function):** Using binaural audio with bilinear interpolation (from four directions chosen close to the audio source) enabled the spatialisation of the audio source to change based on its position relative to the listener; perspective correction allowed for the listener's position (i.e. the camera's projection) to affect the perceived spatial direction of the audio source during locomotion throughout the scene.

- **Attenuation:** Using a curve-driven distance attenuation caused the amplitude of the sound signal to lower with the listener's distance from the sound source, with air absorption defined preset simulation parameters for added realism with this effect.

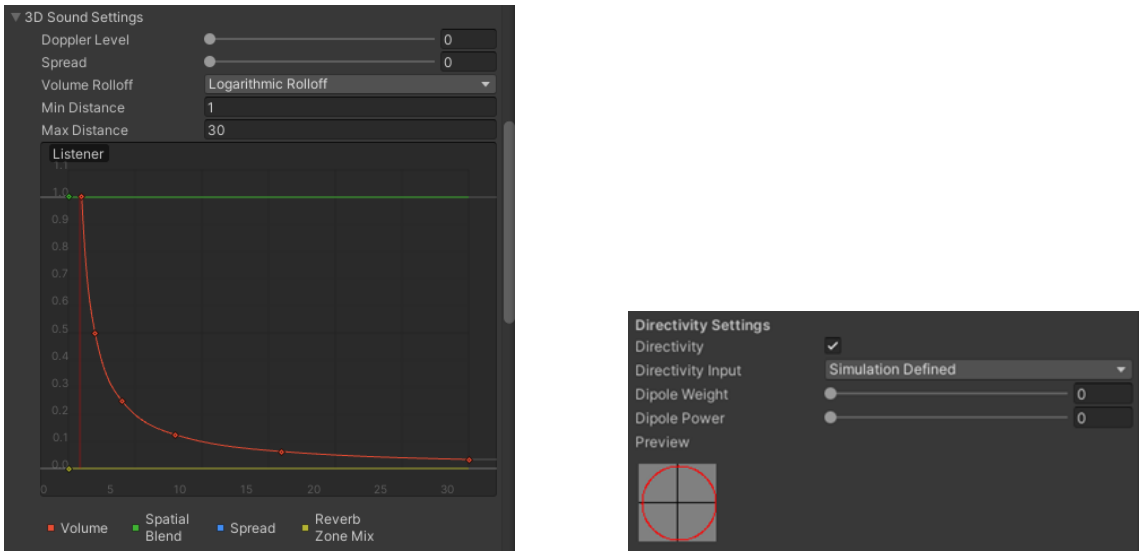
  With "curve-driven" selected, this setting is thus derived from the native Unity Audio Source module, in which the volume roll-off is set to Logarithmic to imitate a natural reduction in direct volume with distance (figure 3.11a).

- **Directivity:** This allows for ensuring constant, equal sound directivity in all directions (figure 3.11b).

- **Occlusion:** Crucially, this automates the ray casting of the sound source to the listener for diminishing the sound source amplitude when occluded by any intermediate geometry. Whilst the volumetric setting would provide a more accurate sense of gradual occlusion, its performance impact was deemed too high for relatively minor qualitative gains (particularly with acoustic modelling focusing on the sound's reverberation and reflection, rather than its direct source).

- **Reflections:** The most important setting for representing the acoustics of the room, enabling reverberations from all scene geometry per their material properties. This was set to real-time to accommodate dynamic changes to the scene (in terms of listener and source position) – the attenuation's distance curve shown under Attenuation is used in reflection calculations to not dominate the sound from the original source, and HRTF is also applied to the reflections for spatialisation per listener position and orientation.

  The mix of the reflections is also reduced to 0.3; this does not affect reflection accuracy, and merely reduces their overall volume to allow for the direct sound source to be heard more acutely (a value arrived through qualitative evaluation, as again this will not affect quantitative results such as the scenery's Room Impulse Responses).



(a) Unity Volume Rolloff settings.

(b) Unity Directivity settings.

Figure 3.11: Unity Steam Audio source settings.

Steam Audio Listener

Defining the object to which the Steam Audio output is sent, this was set to the Main Camera that renders the Unity Scene, with its constituent Reverb parameter set to Realtime to route the real-time reverberations calculated from the Audio Source.

For acoustic testing, a simple script was added to the listener object (Main Camera) to enable keyboard and mouse movement throughout the scene (`KeyboardMove.cs`); this is disabled in the final product, as the scene must be experienced in VR.

Steam Audio Geometry

This feature of Steam Audio is paramount to the generation of an acoustic model for the pipeline output scene: this allows for a static mesh to be produced from all tagged objects in the scene to which a Steam Audio Geometry component is attached; all spatial and reverberation calculations use this mesh for the completion of acoustic room modelling at render time (note that by design, this must be manually exported in Unity's Edit mode before running the scene – a challenge to be overcome in automating our pipeline for minimal user input).

Steam Audio Settings

The global audio output settings can be defined here, in which HRTF was globally applied with perspective correction. GPU-accelerated rendering was enabled by setting scene raycasting to Radeon Rays, and the reverberation effect to the TrueAudio Next preset. With the GPU selected as the device, the maximum Maximum Reserved Compute Units (16), and maximum Fraction Compute Units (1) parameters were also set. Realtime reflection settings, a vital component of the realism of scene acoustics, were increased to the maximum experimentally determined to be achievable without negatively impacting runtime performance.

Each sub-mesh of the pipeline output, split by material as output by the Material

Setting	Value
Real-Time Rays	16384
Real-Time Bounces	64
Real-Time Duration	10
Real-Time Ambisonic Order	3
Real-Time Max Sources	64
Real-Time CPU Cores Percentage	20
Real-Time Irradiance Min Distance	1

Table 3.1: Table representing the final Steam Audio global settings.

Recognition module, is assigned a Steam Audio Geometry component to which a Steam Audio Material is applied. The acoustic parameters for each material are defined in table 3.2.

	Absorption			**Scattering**	**Transmission**		
Material	Low Freq.	Mid Freq.	High Freq.		Low Freq.	Mid Freq.	High Freq.
[**Default**]	0.1	0.1	0.1	0.5	0.1	0.1	0.1
Asphalt	0.6	0.7	0.8	0.05	0.031	0.012	0.008
Brick	0.03	0.04	0.07	0.05	0.015	0.015	0.015
Carpet	0.24	0.69	0.73	0.05	0.02	0.005	0.003
Ceramic	0.01	0.01	0.02	0.05	0.06	0.044	0.011
Concrete	0.05	0.07	0.08	0.05	0.015	0.002	0.001
Fabric	0.24	0.69	0.73	0.05	0.02	0.005	0.003
Foliage	0.3	0.5	0.9	0.05	0.1	0.1	0.1
Food	0.8	0.5	0.8	0.05	0.1	0.1	0.1
Glass	0.06	0.03	0.02	0.05	0.06	0.044	0.011
Gravel	0.6	0.7	0.8	0.05	0.031	0.012	0.008
Metal	0.2	0.07	0.06	0.05	0.2	0.025	0.01
Paper	0.10	0.07	0.06	0.05	0.056	0.056	0.004
Plaster	0.12	0.06	0.04	0.05	0.056	0.056	0.004
Plastic	0.07	0.14	0.14	0.05	0.056	0.056	0.004
Rock	0.13	0.2	0.24	0.05	0.015	0.002	0.001
Rubber	0.05	0.1	0.05	0.05	0.1	0.1	0.1
Soil	0.5	0.8	0.5	0.05	0.1	0.1	0.1
Stone	0.02	0.02	0.05	0.05	0.1	0.1	0.1
Water	0.01	0.01	0.02	0.05	0.1	0.1	0.1
Wood	0.11	0.07	0.06	0.05	0.07	0.014	0.005

Table 3.2: Steam Materials and their properties: sample materials in bold, custom otherwise.

Using publicly available data, it was determined that the absorption coefficients provided in the sample materials (bolded in the above table) correspond to around 250-500 Hz for low, 1000-2000 Hz for medium, and 2000-4000 Hz for high. Therefore, for any custom materials we defined, we referred to these ranges from numerous tabular sources to determine their parameters [26][27][28][29]. Transmission values were either kept as default or copied from analogous materials, as our acoustic modelling does not have transmission enabled. The scattering value of 0.05 present in all predefined materials was also kept constant for all subsequently defined materials.

3.4.2 Scene Setup

To create a scene for acoustic modelling, the Steam Audio components were manually applied as listed previously to a Unity scene with an EdgeNet voxel output mesh for which the scene hierarchy was as follows:

- Main Camera
  - [Unity] Audio Listener
  - Steam Audio Listener
    - Settings and parameters as described in the previous Steam Audio section.
  - KeyboardMove script
- Scene Mesh "DWRC_prediction"
  - [Unity] Mesh: Sub-mesh 1...n
    - Steam Audio Geometry is applied with Audio Material as outlined in the above table.
- Audio Source
  - [Unity] Sphere (Mesh)
  - [Unity] Audio Source
    - Spatialise, with Steam Audio overriding default Unity audio rendering pipeline.
  - Steam Audio Source
    - Settings and parameters as described in the previous Steam Audio section.
- [Steam Audio Static Mesh and Steam Audio Manager]
  - Dynamically generated by Steam Audio at runtime to handle audio pipeline.

This type of scene is an intermediary, as it contains no Virtual Reality or automation components, and was used to finalise the audio settings used when implementing these in the next section.

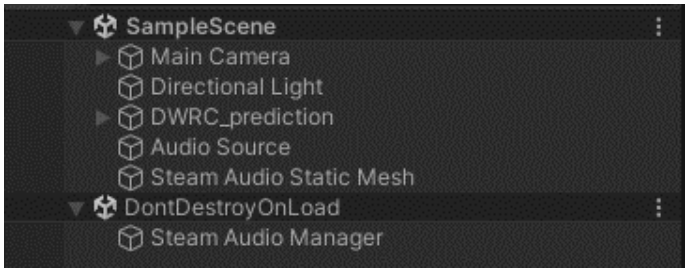


Figure 3.12: Screenshot of the implementation of scene hierarchy.

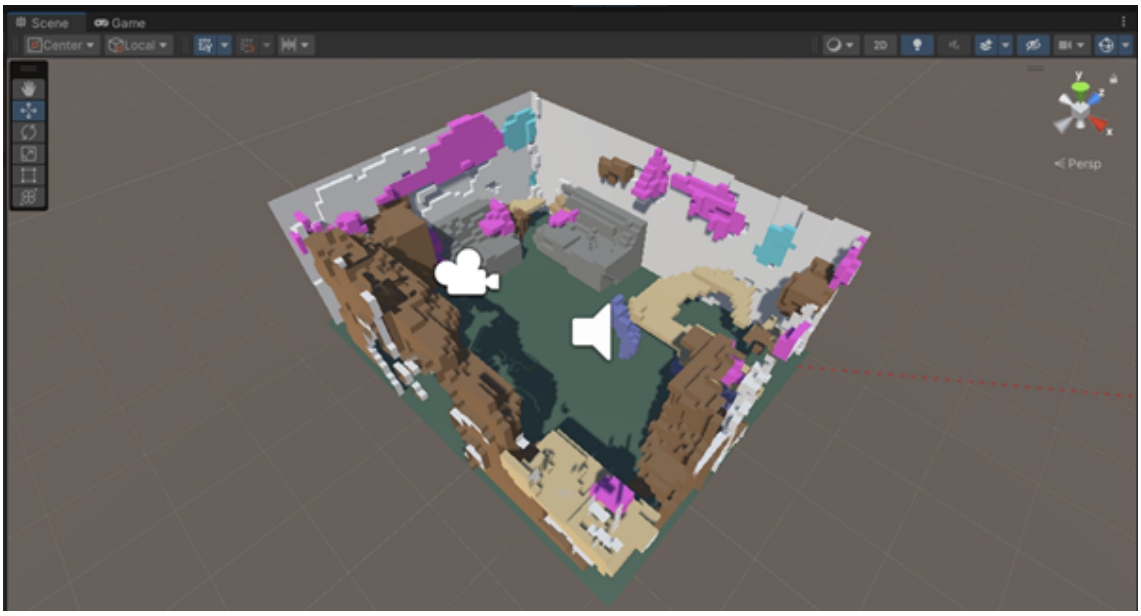


Figure 3.13: Screenshot of the Scene View, showing Main Camera, prediction mesh, and Audio Source in Editor.

3.5 3D Audio-Visual Scene Reproduction

3.5.1 Virtual Reality Scene

For the hardware aspect of the project, the first stage was the evaluation of the compatibility requirements for the university-provided VR setup. Whilst the laptop did not possess the required DisplayPort connection for the headset's display cable, it was determined this connectivity issue could be resolved by requisitioning an adapter [30] to its USBC port (which supported the required data throughput) and using the headset's included USBC-USBA adapter for the data cable.

With the setup complete, development proceeded with the OpenXR API [31] for Virtual Reality development, chosen as it is a popular open-source and platform-agnostic (allowing for development on multiple differing headsets) environment for developing the VR scenery within Unity.

Using OpenXR, the VR environment is set up with the platform-defined XR Origin object, encapsulating the scene's Main Camera (corresponding to the headset's rendered output), and the headset controllers. An Interaction Manager object is defined within the XR Origin, serving as the Input Action Manager through which all controller inputs are routed to enable interactivity within the scene. Finally, a Locomotion System object references a Locomotion Area: a simple plane mapped to the scene's mesh floor upon which the user can move. The scene hierarchy is thus shown in table A.1 in Appendix A.

3.5.2 Controllers

For each controller, all input actions were mapped to OpenXR's platform-agnostic input actions, allowing for tracking the controllers within the scene and for each button to be mapped to a different function.

Teleportation

Using a prefab model as the source, the first action a user can accomplish is via the teleportation locomotion system. A projectile ray interactor, terminating in a spherical reticule on the ground, allows for the selection of a location on the Locomotion Area plane object (figure 3.14).

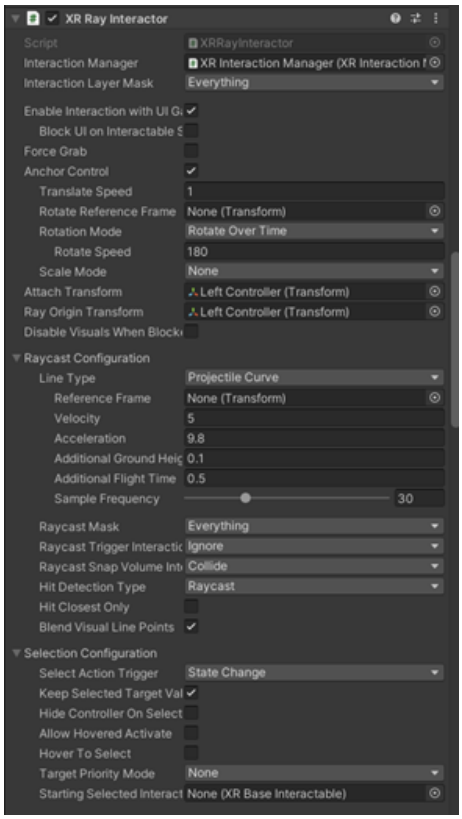


Figure 3.14: Ray Interactor Settings.

With OpenXR's Input Actions dialog, this was remapped to the controller trigger button for activation (figure 3.15).

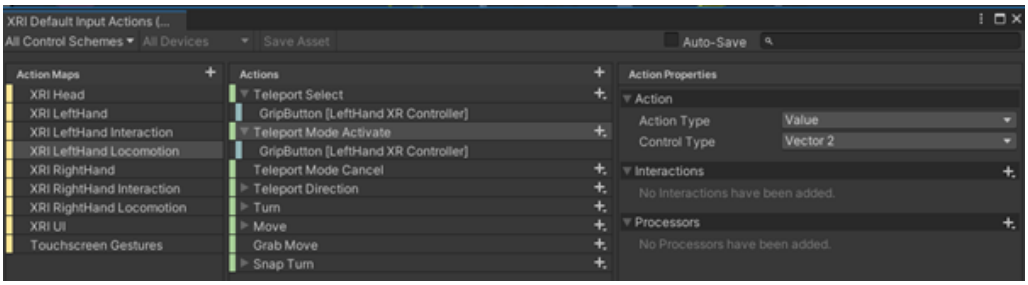


Figure 3.15: XRI Default Input Action mapping

A simple script, RayHider, as well as the following Interactor Events, allowed for the Ray Interactor to be shown only when selecting a teleport location (figure 3.16).

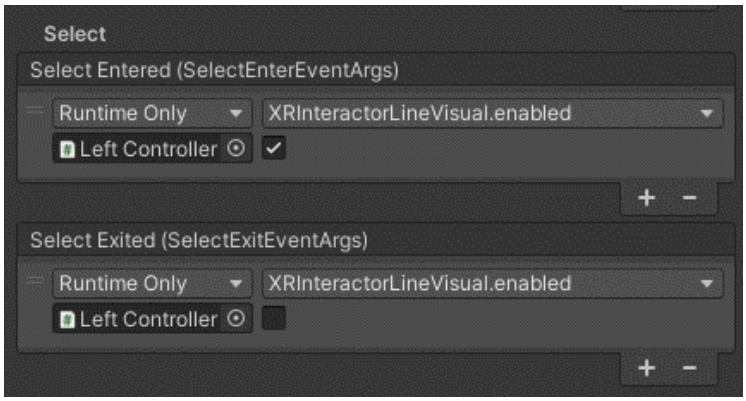


Figure 3.16: Controller Interactor Events.

Smooth Locomotion

With OpenXR providing default scripting from smooth locomotion, enabling this feature was as simple as assigning input references (in this case, from the Left Controller's analogue stick) to these scripts, and routing the action through the Locomotion System object.

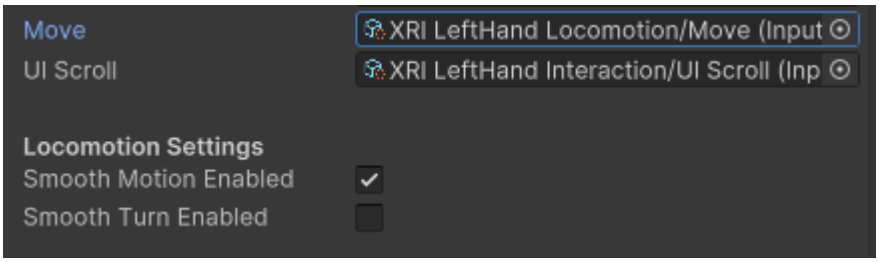


Figure 3.17: Smooth Locomotion settings.

Snap Turning (Locomotion)

Similarly to smooth locomotion, this function required only setting the related parameters to our preference, and mapping the correct controller input action (Right Controller's analogue stick) to activate.

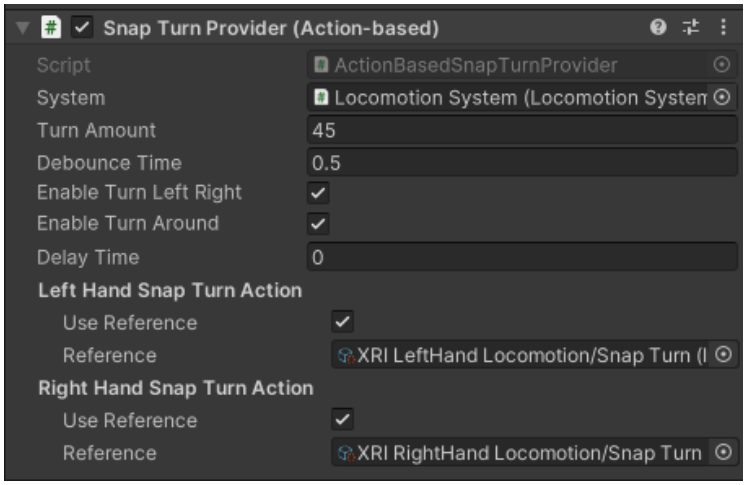


Figure 3.18: Snap Turn settings.

3.5.3 Interactables

To allow for additional functionality, the XR Grab Interactable component was added to the object containing the audio source for the scene. With Unity Collider components added to both controllers as well as the object, the interaction became possible upon the Collider intersection between a controller and the object (with the input action activated) – allowing the sphere from which the audio is produced to be moved throughout the scene.

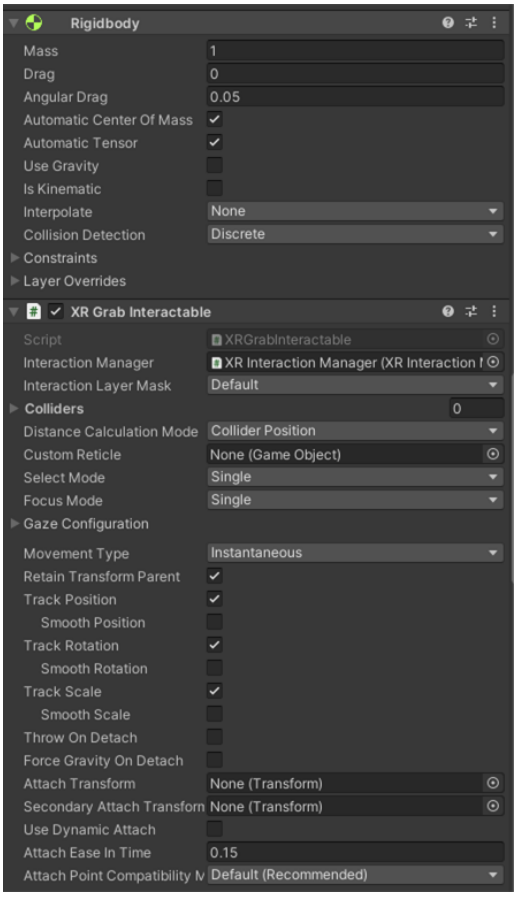


Figure 3.19: XR Grab Interactable settings, with Rigidbody component allowing for movement of objects.

For the user experience, the material (i.e. colour) of the sphere changes whilst active using an Interactor Event:

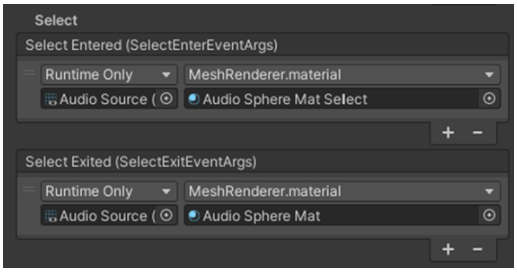


Figure 3.20: Select Interaction runtime events

The controls for all controller-mapped functionality can be found in the User Manual, contained in Appendix D.1.

3.6 Automation and Mesh Importing

For the Unity scene to work with the machine learning pipeline, the generated mesh for the scene needs to be imported, and the scene must dynamically change to generate its particular Acoustic Modelling and scene characteristics. There were several strategies used to accomplish this.

3.6.1 Relocation of pre-existing objects

All objects corresponding to the user position were relocated to coordinates (x: 0, y: 0, z: 0) within the scene (the XR Origin and all its children), with the audio source placed at (x: 0, y: 1, z: 0) . This allowed for compatibility with the mesh of the scene import, which is generated with the origin of the camera producing the source omnidirectional image also corresponding to (x: 0,y: 0,z: 0) within its mesh.

3.6.2 Mesh editing before import

Due to a peculiarity of our pipeline's output, the mesh's normals are inverted (meaning a simple import of the .obj file would result in a visually incomplete scene due to Unity's native back-culling rendering). Thus, an additional script to generate the final mesh had to be appended to the ML pipeline before the import into Unity, which flips the normal of all meshes within. This was accomplished with the bpy Blender Python module.

Additionally, for compatibility with the previous step's origin at (x: 0,y: 0,z: 0) strategy, the mesh also had to be relocated within this script, as shown in `blenderflip.py` A.2 in Appendix A.

3.6.3 Importing the mesh

Due to `.obj` importing being deprecated in modern versions of Unity, a freely available plugin was used to accomplish the mesh import [32]. A menu item was added to select the `.obj` file, which is then imported into a wrapper object (allowing for replacement upon successive imports).

The imported model is then given the `AssignMaterials.cs` script, which traverses its sub-meshes and adds the Steam Audio Geometry component to each, assigning the correct Steam Audio Material as per the mesh's colours as they correspond to the material names (refer to the Material Recognition section). With this complete, the Steam Audio Manager's Export Scene function is automatically called to generate a Static Mesh so that the Acoustic Modelling is complete and ready to be simulated in Play mode.

Finally, a deceptively difficult task was automating the resizing of the Teleport Locomotion Area. This was eventually accomplished by changing the type of mesh from a Plane to a Quad (which is defined by only four vertices): through calculating the bounds of the imported mesh, the Quad is updated based on recalculated vertices derived from these bounds in the `ResizeLocomotionArea.cs` script.

3.6.4 User functionality

As customisation of Steam Audio properties and geometry must be performed in Edit mode, the audiovisual rendering part of the project could not be built into a standalone application without the Unity Editor. Thus, the ability for the user to customise and interact with the system is provided in via a custom ribbon menu named "AVVR", with items corresponding to scripts. Each of the menu items

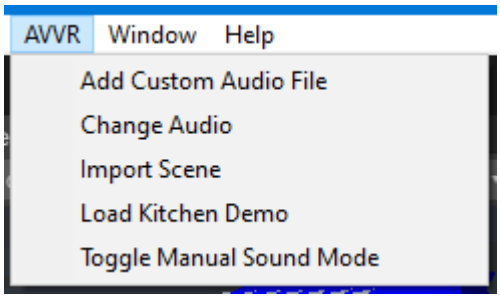


Figure 3.21: The AVVR custom menu ribbon.

performs a specific function:

- **Import Scene:** File selection for importing the mesh, as described in the previous sub-section of the same name.
- **Toggle Manual Sound Mode:** A toggle to swap the sound's behaviour between two modes, executed by the `SoundClickToggle.cs` script:
  - Manual play-on button press (mapped to a controller input action, and executed by the `PlayOnClick.cs` script).
  - Automatically looping from the runtime start.
- **Change Audio:** executed by the `ChangeAudio.cs` script, this opens a pop-up of the Audio Source Component property window, from which the source audio can be selected in the top AudioClip field, which opens a menu containing all files in the project's `Resources/Audio` folder.
- **Add Custom Audio File:** executed by the `ImportSound.cs` script, this opens a file selector to import any custom audio file to the `Resources/Audio` folder, allowing it to be assigned with the above Change Audio item.
- **Load Kitchen Demo:** runs the `LoadDemo.cs` script, which loads the demo scene described as follows:

3.6.5 Demo scene

For an enhanced user experience with increased visual fidelity, the `LoadDemo.cs` script, when called from its corresponding menu item, loads a saved .obj file of our pipeline output (held in `Resources/Demo/DemoModel` in the Unity project's filesystem). The materials of this model are set to set to translucent so that a manually scaled LiDAR scan (provided by our supervisor) can be overlaid. This demo is of the same Kitchen scene used to obtain the Acoustic Modelling results in the Results section, and was shown live at the poster presentation of our project on the 26th of January.

3.6.6 Screenshots

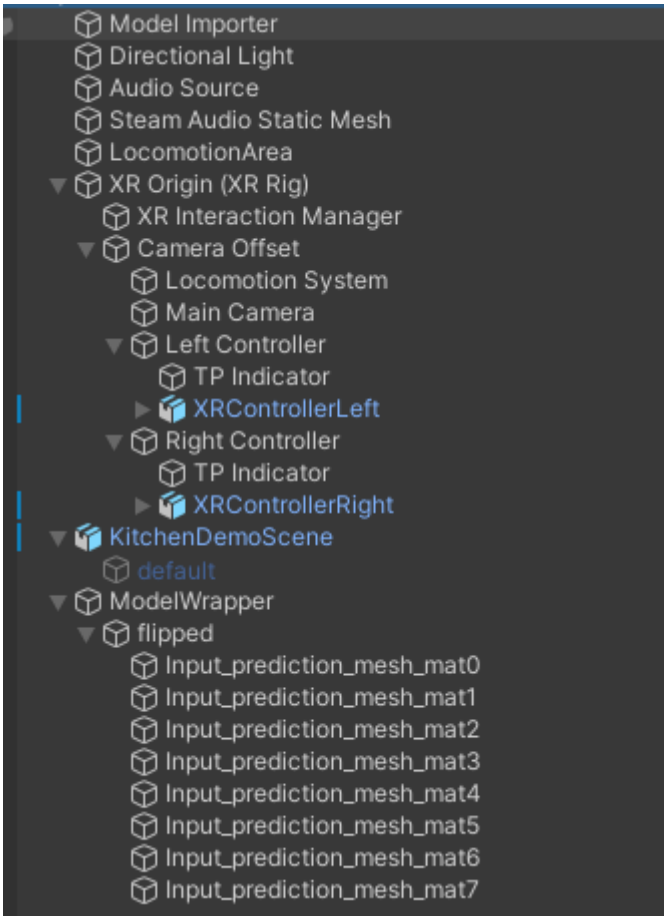


Figure 3.22: Full Hierarchy of an imported scene.

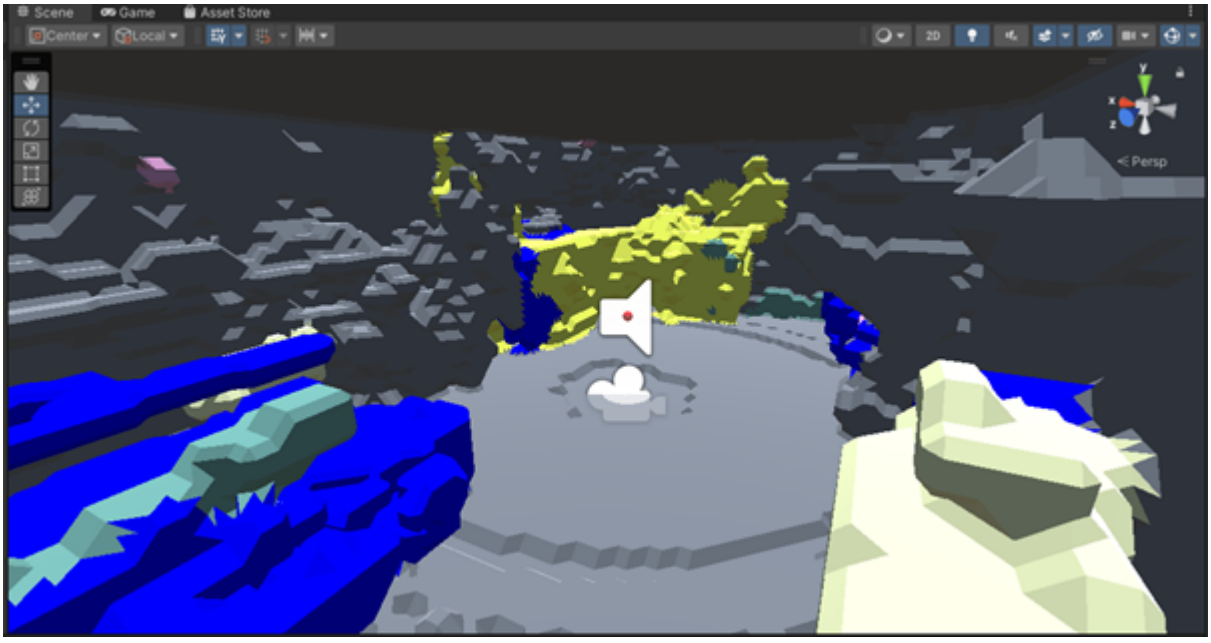


Figure 3.23: Edit view of an imported scene.

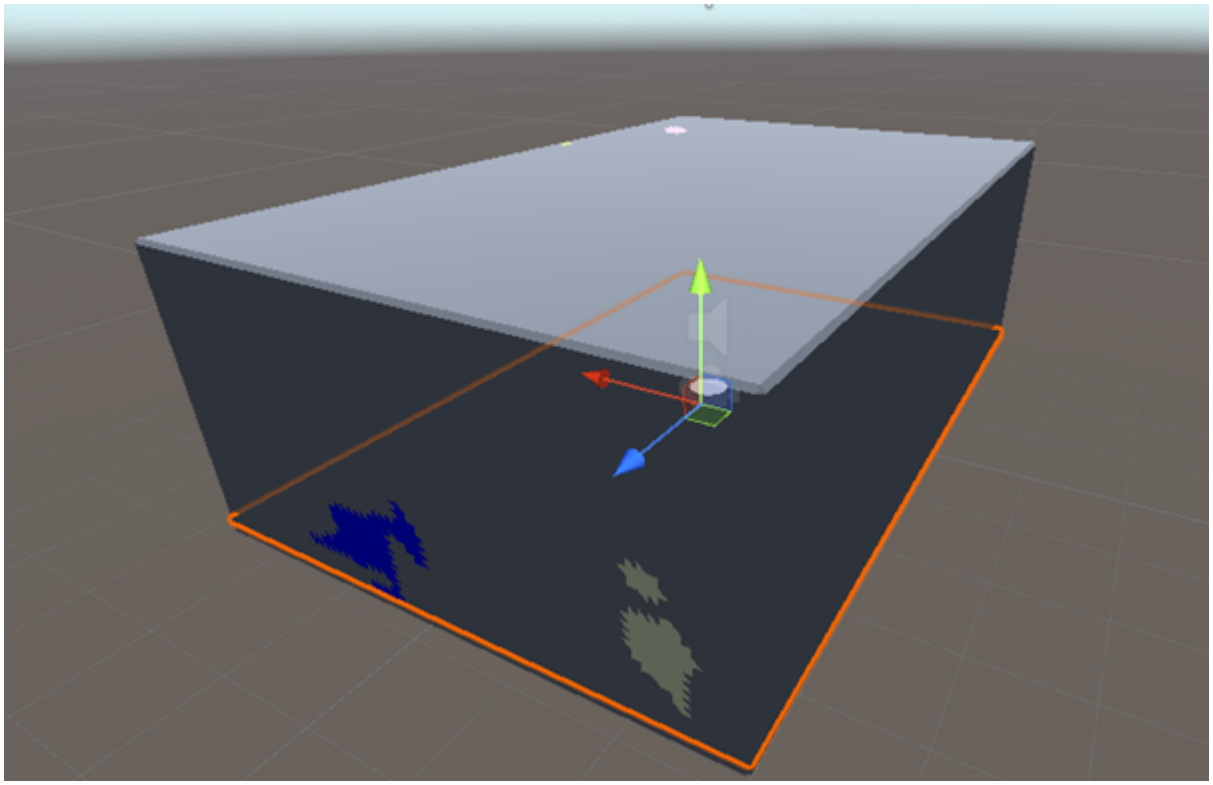


Figure 3.24: Selected Teleport Locomotion Area to show dynamically resized plane.

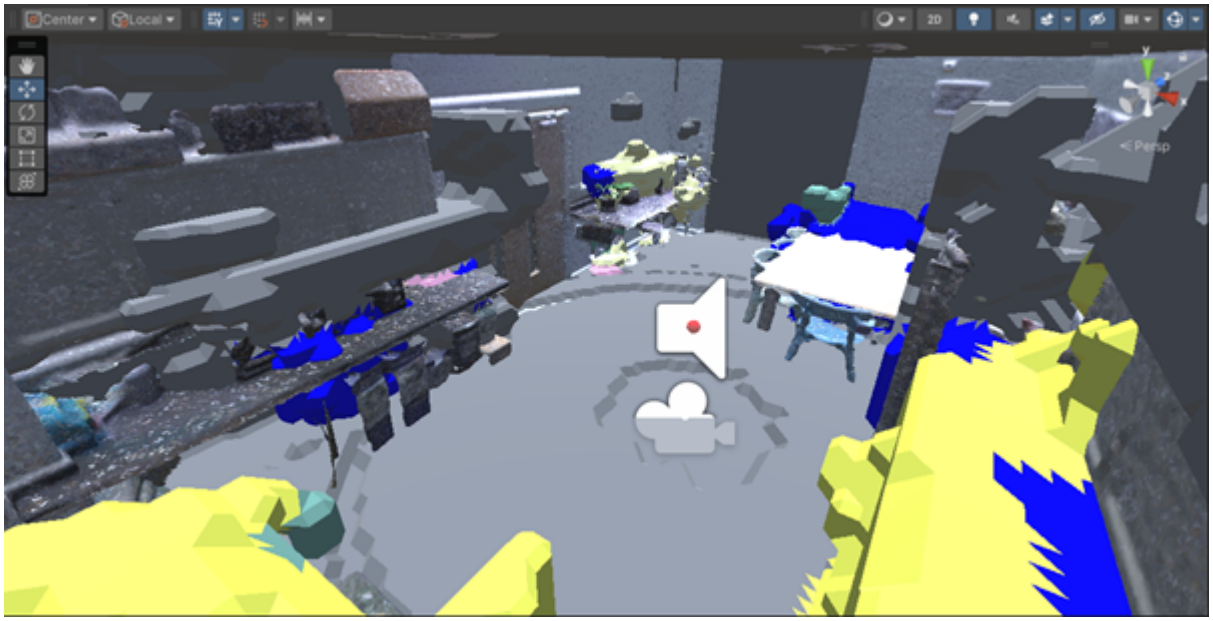


Figure 3.25: The Kitchen demo scene in Edit Mode.

3.7 Scripting for Pipeline Automation

One of the main aims of the project was automation and seamless integration of the 3D reconstruction pipeline. The purpose of scripting is to bring together the different modules of the 3D reconstruction pipeline and interface them appropriately by automating the process of passing the output of one module as input to another. The script automates the process from accepting an input omnidirectional image

to outputting a mesh for the input scene with the material mapped accordingly to the mesh. In essence, the whole machine-learning pipeline is automated up until the point of importing the mesh into Unity. The machine learning pipeline namely includes depth estimation, material recognition, and material mapping modules.

All the different pipeline components rely on different independent environments (platform-based and language-based), which the script takes care of activating and deactivating appropriately to run the different files for the modules. Different modules had different specifications for the systems they would run on. Depth Estimation uses a Linux-based environment, managed with Docker. DBAT model uses Windows, whereas EdgeNet360 for semantic scene completion and material mapping uses Linux. To run the EdgeNet360 model on the laptop provided, Windows Subsystem for Linux (WSL) was used which provided the feature to run a Linux environment on a Windows machine without the need for a virtual machine or the need to switch to a different physical machine. Using the batch script different commands were implemented to run the models in their required systems. The use of WSL made the implementation of batch files very smooth and simplified the process of switching between the Windows and Linux environments as the Linux environment was run from the same machine itself. The batch script also automatically activates and deactivates the appropriate Python environments for the models.

Automating Docker was a slightly challenging process since the 360 image input and the output depth maps produced are contained within the Docker container for 360MonoDepth, sandboxed away from the main OS. It was impractical to rebuild the Docker container every time with modified inputs due to computational and storage constraints and it was also necessary to programmatically retrieve the generated outputs. Automation was this component was not primarily possible to achieve through the use of batch files due to the way Docker works. Instead, a combination of Powershell scripts was used for this part, using `masterscript.ps1` to execute all of the other depending Powershell scripts sequentially. This involves mounting said Docker image in interactive mode, copying the desired input image into the correct folder and modifying the `erp_00_data.txt` to reflect this. Once this is done, the following PowerShell script executes the actual component and performs inference, which generates depth maps that are then copied out of the interactive Docker container with the help of another Powershell script. Following this, the interactive container is killed (which is crucial to prevent issues with Docker on a re-run). The `masterscript.ps1` file is however invoked through the `combined.bat` file making the process of automation seamless.

After which `split_img.py` python script splits the input omnidirectional image into the 6 faces. Then it runs material recognition on these images using the DBAT model and combines the output images using the `combine_img.py` python script saving the material recognised omnidirectional image in the required EdgeNet360 directory. After which the EdgeNet360 model is run, by providing the input omnidirectional image, depth estimated image and the material-mapped image, to output the mesh with the scene reconstructed as a mesh and the materials mapped to the mesh according to the material-mapped image, automating the whole process.

The script created automated the scene reproduction process but was not user-friendly. Therefore, a graphical user interface(GUI) was implemented as a front end, to make it easy for all to use the technology. The design of the GUI was

minimal so that it would be intuitive and for immediate usability. The GUI only contains 4 clickable options. The 2 checkboxes allow the options to either run the pipeline with the state-of-the-art 360MonoDepth depth estimation model or run using images from the Stanford2D3D dataset. The option to run the Stanford2D3D dataset was included such that for future implementations, troubleshooting would be effortless.

Once the user selects the image and clicks run, these user-dependent options are passed into the script, where the rest of the pipeline is run. Due to the minimalist nature of the GUI, very few feedback messages were implemented in the GUI, and the user was still required to track the command prompt to verify the pipeline ran smoothly.

Overall, the GUI provides the user with a three-click functionality to input an omnidirectional image and obtain a mesh with the corresponding material and depth-based structural properties, within 15 minutes.

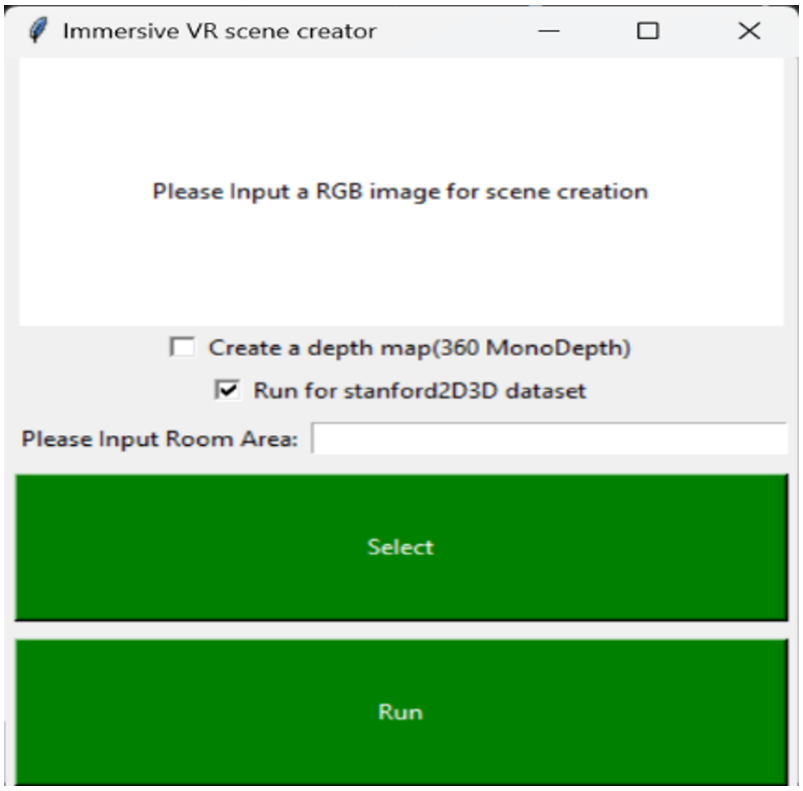


Figure 3.26: Screenshot of the pipeline's Graphical User Interface (GUI).

4 Results

4.1 Introduction

This section initiates an in-depth examination of the pipeline, focusing on the audio-visual aspects. The evaluation protocol is strategically structured, bifurcating the analysis into two discernible dimensions: visual and audio evaluation. This approach aims to scrutinise the efficacy of the integrated system within the realms of both visual and auditory experiences. The subsequent sections will expound upon the methodologies and findings associated with each domain.

4.2 Visual Evaluation

4.2.1 Monocular Depth Estimation

Normalisation of the depth ranges was performed using the example DWRC RGB Image and the ground truth depth map that comes with EdgeNet. When EdgeNet is first initialised with its inputs, depending on the depth map, it outputs predicted dimensions for the room dimensions of the scene in question. Originally, the predicted dimensions for the non-normalised greyscale depth map using 360MonoDepth were significantly higher than those produced by the ground truth depth map.

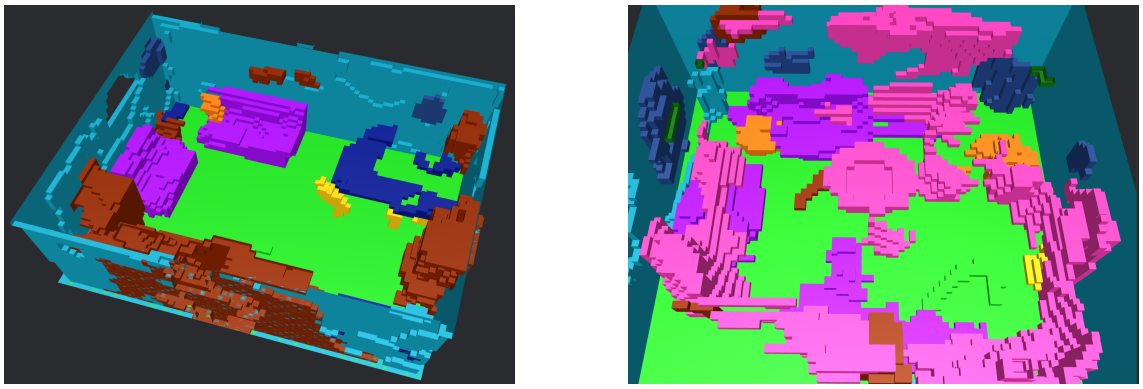
In most cases, this produced problems with EdgeNet reconstruction where a Marching Cube algorithm error which would cancel the generation of the voxel grid and the meshes. Normalisation was a crucial process in changing this. It involved a long process of trial and error where the minimum and maximum depth ranges for 360MonoDepth were incremented and decremented respectively by a constant amount each time, adjusting the ranges of the depths and thereby reducing the inflated predicted room dimensions.

This was done until a range was found which got results as close to the dimensions produced by the ground truth depth map. This can be seen in the following table. Height, Width and Length in the table refer to the room dimensions.

	Height	Width	Length	cmap_LB	cmap_UB
Ground Truth	2.30	4.21	5.34	N/A	N/A
360MonoDepth	18.57	31.35	51.61	0	1
360MonoDepth	3.21	6.87	8.54	0.2	0.8
360MonoDepth	2.57	5.72	6.68	0.25	0.75
360MonoDepth	2.35	5.17	5.70	0.30	0.70
360MonoDepth	2.32	5.07	5.52	0.3113636364	0.6886363636
360MonoDepth	2.30	4.99	5.44	0.3198863636	0.6801136364

Table 4.1: How the room dimensions vary with different depth ranges, cmap_LB and cmap_UB refer to the lower and upper bounds of the normalised relative depths from 360MonoDepth

These results are reproducible and the final row of the table was used for the final normalised depth ranges since they generate dimensions as close as possible to the ground truth results. The results of the normalised depth map, produced through this adjusted custom colour scheme, not only solved the marching cube algorithm problem with EdgeNet but also displayed promising outcomes overall with SSC. However, it falls short of achieving the performance achieved with the depth map provided along with the test set that comes with EdgeNet360, where the depth map is constructed using stereo matching.



(a) EdgeNet prediction using ground truth depth map.

(b) EdgeNet prediction using normalised 360MonoDepth depth map.

Figure 4.1: Comparison of output generated by EdgeNet using the ground truth depth map and the best output generated by the normalised 360MonoDepth depth map

The ground truth depth map from the EdgeNet test set produced using Stereo matching has a lot of missing data, highlighted by various black patches all across the image, indicating a lack of depth data, forming notable non-uniform edges around the objects in the scene. Despite this, EdgeNet360 performs almost perfectly on this depth map. A small experiment was hence conducted in hopes of further improving the performance using 360MonoDepth. Edge Detection was performed on the original image and the edges were superimposed over the normalised depth map generated by 360MonoDepth at varied opacities and tested with EdgeNet in hopes of further improving the results.

Figure 4.2: Edge Detection result on DWRC indoor image

Figure 4.3: Edges superimposed on the normalised 360MonoDepth Map at 30% opacity

Figure 4.4: Edges superimposed on the normalised 360MonoDepth Map at 70% opacity

However, this particular approach produced a significantly worse outcome with EdgeNet.

While 360MonoDepth showcases significant advancements in the Monocular Depth Estimation of 360 images, the decision to further explore alternative approaches stems from the recognition of ongoing challenges with SSC and the pursuit of further possible improvement. In conclusion, 360MonoDepth stands as a robust framework for monocular 360 depth estimation, excelling in handling high-resolution images from more modern datasets or photos taken using modern stereoscopic cameras. However, the intricacies of depth representation, normalisation, and integration with components like EdgeNet360 highlight the ongoing effort for more refined solutions.

Monocular Depth Estimation is an emerging and ever developing technology, with the likes of UniFuse and 360MonoDepth presenting themselves as versatile options that produce higher resolution outputs with quality depth estimation compared to other similar works. A comparison can be seen in Figure 4.5

(a) Normalised Depth Map from 360MonoDepth

(b) Normalised Depth Map from UniFuse

Figure 4.5: Comparison of depth maps generated by 360MonoDepth and UniFuse

When normalised, the depth map produced by UniFuse still has further details on objects in the background, but it is observable that the objects detected themselves appear rather flat. The depth ranges on the objects themselves are substantially worse when normalised. This explains why the normalised 360MonoDepth depth map produces the best (but not desirably precise) SSC reconstruction.

Limitations

Through thorough research and development, several depth estimation solutions were tried and tested, and despite this, they didn't produce the desired results with SSC to a sufficiently precise standard. Due to the nature of depth estimation technology, it is likely that it is a limitation of the technology itself. It is just as likely that the fault lies with EdgeNet in the way it is designed to infer over a given depth map. It may be the case that there is a loss of translation between the extremely smooth noiseless depth maps produced by Monocular Depth Estimation networks and the type of noisy patchy depth maps that EdgeNet has been built to work with.

This underscores the necessity for continued research and development to refine existing MDE models and further enhance them to address the intricacies of this technology. Further research and development in the realm of monocular depth

estimation is necessary to enhance the overall performance and reliability of these systems. Future work should focus on examining precisely why EdgeNet360 appears to perform as poorly as it does with monocular depth maps and see if further post-processing can be applied to Monocular Depth maps to make them compatible with EdgeNet in a way where the reconstruction largely and accurately embodies the scene in the original RGB image.

4.2.2 DBAT Evaluation

Testing

The complete process of material recognition starting from the input omnidirectional/equirectangular image to the output omnidirectional image with materials identified is shown in figure 4.6.

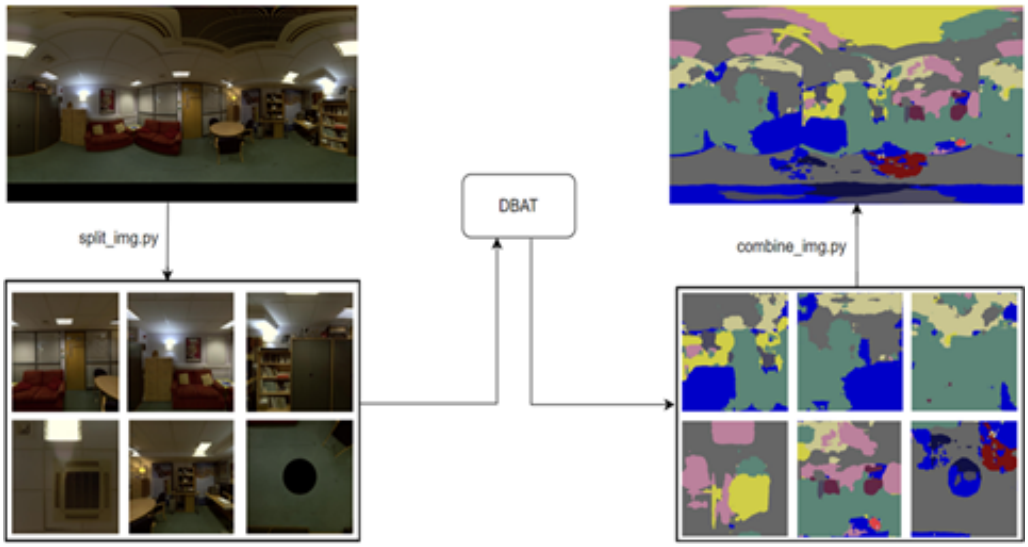


Figure 4.6: Pipeline for Material Recognition

The model was tested on several 360-degree images to infer the model's performance with projected 2D images. Overall, the model was able to identify materials present in the image fairly accurately. However, in some instances, major materials identified in the image which covered large areas tended to spread over onto other objects that did not belong to the material. The precision for material assignment is not to a high degree as often objects are assigned the material label of other more prominently present materials in the image. Issues also arise with the model being confused with identifying the material of glass. In cases where there was an object present behind a transparent surface, the model identified it as either glass or the material present behind it.

From Figure 4.6 it is observed that the model can identify the materials correctly on a larger scale for most materials, assigning correctly the sofa as fabric, most furniture as wood, the table and door as wood, and the wall as plaster. The bottom floor and ceiling however are inaccurately identified with the floor having plaster instead of fabric and the ceiling having plaster and metal even though not present in the scene. It also seems to identify water and asphalt in small areas in the scene that are clearly not present. The model also does not have sharp material boundaries in

most instances with them spreading across. However, it identifies materials correctly on a larger scale.

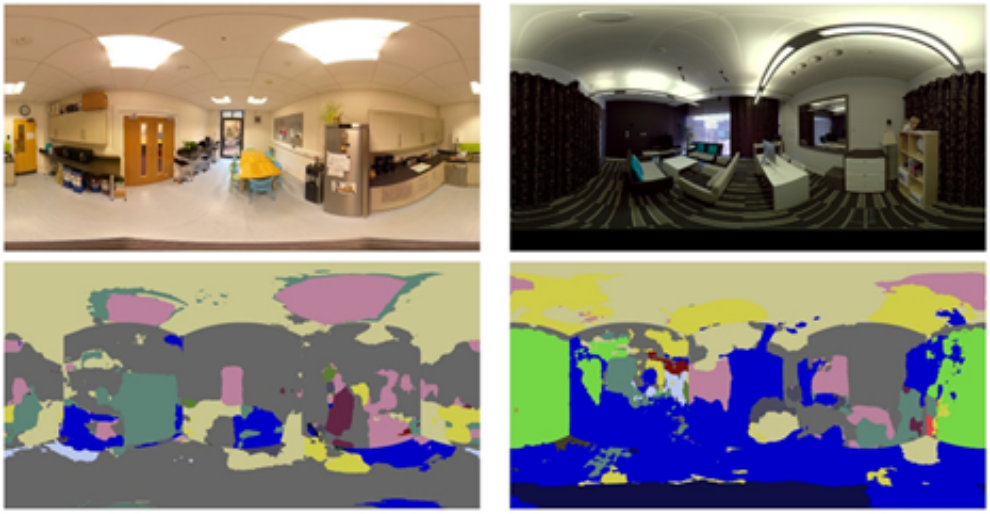


Figure 4.7: DBAT inference on omnidirectional images, with the input images on top and their corresponding material recognised image on bottom.

Figure 4.7 shows the material recognition for a few scenes with the input image on top and its corresponding material recognised output image on the bottom. From figure 4.7, it is observed that the model performs fairly accurately in most cases, identifying major materials correctly. The door is correctly assigned wood, the light panes on the ceiling as glass, and the wall as plaster in the image on the left. However, the fridge is not assigned metal and has a mixture of materials assigned to it, and the table is not assigned wood. The image on the right in figure 4.7 is correctly assigned fabric for the carpet on the floor, the window is assigned glass in all the areas, and the wall where visible is assigned plaster. However, the curtains on the walls have been incorrectly identified as stone (green) material.

The DBAT model was also tested on images from the Stanford2D3D dataset. The performance of the model on some of these are shown in figure 4.8. From the inferencing figure 4.8, it was observed that DBAT performed quite well in recognising the materials of major objects in the scenes. This is especially seen in the image on the left for which the open door on the far left in the image has been correctly identified as wood even though not clearly visible. Also, the shelf's boundaries on the far left were again detected correctly as wood. However, the model struggled on occasions with identifying materials, failing to identify the whole object as belonging to one material even though prominently visible, as seen in the image on the oeft in figure 4.8 where the table in centre is not completely recognised as wood with the fabric from floor blending into it. Also, in the image on the right in figure 4.8 again the whole table is not identified as wood and the whiteboard is incorrectly labelled as asphalt.

Another artefact introduced in the output images for the stanford2D3D dataset was the black strips at top and bottom in the images which were always detected as fabric. From figure 4.9 it is seen that the top and bottom faces have a black artefact present at the centre, this is the position where the camera would have been present. This is a common occurrence in all the images for the stanford2D3D dataset. This

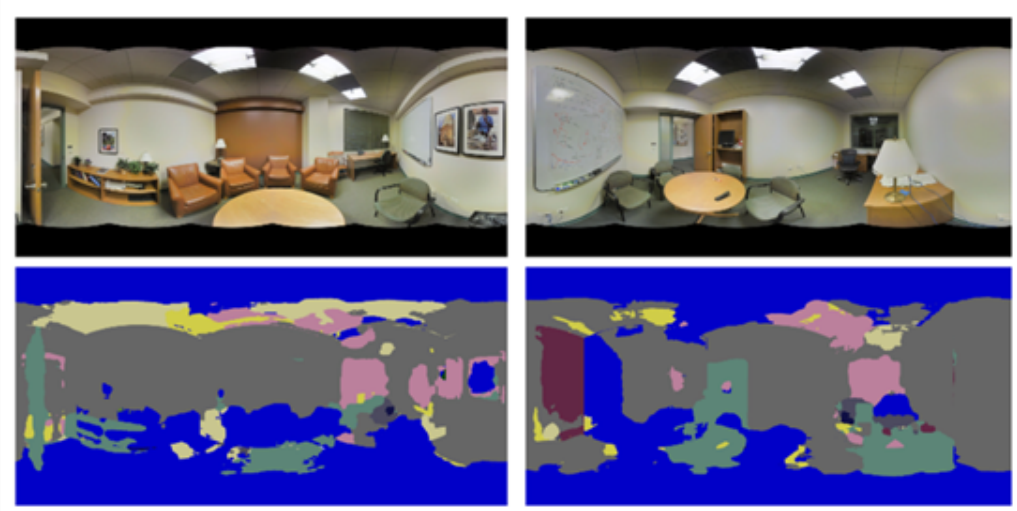


Figure 4.8: Performance of the DBAT model on some images from the Stanford2D-3D dataset, with the input images on top and their corresponding material recognised image on bottom.

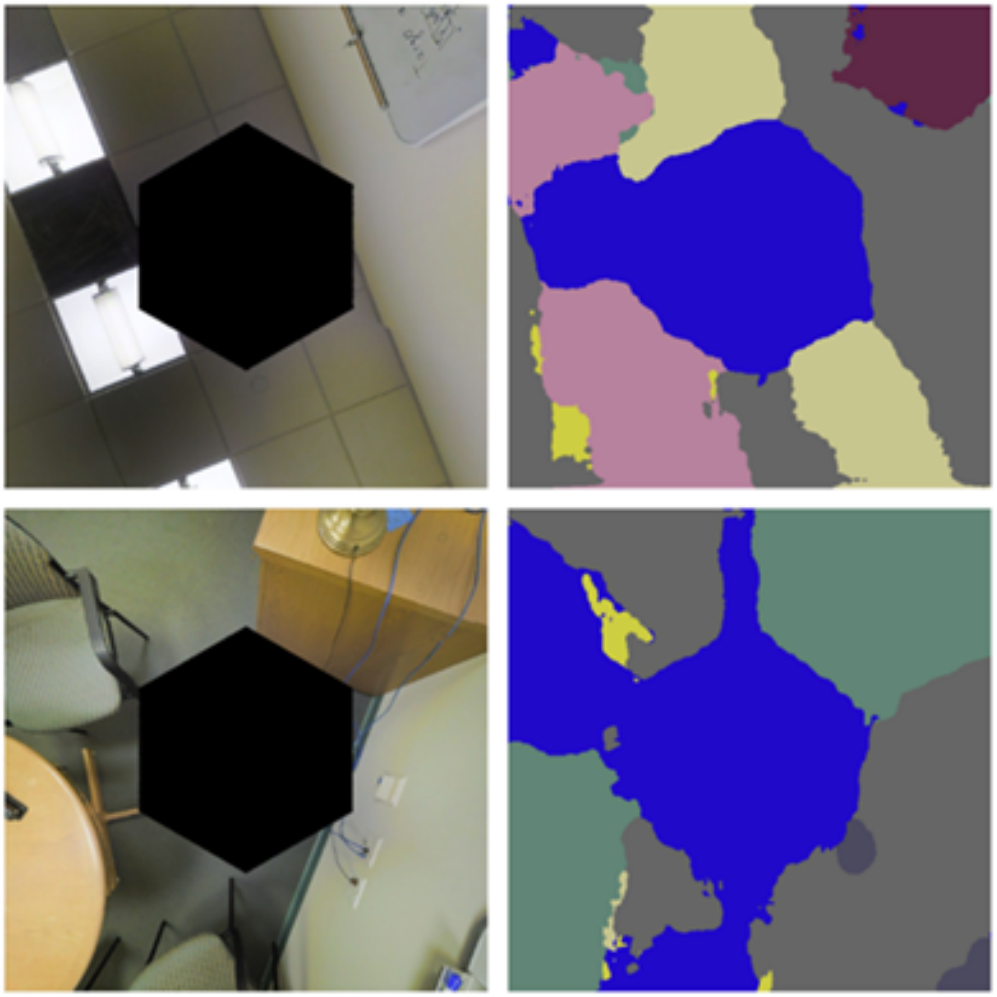


Figure 4.9: The performance of DBAT for the top and bottom face of an image from the Stanford2D-3D dataset

would explain DBAT recognising the black top and bottom strips as fabric for all images, the black artefact present in all images being the issue. Since the DBAT model is not trained on such images it would be detecting the black artefact as fabric randomly. However, why this is always fabric is unknown. Although it is seen that DBAT still manages to recognise the table in the bottom face as wood and the glass panes for the lights in top face as glass, showing its better performance on the Stanford2D3D dataset. Overall, the model can detect the materials accurately to a good extent in most instances at a higher level with occasional shortcomings and some struggles on a smaller level.

Evaluation and Limitations

Based on the above observations and the inferencing done on different input images DBAT's performance on 2D projected images for 360-degree images is summarised in the following paragraphs. Often there were materials assigned to objects that were not present in the scene at all, or incorrect materials assigned to objects or certain parts of the objects. The model could identify the materials to a good degree on the higher level but failed in most instances to capture intricacies on the lower level i.e., smaller objects/details. Therefore, it is better at capturing the global context than local material recognition. The performance could be described as par in most instances.

The model's poor performance in some instances could be accounted for by the fact that when the images were split for the cube map projection, the objects, furniture, and areas could have been split into different faces partially based on their presence in the scene and if present at the boundary of faces. This would get rid of the global contextual information present in the image and the local material information in some cases, which provides the model important information and helps identify materials more accurately. It can make the objects look discontinuous and split up (not representing their actual characteristics visually) when viewed on its own. [33] explain this, "materials are recognised by identifying the actual objects they make up (e.g., "mirror," which is an object, is identified as a material by finding mirrors, "leather" is recognized by finding sofas and ottomans, and "fabric" is identified by recognizing pillows) [33]" and "global image context, including both what the object is and where it is, can provide rich information to narrow down what things and stuff are made of [33]". Therefore, causing the material assignment to be inaccurate in some cases.

Another shortcoming observed in DBAT's performance on 360-degree images was specifically on the 'Top' and 'Bottom' faces of the projected images. These faces/areas had incorrect materials assigned to them in most instances as seen in figures 4.10 and 4.7. The top and bottom faces in the cube map projection normally consist of the ceiling, roof, sky (top), and floor, and ground (bottom). These areas lack context/features found in the other faces, which usually contain objects, furniture, and walls which provide more features and material clues. When the top and bottom faces are viewed on their own, they would usually contain smooth areas with the same materials and minimal objects due to their position in the scene and lack of global and surrounding context as seen in figure 4.9, 4.10 and 4.6. Also, the splitting of the images could cause some areas of the ceiling and the floors to be split into other faces apart from the top and bottom, again removing context. [34] found that

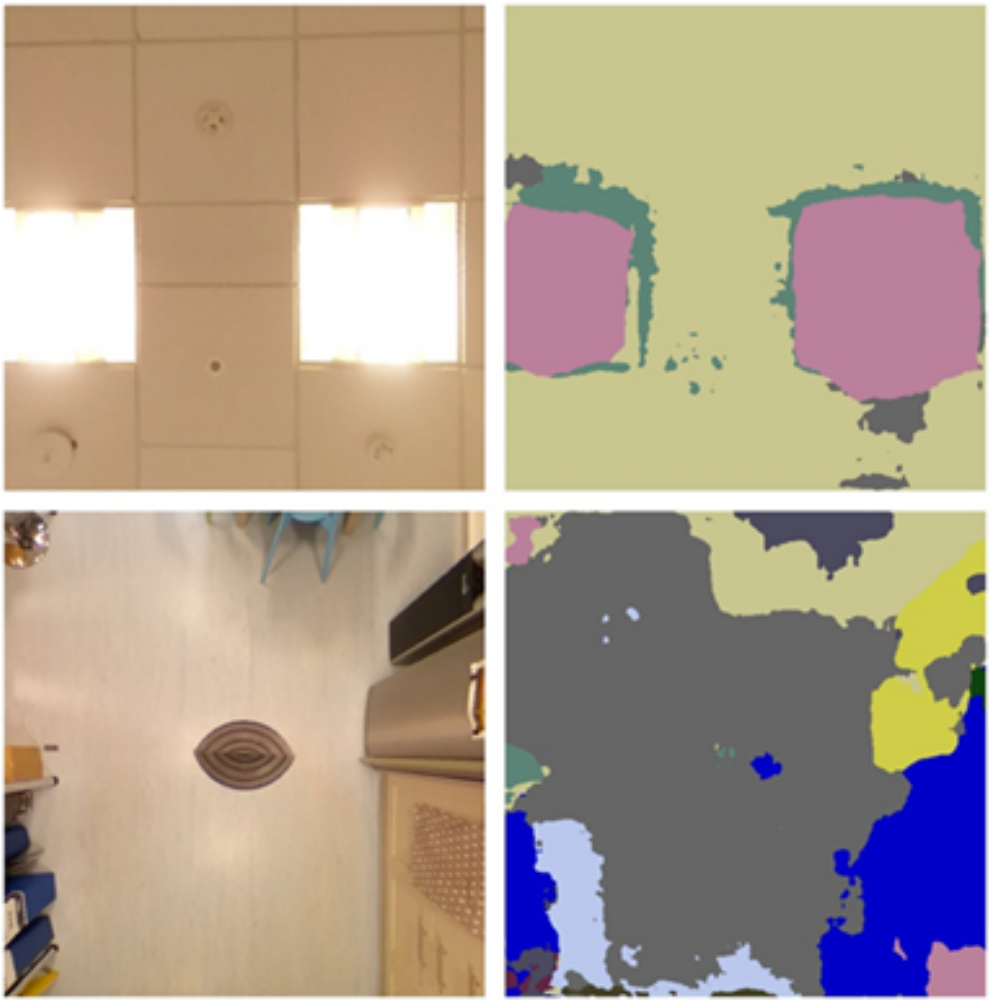


Figure 4.10: The top (in the top row) and bottom (bottom row) faces (for the image on the right in Figure 4.7) with their corresponding material recognised image on their right

"surrounding context is crucial for real-world material classification [34]", emphasising the importance of context for material recognition, especially in this case where the context is likely to be limited due to the splitting of the images. Hence, explaining the observed performance of DBAT on the top and bottom faces. Another thing to note would be that DBAT is not trained on such isolated images for just top and bottom faces and images with black artefacts present in them as seen in figure 4.9 and 4.10.

Conclusion

Overall, the performance of DBAT on 360-degree images (2D projected faces of 360-degree images in fact) can said to be fairly accurate for all faces or more precisely areas, apart from top and bottom in most cases. Also, the 2D projected images can be split at different points in different images and the DBAT model is not necessarily trained on such data. Given this, the performance can be considered good. The model is able to identify materials in images to a good degree for larger more prominently visible objects and to a lesser degree for smaller objects in most cases.

Being better at capturing materials from a global context than local materials. Due to the lack of other state-of-the-art models for material recognition for the specific task of 360-degree images, the model evaluation is only done with the DBAT model and not compared to other models. However, much work is required to improve the model performance on the top and bottom faces and the material recognition accuracy of the model in general. The model can capture the materials of the scene at a larger level quite well.

Due to the short timeframe of the project combined with the uncertainty and complexity associated with the next task of material mapping, it was decided to move to and focus on the material mapping task and use the material recognition model provided with the baseline performance. It was decided to improve the DBAT model (i.e., a technique to improve material recognition of top and bottom faces/areas) at the end after the whole pipeline was working if time permitted. However, due to the material mapping taking a long time for implementation something which was factored for, the DBAT model performance was accepted as a limitation of the model provided. This helped us focus on and achieve a working product (one of the agile values) in the end.

4.2.3 Automatic Material Mapping

Various tests were carried out during and after the implementation of the material mapping to evaluate the outputs. These tests were evaluated visually and in a subjective manner to validate the accuracy of the results and to evaluate the performance of the mapping within the context of the scenes. Multiple scenes were tested upon to cross-validate the results to make sure the mapping functionality was not biased to select scenes.

Volumetric Encoding of Materials Test

The edited TSDF function now generates an additional material projected voxel grid, with material labels assigned to each voxel. However, it was not possible to verify the accuracy of the mapping functionality visually.

Therefore, as a solution to confirm the process was on track, a frequency counter was implemented which calculated the frequency of each material label assigned per view and ranked them in descending order. These views represented 8 different sections of the scene, as EdgeNet360 split the voxel structure and carried out semantic segmentation and scene completion for each view individually. Cross-checking the most dominant material labels for each view with the most prominently classified materials in the material map verified, in a general sense, that the material assignment to the voxels was on the right track. This did not provide a foolproof test that the materials assigned were perfectly correct but instead provided a sanity check, serving as a tool for debugging the implementation of the material label assignment to the voxels in the grid.

Figure 4.12 shows the frequency table for view 1 and view 2 for the material map input. From the frequency table, it is seen that grayscale values 119, 23, 102, and 149 are the most prominent materials for these views which correspond to wood, fabric, plaster, and glass respectively. Since the material map image, as seen in figure 4.12, mainly contained wood, plaster, and fabric, this ensured that the mapping

was not random and correlated with the scene. Additionally,from the list of the frequency, it is noted that the presence of materials such as concrete, rubber, glass, and paper indicate that view 1 and view 2 corresponded to the right-most section of the material image, corresponding to the area containing the bookshelf, posters and computer in figure 4.11.

Material	**Grayscale Value**	**Colour**
asphalt	47	
ceramic	193	
concrete	200	
fabric	23	
foliage	106	
food	88	
glass	149	
metal	191	
paper	60	
plaster	102	
plastic	77	
rubber	22	
soil	63	
stone	169	
water	114	
wood	119	

Table 4.2: Materials and their colours.

Figure 4.11: Image for the living room scene.

View 1:	View 2:
Rank: 1, Value: 0, Frequency: 87220	Rank: 1, Value: 0, Frequency: 99439
Rank: 2, Value: 119, Frequency: 13518	Rank: 2, Value: 23, Frequency: 14245
Rank: 3, Value: 23, Frequency: 12980	Rank: 3, Value: 102, Frequency: 7734
Rank: 4, Value: 102, Frequency: 10718	Rank: 4, Value: 149, Frequency: 6173
Rank: 5, Value: 149, Frequency: 2809	Rank: 5, Value: 119, Frequency: 1137
Rank: 6, Value: 191, Frequency: 1212	Rank: 6, Value: 193, Frequency: 417
Rank: 7, Value: 193, Frequency: 957	Rank: 7, Value: 191, Frequency: 286
Rank: 8, Value: 200, Frequency: 145	Rank: 8, Value: 200, Frequency: 145
Rank: 9, Value: 77, Frequency: 25	Rank: 9, Value: 22, Frequency: 24
Rank: 10, Value: 22, Frequency: 15	
Rank: 11, Value: 60, Frequency: 1	

Figure 4.12: Top: Table showing the frequency of the materials present in the voxels grid in descending order for views 1 and 2, Bottom: Material map for the living room scene in figure 4.11.

Material Assignment

Another subjective test was carried out after the whole material encoded voxel grid, with all 8 views combined, was created to assign each object classified in the scene a material value. Majority voting was used as explained previously to assign the most frequent material label for an object as the material for that object. This object-based material assignment was printed on the terminal as shown in figure 4.13 for the same material map in figure 4.12 and visually verified with the material map to verify if the objects were assigned the same materials as in the material map. It is observed that the sofa was correctly assigned the material of fabric in the voxel grid as in the material map. Also, correctly assigning floor as fabric, window as glass, etc. The majority voting-based material assignment for objects also removed outliers from DBAT's material recognition in some cases. As seen here, in the material map in figure 4.12 the floor was assigned various materials such as fabric, plaster, asphalt, and ceramic instead of just fabric as it should have been as seen in the actual input image in figure. However, the object-based majority voting assigns the whole floor as fabric getting rid of the anomalies.

```
Material assigned to object->  {'floor': 'fabric', 'wall': 'wood', 'furniture': 'wood',
 'table': 'wood', 'objects': 'wood', 'ceiling': 'plaster', 'chair': 'fabric', 'window':
 'glass', 'sofa': 'fabric', 'bed': 'fabric'}
```

Figure 4.13: Materials assigned for the living room scene in figure 4.11.

Stanford2D3D Dataset

Further tests were conducted on the Stanford2D3D dataset to validate the functionality of the automatic material mapping. The Stanford2D3D images that were used for testing were all scenes depicting an indoor room in area 3. The results obtained confirmed that the mapping was accurate and that the mapping was not fine-tuned to work on a select few scenes. The results can be seen below, which clearly demonstrate the capabilities of material mapping.

(a) The omnidirectional image of office 3, area 3, reproduced from [35].

(b) Material segmented image of office 3, area 3, reproduced from [35].

Figure 4.14: RGB and Material Segmentation of Office 3 Area 3 from the Stanford2D3D Dataset.

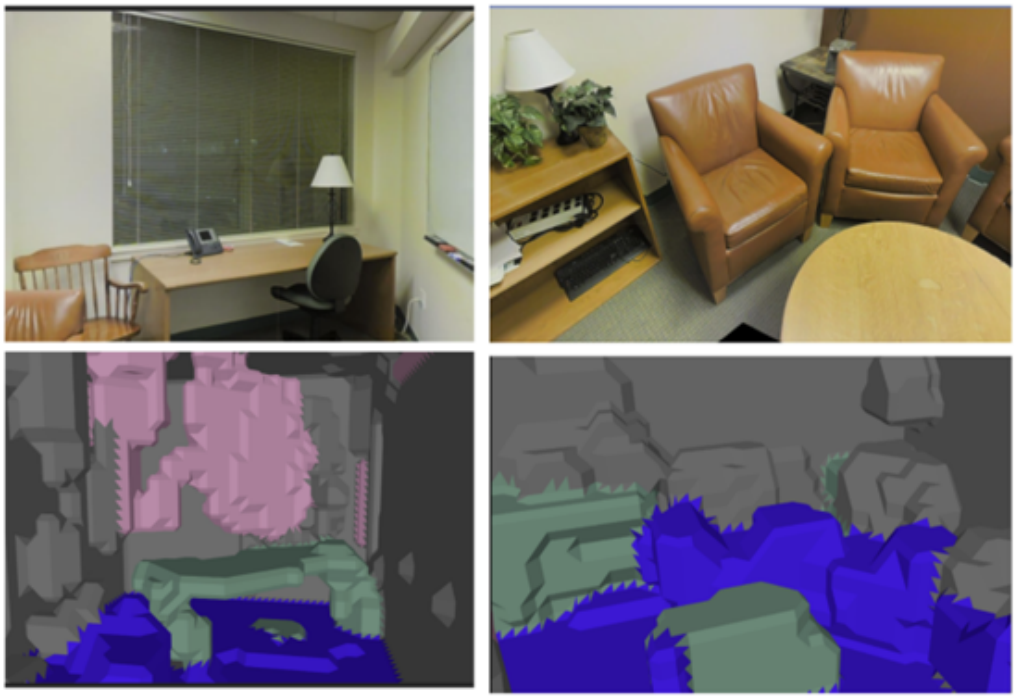


Figure 4.15: Comparison between mesh output and the image fed in by areas of the room.

Figure 4.14b shows the material segmentation of the scene. The DBAT model correctly assigned sofas as fabric, tables, and shelves as wood, windows as glass, and

walls and ceiling as plaster. When the Edgenet model was run with the material mapping, the mesh output clearly demonstrated the mapping was done correctly. Figure 4.15 shows the parts of the mesh output with the same parts of the original scene. With material mapping, the scenes look alike, as seen in the left image, with the materials of the window, wooden desk, and floor matching that of the scene. The position where the wooden chairs are in the scene is not green, but this is due to the object detection limitation of EdgeNet360 rather than incorrect material mapping. The sofas on the right image correctly translated to the mesh, covered by the blue blocks surrounding the wooden desk. Additionally, the desk behind the second sofa is properly translated in the scene, where there is a slight green face behind the blue faces representing sofas.

Similar to Office 3, the generated mesh output, shown in figure 4.17 depicted to a great extent the real scene, with correct mappings of the material onto the objects. Since the objects were classified in an exhaustive list, the lamp was grouped under "objects", and since the majority voting was applied to each category, plaster was applied to the faces covering the lamp. Therefore, to improve the precision of the materials that were applied to the scene, object splitting was implemented.

Figure 4.16: The omnidirectional image of office 5, area 3, reproduced from [35].

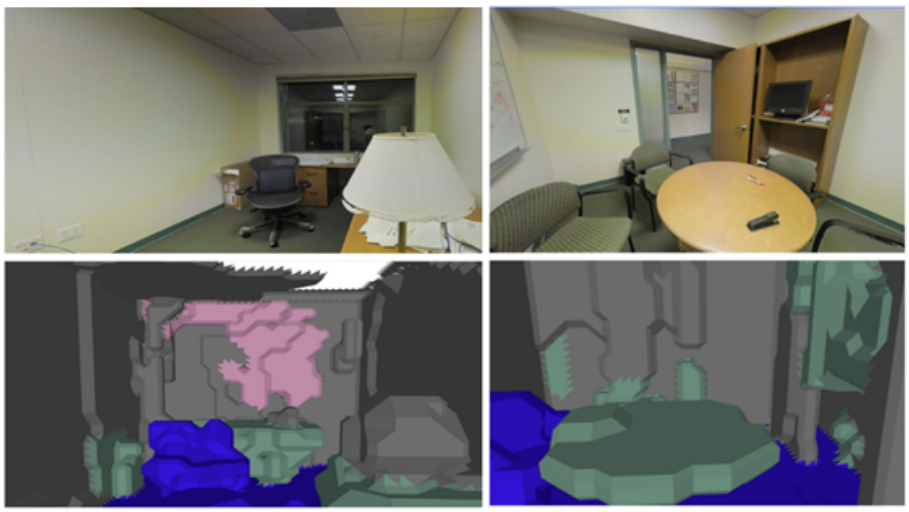
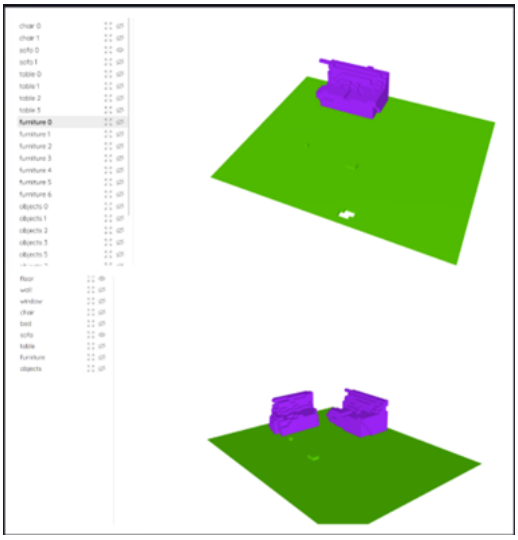


Figure 4.17: Material recognition output of office 5 area 3 represented in figure 4.16.

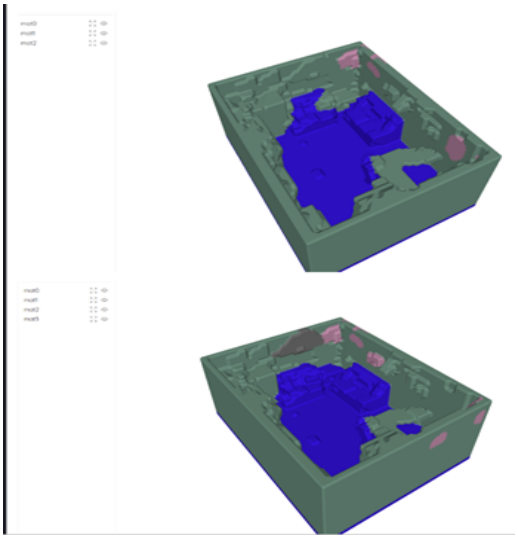
Object Splitting

The Object splitting was tested on a single scene to confirm the functionality.Figure 4.18a displays the outcome of the object splitting. The image on the bottom was prior to object splitting, where the two sofas were part of the same object labelled "sofa". The tab on the left, within the image shows that there were only 9 different objects in the scene (the other 8 are hidden). The image on the top shows the result of splitting. The sofa label has been replaced with the original label and a numeric (in this case sofa 0). The result shows that individual objects have been isolated and no longer grouped together, with the other sofa labelled as sofa 1 (hidden from viewing).

The result of object splitting is also reflected in the final mesh generated, as seen in figure 4.18b. After object splitting, the scene contains more materials, increasing the accuracy of the model closer to the material image from DBAT.



(a) Top image shows the outcome of object splitting, with the right sofa under object label "sofa0", the bottom image shows prior object splitting with both sofas under a single "sofa" label.



(b) Top half of the figure shows the mesh output before object splitting and bottom half shows mesh output after object splitting.

Figure 4.18: Comparison of the meshes before and after object splitting.

Evaluation

The material assignment for objects is only as good as EdgeNet360's object detection. The object detection by EdgeNet360 was not accurate in a lot of the cases. As seen in figure 4.19, where the top left image shows the actual input image, the bottom left image shows the corresponding material map for it, and the image on the right shows the corresponding voxel grid created for it with the different objects identified by EdgeNet360 represented by different colours.

In the RGB image, we see that the window is adjacent to the sofa, but the 3D reproduced scene displays that region as part of the wall (cyan) instead of window (medium dark blue). Additionally, the window(medium dark blue) in 3D voxel grid

has been detected incorrectly to be opposite the sofa. Due to this incorrect detection, the window has been assigned as concrete, as seen in figure 4.20.

However, upon cross-comparing the material map, the region which the voxel detected as window contains several materials such as metal, concrete and asphalt. The majority voting mechanism correctly calculated the most dominant material to be concrete. Hence, the material mapping functionality is accurate but due to the limitations of EdgeNet360, the material assignment of objects are not contextually correct.

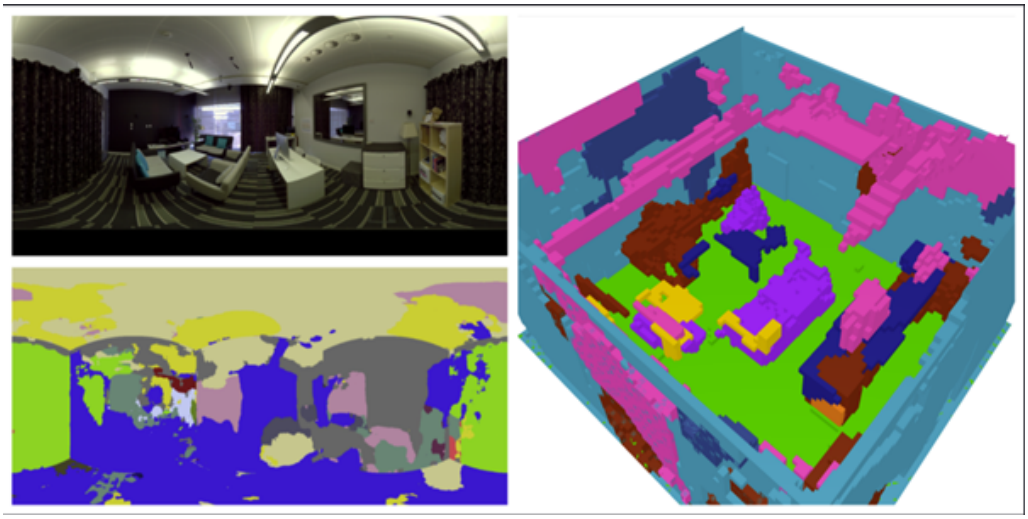


Figure 4.19: Top Left: Image for the usability scene, Bottom Right: Material map for the usability scene, Right: Voxel grid generated for the usability scene

```
Material assigned to object->  {'floor': 'fabric', 'wall': 'plaster', 'ceiling': 'plast
er', 'furniture': 'fabric', 'objects': 'fabric', 'window': 'concrete', 'table': 'fabric
', 'bed': 'fabric', 'sofa': 'fabric', 'chair': 'fabric'}
```

Figure 4.20: Materials assigned for the scene in figure 4.19.

Moreover, the mapping of the materials also depended on the results of the DBAT material segmented image as well. The result in figure 4.12 shows the region which covers the walls of the room is mainly dominated by the colour green, which signifies wood, and hence the material mapping allocates wood to the wall, as seen in figure 4.13. However, in the context of rooms, walls are made of the same material as the ceiling, which would be plaster. This is a limitation with automatic mapping, if the material assigned for a certain object is incorrect in the material map that same material will be mapped in the same position in the voxel grid.

The mapping was tested with another scene, depicting different indoor rooms. The scene was the kitchen, which had different room dimensions to the living rooms, as seen above. The results provided were accurate, as the material image (4.21) mainly consists of plaster and fabric. However, an error was detected when testing this scene. The ceiling was assigned plaster figure 4.22, when the material image, mainly labeled the ceiling region as ceramic.

Reviewing Dourado et al.'s [36] paper, it was stated that the performance of the ceiling reconstruction was not as good as the reconstruction of sofa and other object within the scene, therefore it was approximated to a smooth plane. It can be

Figure 4.21: Material map for the kitchen scene.

Figure 4.22: Materials assigned in the kitchen scene.

hypothesised that the issue causing the poor reconstruction of the ceiling during 3D reconstruction also had an effect on the material mapping to the ceiling. However, this error was not further investigated, since in the context of indoor rooms, ceiling are generally made from a material similar to concrete or plaster. Therefore, the implications of this error were deemed negligible in the context of how realistic the acoustic properties were of the scene.

Another limitation discovered during testing material mapping using results of 360MonoDepth was the absence of materials in certain areas of the mesh scene. Upon testing with the kitchen scene reproduced using the monodepth generated depth map, there were areas within the reproduced mesh, which were black, indicating no material was assigned to that area, as seen in figure 4.23. Upon further investigation, looking at the material-object Hashmap, the object occupying that area was not assigned a material label. Therefore, the voxels that make up that object had no material values assigned, during the volumetric encoding of materials, suggesting that this object was occluded.

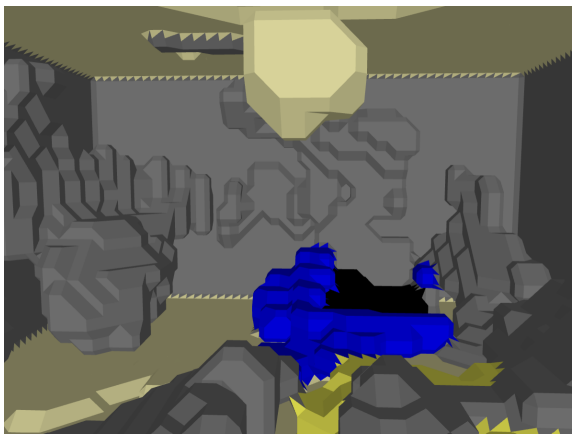


Figure 4.23: Materials assigned in the kitchen scene, in figure 4.7, generated using 360MonoDepth.

On generating the final mesh for the scene in figure 4.11 it is seen that the material mapping worked as desired, and the correct material colours were assigned to the mesh based on the material map and the object-based majority voting for materials. The mesh was also split based on the different materials identified in the scene as seen in figure 4.24, providing the intended outcome, such that material properties can be assigned on Unity.

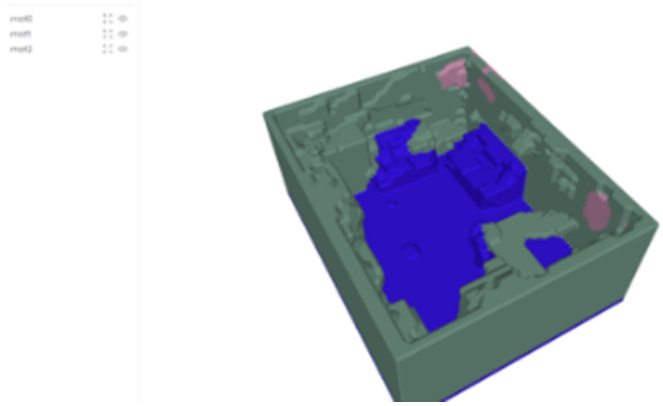


Figure 4.24: The generated mesh output with the material assignment of the living room scene from figure 4.11.

Conclusion

Overall, the automatic material mapping implemented works as required mapping the material identified in the material-map to the exact corresponding coordinates in the mesh. Greatly improving on the previous implementation, where this was done manually. However as discussed in previous sections, its performance is capped by the accuracy of the material recognition model and the object detection (done by EdgeNet360 for SSC). If an incorrect material is identified, then it will map that incorrect material to the corresponding location in the mesh and if an object is identified incorrectly then it will be assigned the material identified for that object in the material-map. In essence, regardless of the issues associated with material-recognition and object-detection the one-to-one mapping works successfully.

4.3 Audio Evaluation

In order for our pipeline's efficacy in reproducing acoustic modelling to be quantitively evaluated, the room impulse response of a scene generated from our meshes was to be measured, in terms of its Room Impulse Response (RIR), as compared to the ground truths generated by the actual room from which an omnidirectional image was supplied.

4.3.1 Scene setup

Terminology

- **Linear waveform:** waveform shown with amplitude scaled linearly between 0 (silence) and 1 (max volume).
- **Decibel waveform:** waveform shown with amplitude scaled to the decibel scale (logarithmic).

A scene of our pipeline's output was manually set up in Unity, loading the following test sounds as the sources, with the waveforms displayed below in Audacity [37] (relative linear and dB scaling):

Source Audio

Gunshot

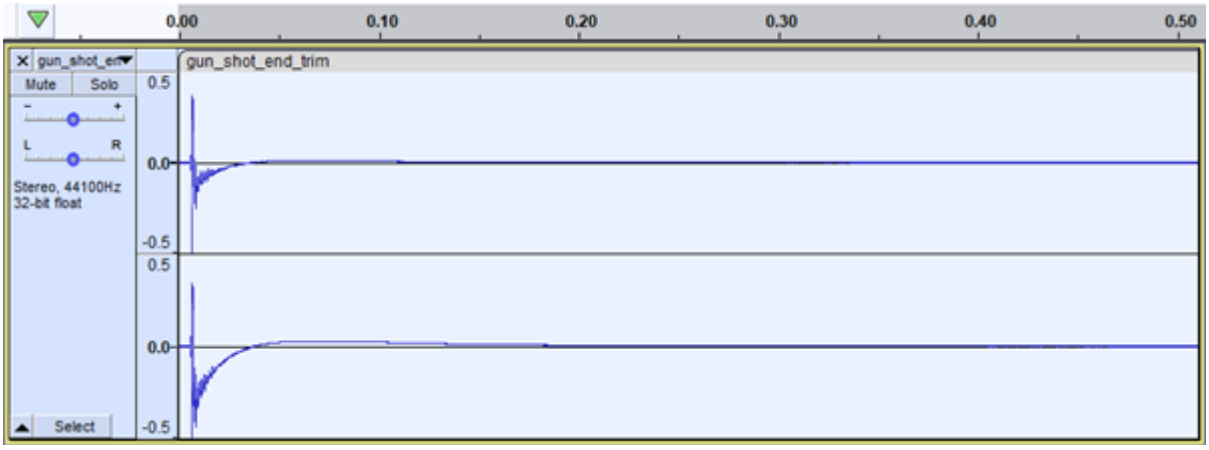


Figure 4.25: A recording of a gunshot in an anechoic chamber (linear waveform), eliminating sound reverberations and leaving a short, high-impulse "click" as the waveform.

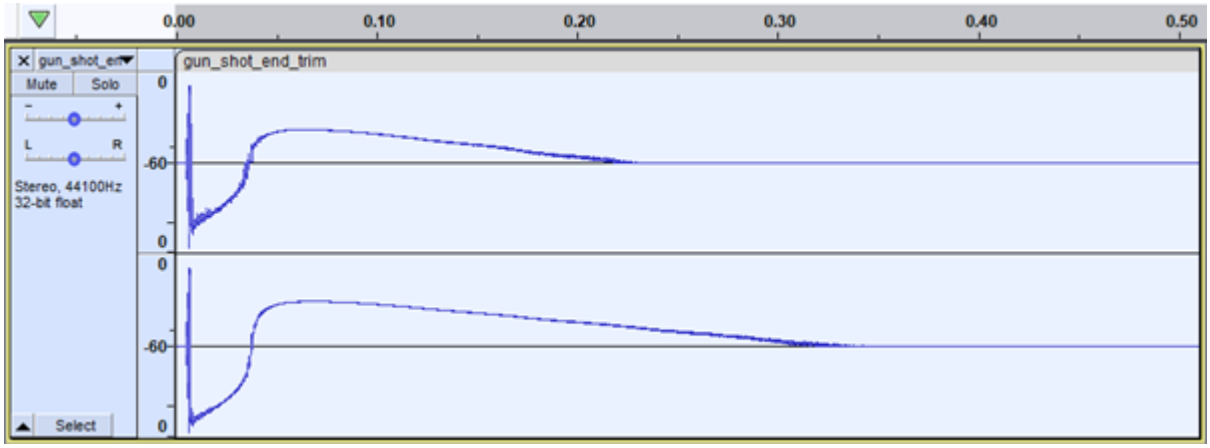


Figure 4.26: A recording of a gunshot in an anechoic chamber (decibel waveform), eliminating sound reverberations and leaving a short, high-impulse "click" as the waveform.

Sweep

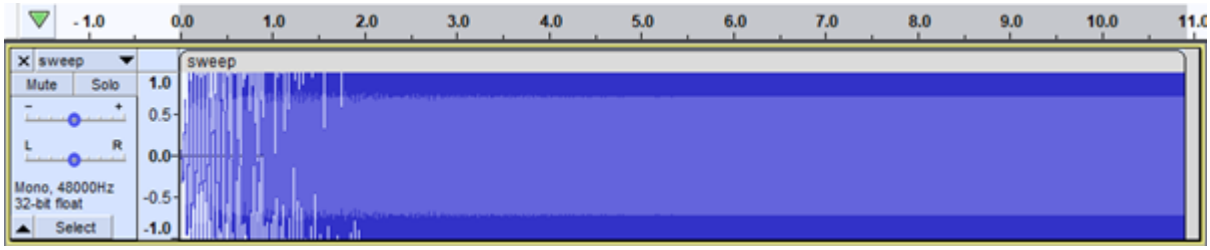


Figure 4.27: A simple generated mono sound wave (linear waveform) that sweeps from low to high frequency, useful for evaluating Room-Impulse-Response at all frequencies.

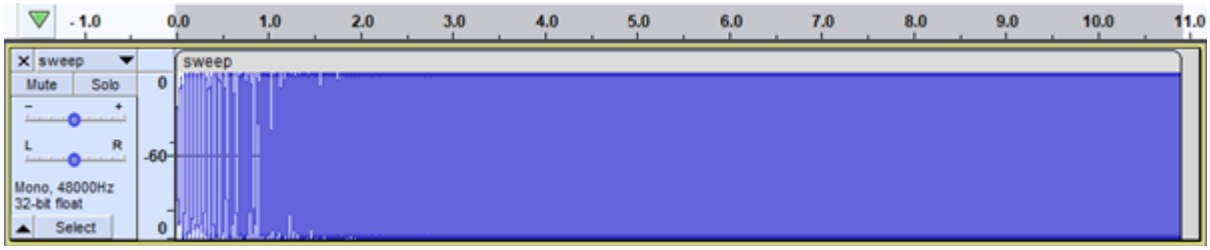


Figure 4.28: A simple generated mono sound wave (decibel waveform) that sweeps from low to high frequency, useful for evaluating Room-Impulse-Response at all frequencies.

Scene Setup

With the kitchen scene mesh (using EdgeNet360 at this stage rather than 360MonoDepth) loaded, manual positioning of the coordinates of the ground truth setup had to translated to be translated to the relative scale (bounds calculated within unity). The diagram in figure 4.29 displays the scene as recorded for ground truth results.

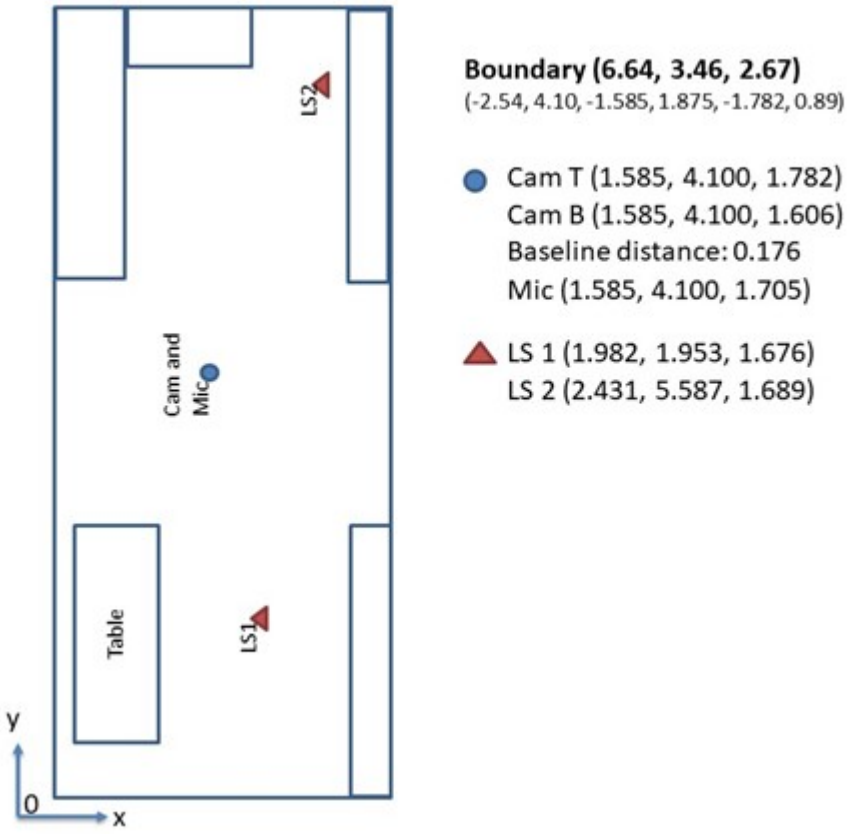


Figure 4.29: The provided diagram of the kitchen with the coordinates of the microphone/cameras, and sound sources LS1 and LS2. Note that the diagram's boundary coordinates are in the form (y, x, z) with all other coordinates in the form (x, y, z).

Unity Coordinate Calculation

Note that the format for each location differs such that each of the diagram's values (y_{ref}, x_{ref}, z_{ref}) correspond to the equivalent (z, x, y) in the Unity coordinate system. The following calculations demonstrate the conversion from the reference coordinates upon import of the mesh created with non-MonoDepth (ground truth) depth map, but the process was generalised to all meshes imported to obtain the results.

Bounds

- **Reference Bounds:** (y_{ref}: 6.64, x_{ref}: 3.46, z_{ref}: 2.67) in metres.
- **Unity Scene Bounds:** (z: 7.52, x: 5.68, y: 2.96) in units.

The latter is calculated with the following code excerpt, which encapsulates the combined bounds of the imported mesh and its sub-meshes:

```
void GetMeshBoundVertices(GameObject targetObject)
{
    // get all meshes in parent object
    MeshFilter[] meshFilters = targetObject
    .GetComponentsInChildren<MeshFilter>();
    // holds mesh bounds
    Bounds combinedBounds = new Bounds(
    meshFilters[0].transform.position, Vector3.zero);

    // iterate through each sub-meshes bounds
    foreach (MeshFilter meshFilter in meshFilters)
    {
        combinedBounds.Encapsulate(meshFilter.sharedMesh.bounds);
    }

    // get world coord scaled bounds
    Vector3 bounds = targetObject.transform.TransformPoint(
        new Vector3(
            combinedBounds.max.x - combinedBounds.min.x,
            combinedBounds.max.y - combinedBounds.min.y,
            combinedBounds.max.z - combinedBounds.min.x));
}
```

Therefore, the Scale Factors ($bound_{ref}/bound_{Unity}$) = (SF_Z: 0.88, SF_X: 0.61, SF_Y: 0.90).

Coordinate Conversion

The bottom left (origin) coordinate reference, O_{ref}: (y: 0, x: 0, z: 0), corresponds to O_{Unity}: (z: -3.88, x: -3.00, y: 0.00), calculated by the following code snippet:

```
Vector3 bottomright = targetObject.transform.TransformPoint(
    new Vector3(
        combinedBounds.max.x,
        combinedBounds.min.y,
        combinedBounds.min.z));
```

Each coordinate, referred to as ref (for the microphone, LS1 source, and LS2 source) was then placed at the following calculated coordinates (where O refers to the origin coordinates within Unity):

$$\text{unity coordinates} = \left(\mathbf{z} : O_z - \frac{ref_y}{SF_Z}, \mathbf{x} : O_x - \frac{ref_x}{SF_X}, \mathbf{y} : O_y + \frac{ref_z}{SF_Y}\right) \tag{4.1}$$

4.3.2 Screenshots of the Scene Setup

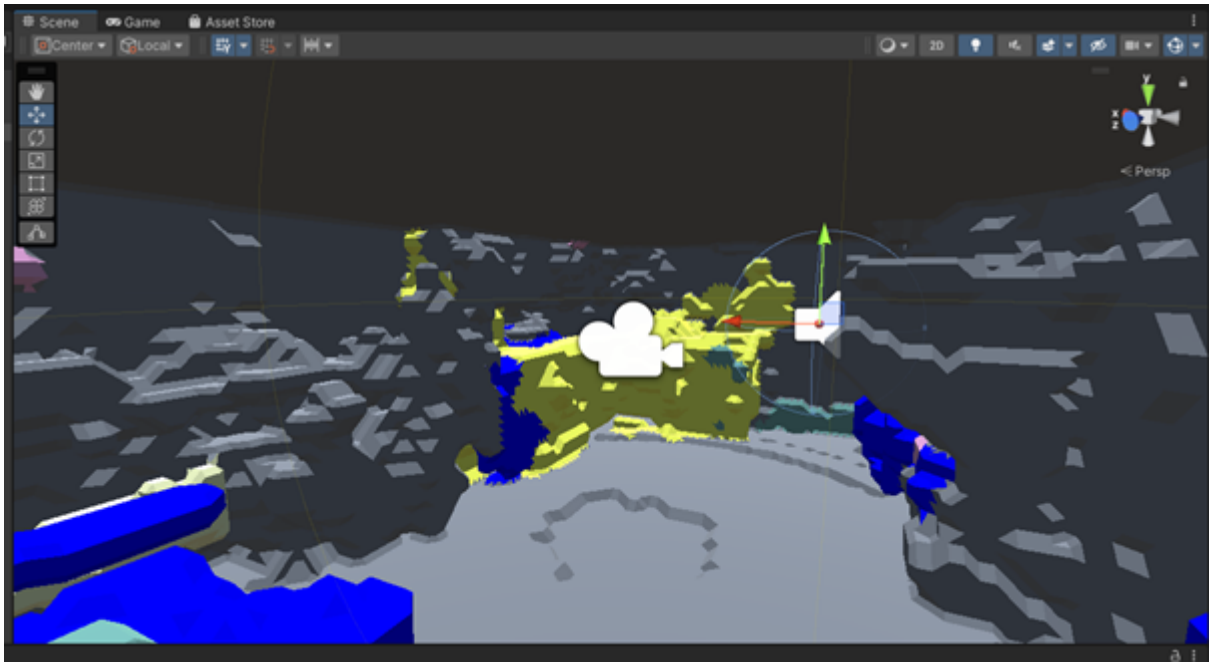


Figure 4.30: Editor view of ground-truth depth map scene with LS2 Audio Source selected.

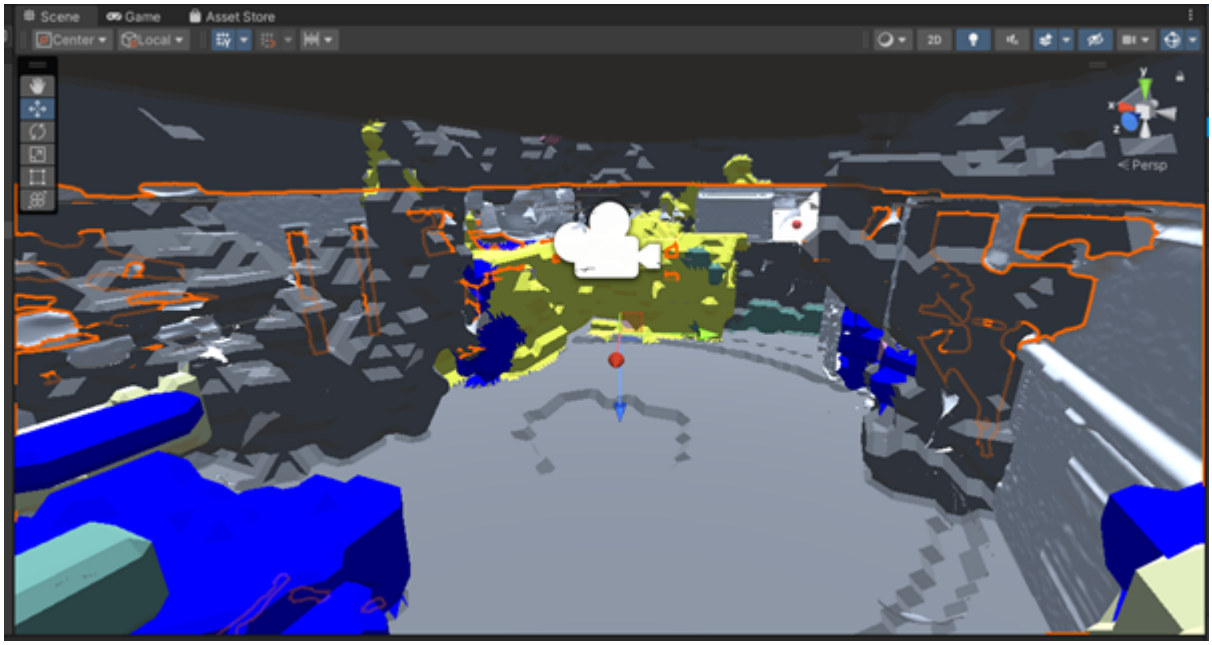


Figure 4.31: Editor view with selected superimposed LiDAR scan of the kitchen scene.

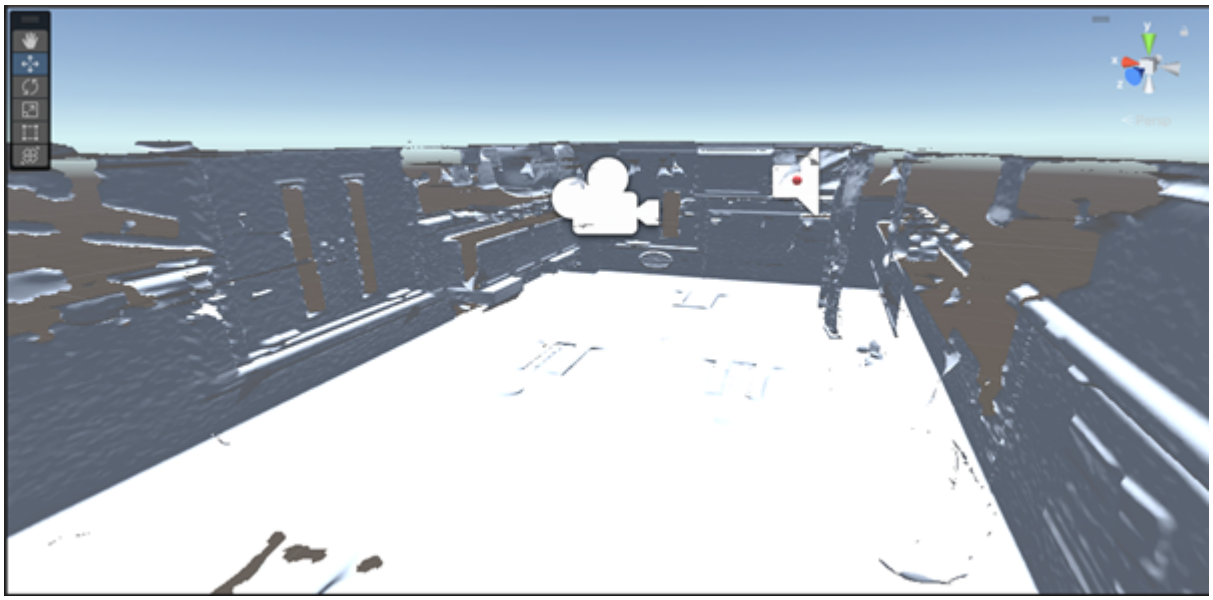


Figure 4.32: Editor view with the LiDAR scan, and the prediction mesh hidden.

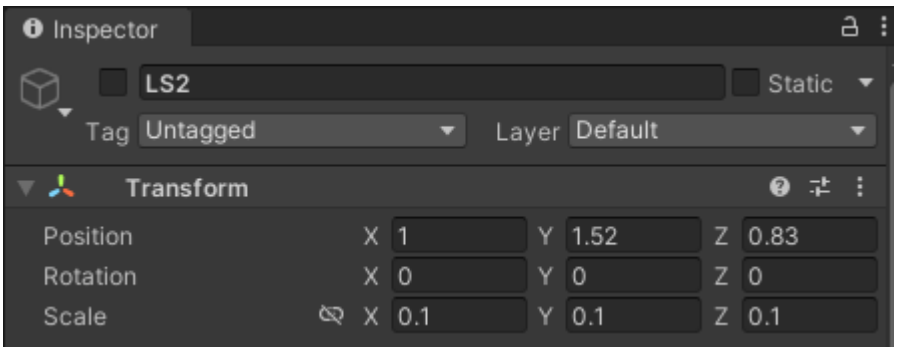


Figure 4.33: LS2 Audio Source's Unity coordinates.

4.4 Initial Results

With the coordinates for each object in the scene successfully converted, results were obtained by obtaining RIR values for recordings generated at the camera/microphone position for each of the sound source files at the specified locations.

The metrics calculated were the RT60 (indicating the scene's acoustics through reverberations) and the RT60 (indicating more early reverberations). For more detailed definitions of these metrics, please refer to section 2.5.2.

4.4.1 Waveforms

The following waveforms represent the audio recordings generated at this stage, the change from the sources' indicating the effect of reverberation.

LS1

Gunshot

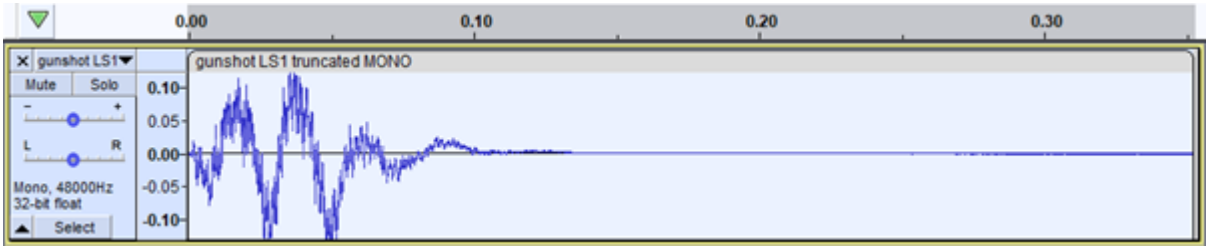


Figure 4.34: Linear waveform recording of the gunshot from LS1.

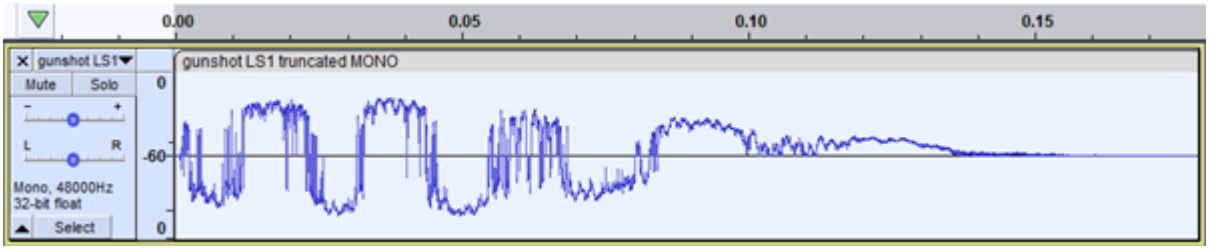


Figure 4.35: Decibel waveform recording of the gunshot from LS1.

Sweep

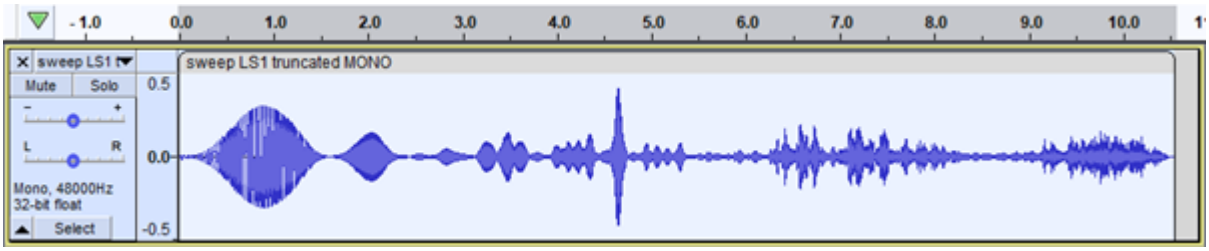


Figure 4.36: Linear waveform recording of the sweep from LS1.

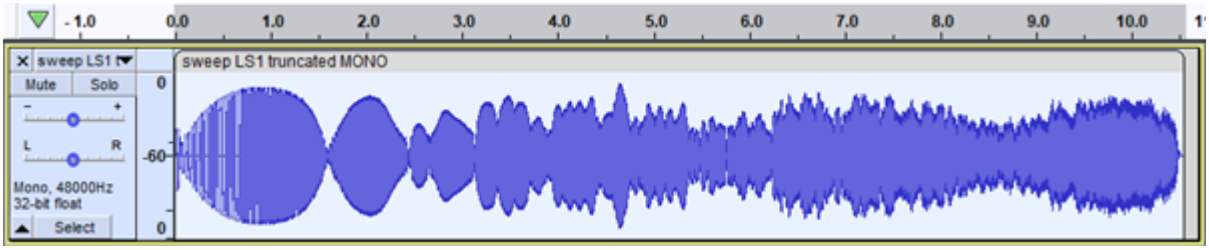


Figure 4.37: Decibel waveform recording of the sweep from LS1.

LS2

Gunshot

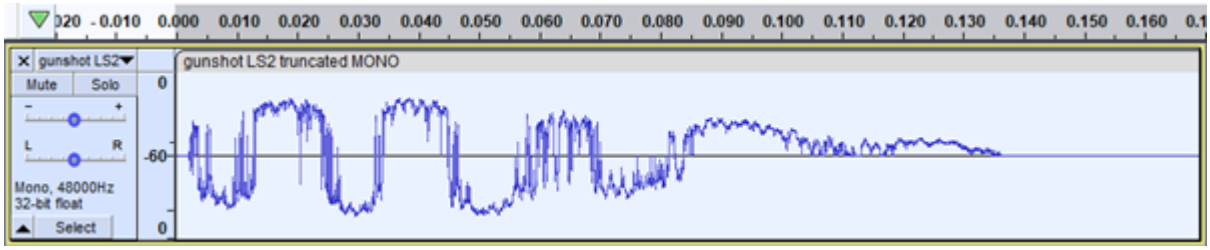


Figure 4.38: Linear waveform recording of the gunshot from LS2.

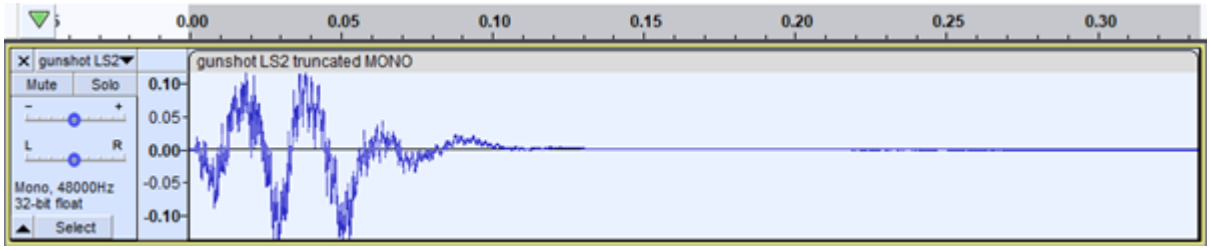


Figure 4.39: Decibel waveform recording of the gunshot from LS2.

Sweep

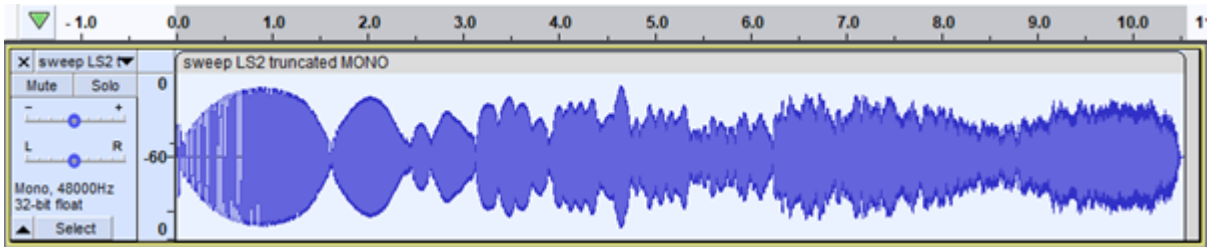


Figure 4.40: Linear waveform recording of the sweep from LS2.

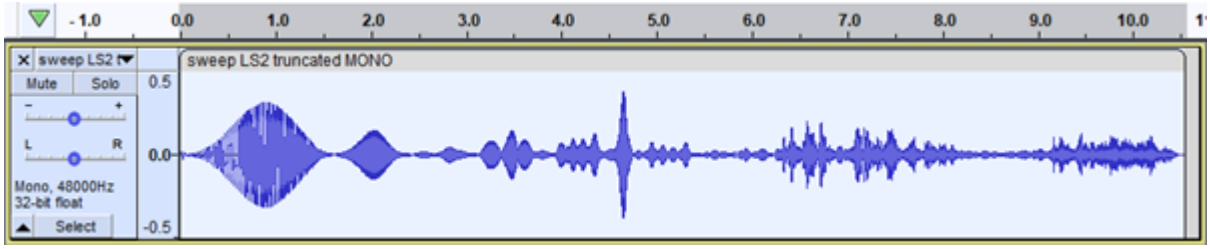


Figure 4.41: Decibel waveform recording of the sweep from LS2.

4.4.2 Evaluation

Results (for LS1) calculated by Dr Atiyeh Alinaghi:

- $RT60$ = 0.1179 seconds;
- EDT = 0.2167 seconds.

Qualitatively, reverberations are poor (notedly: the gunshot's impulse seems weak), and the results do not match actual recorded ground truth of kitchen scene: *RT60* = ~0.5 seconds; *EDT* = ~0.4 seconds.

4.5 Final Results

Changing Steam Audio settings to maximum performant reverberation calculations yielded a much better quality of results at this stage (figure 4.42).

Figure 4.42: Steam Audio Engine Settings dialog.

All parameters were increased (bar Irradiance) as far as would performantly run with the GPU, i.e.: until FPS stability began to drop. Raytracing and Reflection Effect settings were also changed from CPU to GPU-accelerated methods (Radeon Rays and TrueAudio Next, rather than Unity-provided raytracing and Convolution-calculated reflections, as was the default).

Additionally, the final results were calculated to evaluate our pipeline with multiple depth map configurations. Two of the scenes were generated from 360MonoDepth, with two differing images of the same scene (referred to as "shifted" and "non-shifted"). The other, referred to as "non-MonoDepth", uses the ground truth depth map. These two input images are displayed below for reference:

Figure 4.43: Shifted 360MonoDepth input image.

Figure 4.44: Non-shifted 360MonoDepth input image.

4.5.1 Waveforms

To demonstrate the increased efficacy of increasing the Steam Audio global parameters, the waveforms of the sound recordings generated for obtaining the results for non-Monodepth (ground truth) are displayed hence; note the increased length and variance as compared to the initial results, indicating greater reverberation with the improved settings.

LS1

Gunshot

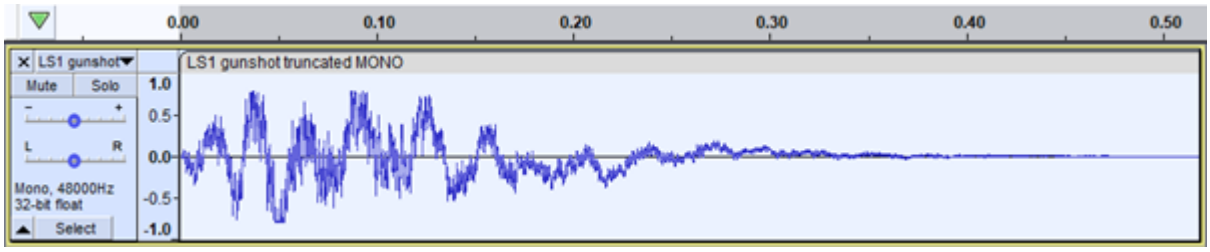


Figure 4.45: Linear waveform recording of the gunshot from LS1.

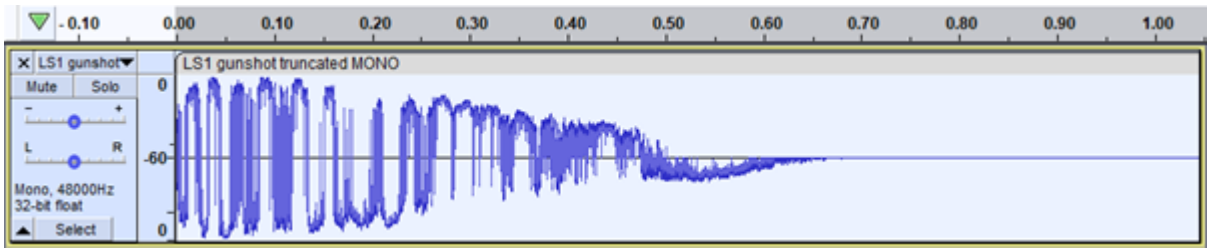


Figure 4.46: Decibel waveform recording of the gunshot from LS1.

Sweep

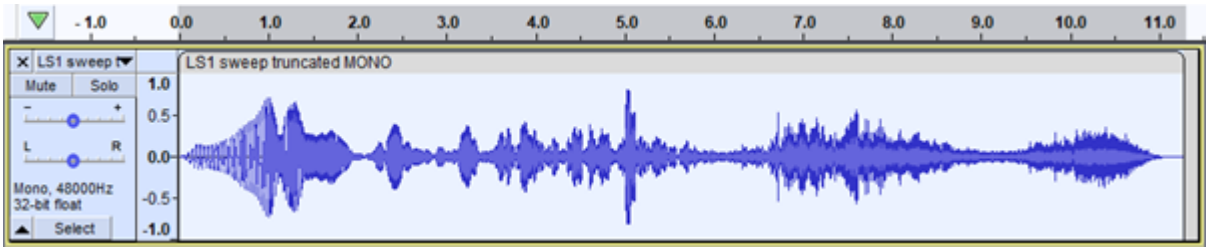


Figure 4.47: Linear waveform recording of the sweep from LS1.

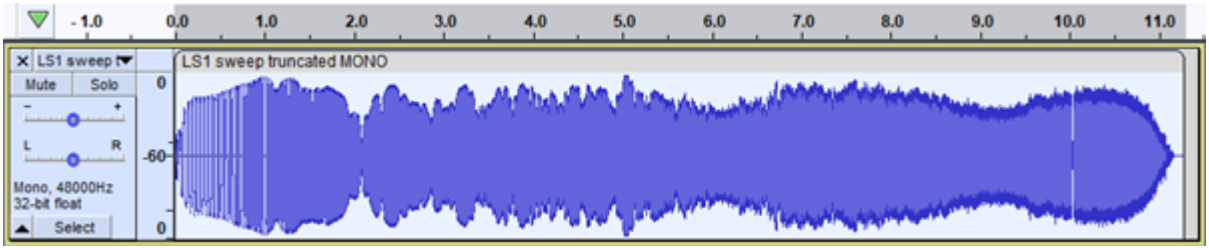


Figure 4.48: Decibel waveform recording of the sweep from LS1.

LS2

Gunshot

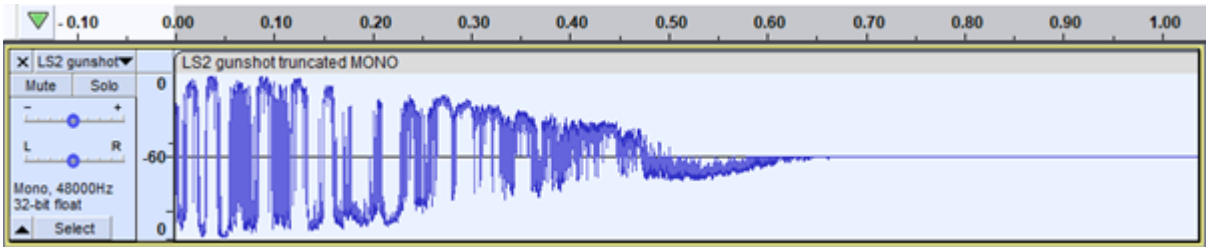


Figure 4.49: Linear waveform recording of the gunshot from LS2.

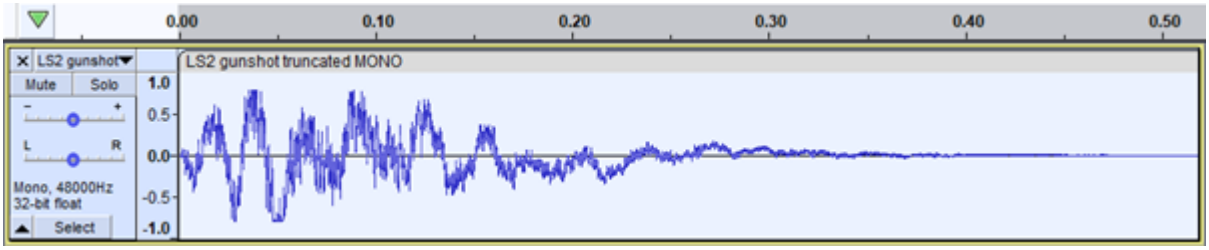


Figure 4.50: Decibel waveform recording of the gunshot from LS2.

Sweep

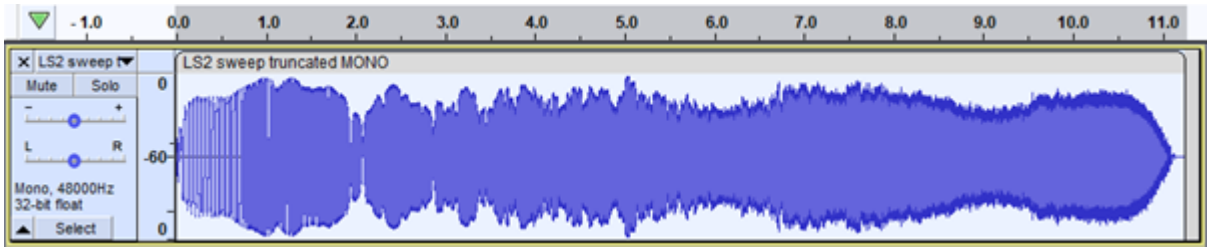


Figure 4.51: Linear waveform recording of the sweep from LS2.

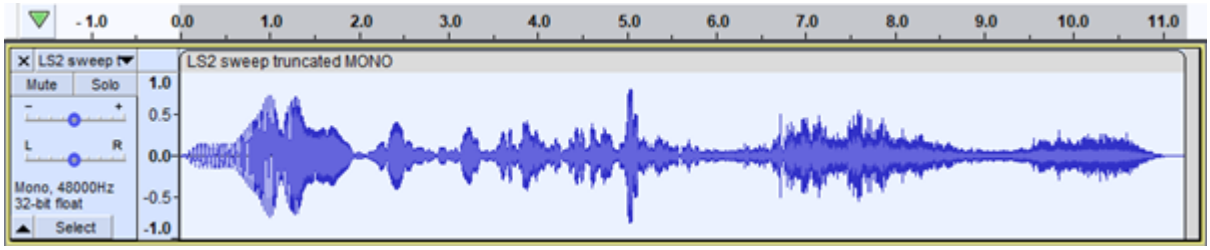


Figure 4.52: Decibel waveform recording of the sweep from LS2.

4.5.2 Evaluation

The final results are much better with increased Steam Audio compute settings. In addition, the RIRs obtained from the MonoDepth depth maps are of sufficient accuracy given their monocular estimations: the RT60 deviates somewhat from the ground truth depth map, but is within tolerance of the true value of 0.5 seconds, and the EDT is more accurate to the true value of 0.4 seconds than that of the ground truth depth map.

Again, data analysis was performed by Atiyeh Alinaghi:

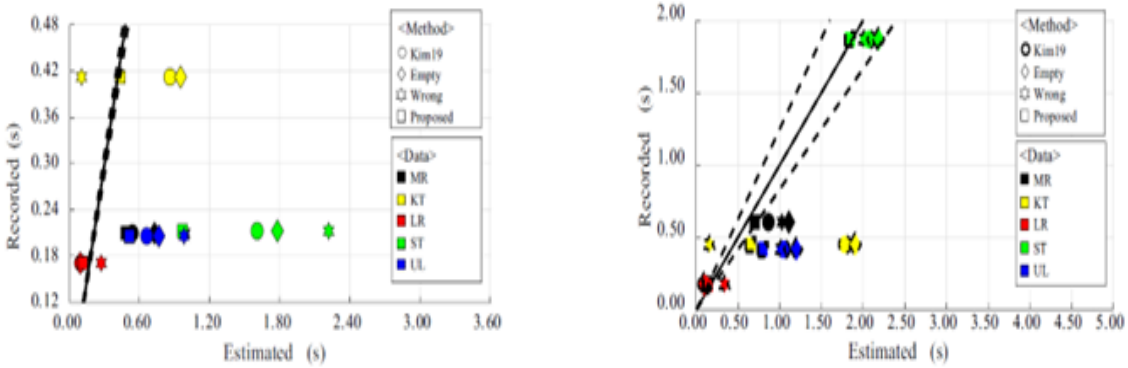
Position	RT60 (s)		
	MonoDepth (shifted)	MonoDepth (non-shifted)	non-MonoDepth
LS1	0.405	0.402	0.496
LS2	0.422	0.387	0.521

Table 4.3: RT60 values derived from final recordings.

Position	EDT (s)		
	MonoDepth (shifted)	MonoDepth (non-shifted)	non-MonoDepth
LS1	0.447	0.419	0.486
LS2	0.463	0.425	0.521

Table 4.4: EDT values derived from final recordings.

Qualitatively, the reverberations in the room were much more realistic with this setup (the gunshot impulse is properly heard reverberating through the room). This is close to the previously mentioned ground truth, and the results from Hansung's stereoscopic method ($RT60 = 0.46$; $EDT = 0.41$).



(a) EDT results in [3].

(b) RT60 results in [3].

Figure 4.53: EDT and RT60 results in Dr Hansung Kim's previous paper [3].

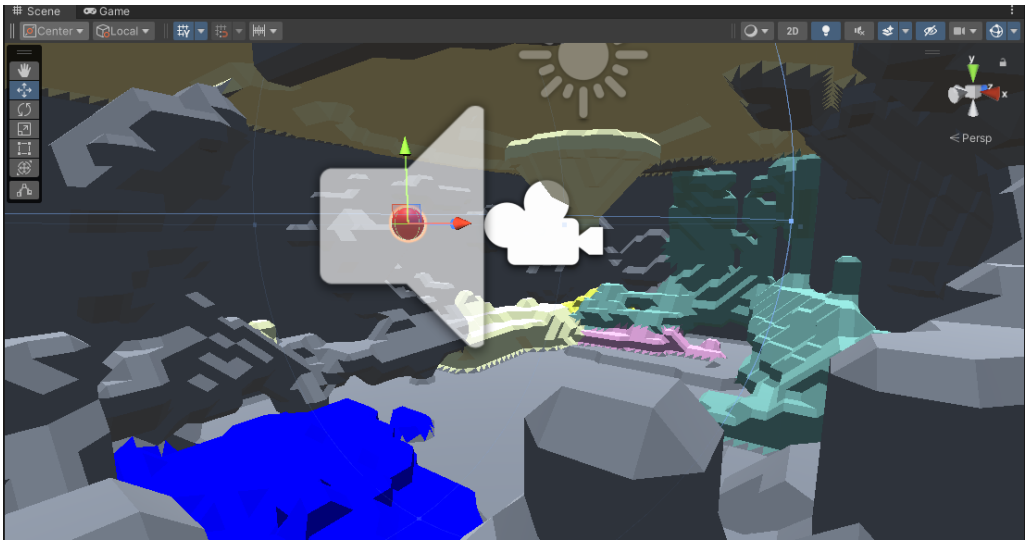


Figure 4.54: Editor view of 360MonoDepth (non-shifted) scene with LS1 Audio Source selected.

5 Research and Development

5.1 Introduction

Over the course of the semester, the team has continuously dedicated extensive efforts in Research and Development (R&D) to optimise pipeline performance. Various approaches were explored to enhance the efficiency of individual components and are explained in this section.

5.2 Monocular Depth Estimation

5.2.1 RWTD

Motivation

Reverse-Gradient Warming-up Threshold Discriminator (RWTD) [38] introduces a novel approach to the field of Monocular Depth Estimation of omnidirectional images. The primary objective of RWTD is to facilitate the generation of depth map predictions on a network trained exclusively on computer-generated datasets as opposed to real-world data and their ground truths.

Architecture Summary

A key component of RWTD is the incorporation of a discriminator. RWTD employs the concept of reverse-gradient descent, where the discriminator undergoes training with the aim of becoming incapable of distinguishing whether feature vectors originate from the source domain or the target domain. The adaptive weighting mechanism allocates varying weights to different scenes in the training dataset, allowing the knowledge learned from the source domain to be applied effectively in predicting depth maps for unlabelled scenes in the target domain.

In addition to its discriminative functionality, RWTD addresses a significant challenge observed in previous GAN-based (Generative Adversarial Network) domain adaptation methods. It happens to be the case that increasing domain label losses dominate the loss function, thereby pushing the architecture in an incorrect gradient direction. To counter this, RWTD introduces "warming up" thresholds, setting constraints on loss values throughout the training process. These thresholds are dynamically adjusted based on training epoch progression, ensuring optimal performance.

R&D Results

Despite the innovative aspects of RWTD, it poses challenges that make it unsuitable for the project. The absence of comprehensive instructions and guidance for implementation raises concerns about the feasibility of seamlessly incorporating RWTD into the project. Additionally, the lack of a pretrained model implies that it needs to be trained from scratch, putting strain on resources and time. Most importantly, direct advice from the author of RWTD indicates that it is not suitable for this task as it is mainly focused on "domain adaptation" and since the work is focused on google performance of a depth estimation model, in this context, it was recommended to look at something else that is more suitable and easier.

5.2.2 SliceFormer

Motivation

Through further discussions with the project supervisor, a new novel depth estimation network SliceFormer, currently not published for public use, was made available exclusively for the project, by the author of RWTD. This network demonstrates superior performance on real-world datasets, indicating its potential for depth estimation for indoor 360 scenes.

At the time of writing, the paper for SliceFormer is not publicly available.

Architecture Summary

The concept proposed in the paper suggests employing a slice-based transformer for estimating depth in single indoor 360 images. The strategy utilises gravitational features and divides the image features into slices aligned with the direction of gravity[39]. The model architecture consists of an encoder-decoder network and a slice-based transformer, thereby processing segmented features to produce a depth map. The method aims to address the challenges of indoor depth estimation, showcasing better performance on real-world datasets compared to existing depth estimation models.

The proposed architecture in the paper incorporates two key components: an encoder-decoder network and a slice-based transformer. Firstly the encoder-decoder network uses a U-Net-shaped structure with an EfficientNet backbone [39], capturing global image characteristics and reconstructing detailed information as it relates to the target objects. It utilises convolutions to segment the decoded encoder-decoder features into "slices" aligned with the gravitational direction. The second component, the slice-based transformer, processes the segmented features through patch embedding and a transformer architecture, leading to the generation of a depth map. This involves using a 100-dimensional vector representing depth bins and an attention map that is derived via convolution kernels to produce the final depth map[39].

R&D Results

SliceFormer is not designed to take in images of varying dimensions as an input so anything that isn't a 512x256 pixel image is rejected by the network. Notably,

this includes all the high-resolution images that this project is being built to incorporate. This restriction means that all inputs have to be rescaled to those exact dimensions using either imagemagick or ffmpeg and then passed in as input. The generated depth map, while surprisingly detailed is in most cases even smaller in resolution than the input image and as such the produced depth map lacks enough resolution for the detail in it to be meaningful for Semantic Scene Completion. This is highlighted in Figure 5.1

Figure 5.1: The low resolution but very detailed output for the DWRC image generated using SliceFormer

Another main complication here is that the depth map generated is not an image file, but rather an EXR file that contains the real depth values for a given scene. It is possible to convert this into an image file of the appropriate dimensions using code, but one must normalise the depth range when converting. Without normalisation, the converted outcome is not always consistent because the gamma range of a produced depth map here deviates from scene to scene. It was possible to address the primary problem of low-resolution input and outputs by retraining the network with higher-resolution images from the 3D60 dataset instead.

Figure 5.2: Output generated using retrained SliceFormer

Retraining it on a higher-resolution dataset produced depth maps of a significantly higher resolution. However, comparing the two results from the previous model and this retrained one proved to demonstrate a decline in quality in the overall detail detected in the image. The finer details from the depth map generated by the retrained network are missing, as can be observed in Figure 5.2. The details from the bookshelf detected on the right in the retrained model has far fewer details and appears as more of a flat surface.

5.2.3 BiFuse

Motivation

BiFuse [40] is one of the only open-source Monocular Depth Estimation networks tailored for omnidirectional images. The exploration of BiFuse stems from its distinctive methodology in addition to the promising results displayed on the main repository in addition to its ease of setup.

Architecture Summary

The paper for BiFuse introduces an end-to-end neural network with two branches designed for Monocular Depth Estimation. This involves using both cube maps and equirectangular projections and combining them through a bi-projection fusion approach that incorporates learnable masks for optimal balance of information[40]. Additionally, a spherical padding technique is suggested to expand the field of view of the cube map projection and diminish boundary inconsistencies on each face.

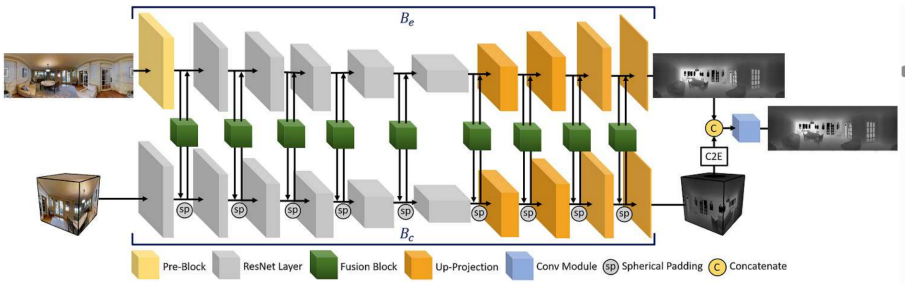


Figure 5.3: Diagram representing the architecture of the BiFuse model.

R&D Results

Despite the promising aspects, the pre-trained BiFuse model provided encounters challenges in achieving consistent results across any images outside of the scope of the test set. It shows superior performance on the provided test images but struggles with generalisation to other images. This is evident in Figure 5.4b

(a) BiFuse result on image provided with it

(b) BiFuse result on our image

Figure 5.4: A comparison of how well BiFuse performs on the image provided in its test set and how poorly it performs on an image outside its test set

5.2.4 UniFuse

Motivation

UniFuse (Unidirectional Fusion) [41] is a compelling alternative to 360MonoDepth for several reasons. The first reason is its hassle-free setup and the second is that generating a depth map with it is computationally inexpensive. In addition, it is evaluated on widely-used datasets of omnidirectional images, managing to achieve leading-edge results, particularly on the Matterport3D dataset.

Architecture Summary

UniFuse aims to achieve depth estimation by utilising both equirectangular projections (ERP) and cube map projections (CMP) for comprehensive depth estimation. The methodology stands out with its unidirectional fusion framework, which effectively merges features from both ERPs and CMPs, departing from the conventional bidirectional fusion approaches[41]. Within this framework, a specialised fusion model is introduced to enhance equirectangular features.

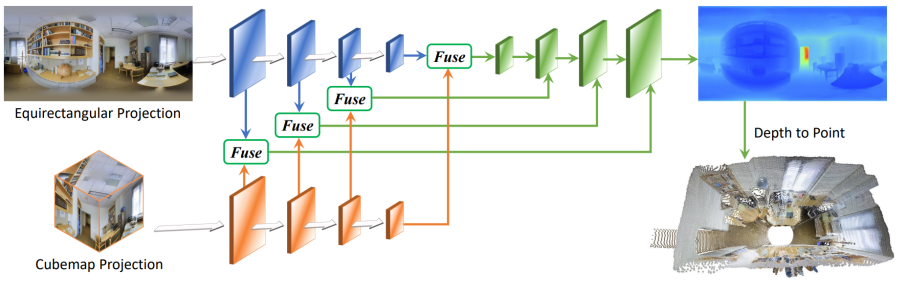
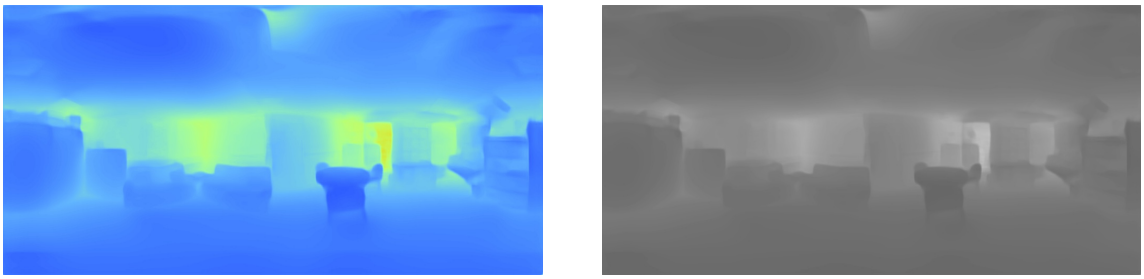


Figure 5.5: Diagram representing the architecture of the BiFuse model.

R&D Results

The pre-trained model consistently produces exceptionally high-quality and detailed depth maps. In fact, UniFuse takes the lead in providing the most detailed Monocular Depth Estimation so far, surpassing 360MonoDepth, managing to detect objects further away in the scene in great detail. This can be observed in figure 5.6a. Similar to 360MonoDepth, it employs a similar but slightly different "rainbow" colour scheme for its depth maps, once again producing a depth map that is represented on a nonlinear scale. Same as before, it is straightforward to convert this to a linear normalised greyscale depth map by defining a custom colour map, as shown in figure 5.6b.

Intrinsically, the depth maps generated by UniFuse are very promising, and the best and most detailed encountered in the course of this project. However, despite this intrinsic advantage, it falls surprisingly short when coupled with EdgeNet360 for SSC. Extrinsically, the 3D reconstruction using this normalised map is substantially worse and far more random than the normalised depth map produced using 360MonoDepth. This only further emphasises the complexity of the project, highlighting limitations in perhaps the technology of MDE itself or perhaps limitations in the way EdgeNet360 is built.



(a) Stock UniFuse result on the DWRC image.

(b) Normalised UniFuse result on the DWRC image.

Figure 5.6: The depth map outputted by UniFuse with the default rainbow colour map and its corresponding normalised greyscale counterpart.

5.3 Semantic Scene Completion

5.3.1 VoxFormer

Motivation

To advance the existing Semantic Scene Completion outputs from EdgeNet360, the team has chosen to retrain the VoxFormer model [42], originally trained by NVIDIA researchers for outdoor settings, on omnidirectional indoor scenes. With the absence of contemporary approaches for Semantic Scene Completion in indoor settings, the retraining of a state-of-the-art outdoor scene model thus stands as the most viable method to improve the EdgeNet360 results. VoxFormer's selection is underpinned by its innovative approach and proven performance in outdoor settings offering promise for indoor applications.

As indicated in its paper, VoxFormer's performance (validated on the SemanticKITTI dataset for outdoor scenarios [43]), with an Intersection over Union (IoU) of 65.38% for distances up to 12.8 meters, showcases its superiority over competing approaches such as MonoScene (IoU 38.42%) [44]. The advantages of this model are further evidenced in the paper, with it achieving occasional performance parity with LiDAR-based methods like SSCNet (IoU 64.37%) [45] and JS3CNet (IoU 63.47%) [46].

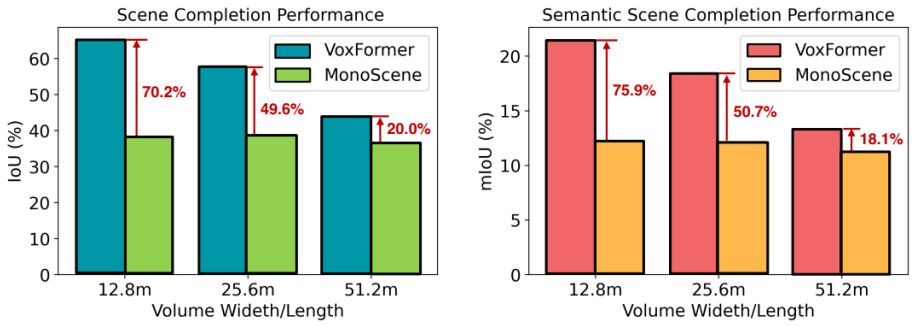


Figure 5.7: Bar charts comparing the Semantic Scene Completion performance between VoxFormer [42] and state-of-the-art MonoScene [44] on SemanticKITTI [43], reproduced from [42].

Architecture Summary

VoxFormer employs a two-stage design: the first stage proposes a sparse set of voxels based on depth estimation and correction, utilising ResNet-50 for 2D feature extraction and a Query Proposal Network for voxel queries. In the second stage, the Transformer Network refines voxel features and predicts semantics through a masked autoencoder-like architecture, incorporating deformable cross-attention and self-attention for enhanced scene completion.

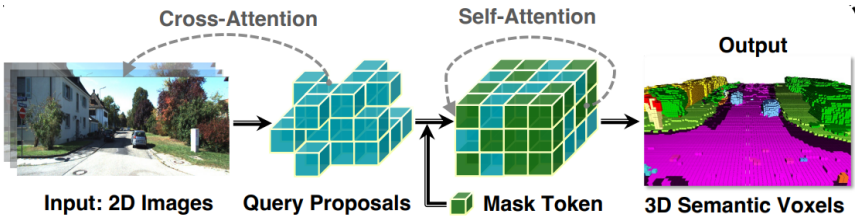
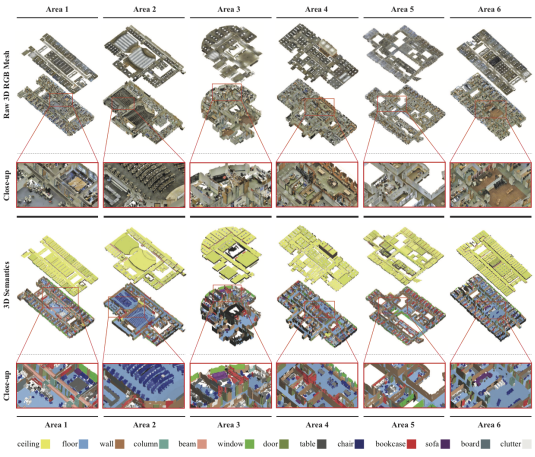


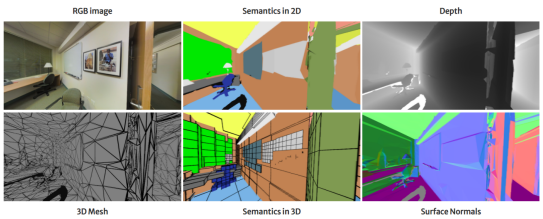
Figure 5.8: Diagram representing a high-level architecture diagram of VoxFormer, reproduced from [42].

Adaptation to Stanford2D3D

To train the VoxFormer network for omnidirectional images depicting indoor settings, the Stanford2D3D dataset [35] is examined, as it is a large omnidirectional dataset for indoor scenery. The Stanford2D3D dataset comprises six areas, covering a combined area of 6,000 m^2, as shown in figure 5.9b. The dataset includes the data required by VoxFormer, namely the omnidirectional RGB images, their corresponding depth maps, and 3D semantics. The semantic labels are: ceiling, floor, wall, column, beam, window, door, table, chair, bookcase, sofa, board, and clutter.



(a) The areas covered in the Stanford2D3D dataset, spanning over 6,000 m^2, reproduced from [35].



(b) The modalities in the Stanford2D3D dataset, adapted from [47].

Figure 5.9: The areas covered in the Stanford2D3D dataset and their corresponding modalities.

R&D Results

Following a conversation with Dr Kim, it was decided not to proceed with retraining VoxFormer with the Stanford2D3D dataset due to the following two factors:

1. The Stanford2D3D dataset's coverage area of 6,000 m^2 is not large enough to train a network with the VoxFormer architecture, which contains data-hungry transformer layers.
2. The format of the Stanford2D3D dataset is not compatible with the SemanticKITTI format required by VoxFormer, where each voxel is mapped to a file that indicates its properties (such as whether it is occluded or not in the RGB image). The next step would be to convert the Stanford2D3D dataset to the SemanticKITTI format, using techniques like ray tracing to determine whether a voxel is occluded or not.

Due to the low chances of success of this approach, Dr Kim strongly suggested efforts to be focused on an alternative solution.

5.3.2 Tweaking EdgeNet360

The results of the VoxFormer investigation and the lack of other state-of-the-art Semantic Scene Completion models for indoors led to the reconsideration of tweaking the existing EdgeNet360 architecture. EdgeNet360 takes in an omnidirectional image, splits it into eight views, and then feeds each view into an EdgeNet model for Semantic Scene Completion. The reconstructed models for each view are then assembled to create the 3D reconstruction of the omnidirectional image. Therefore, a new idea was to replace the EdgeNet Semantic Scene Completion model with a more performant model. However, no compatible state-of-the-art model was found for indoor settings.

5.4 Point Cloud Completion

With no possible modifications to the Semantic Scene Completion module, the next approach to improve its results would be through postprocessing; namely, improving the meshes generated from the voxels outputted from the SSC model.

5.4.1 Partial2Complete

Motivation

Partial2Complete (P2C) [48] is a self-supervised framework for Point Cloud Completion that takes an object's incomplete point cloud and completes it, which would help reduce the anomalies outputted by the Semantic Scene Completion model. The model learns completion from a category-specific dataset of partial objects, by exploiting structural cues and local surface continuity. The dataset used for training is ShapeNet, which includes all the object categories in Stanford2D3D, namely table, chair, bookcase, and sofa.

The paper demonstrates the effectiveness of P2C's novel techniques on synthetic and real-world datasets and shows that it achieves comparable or better results

than existing methods like PCN, Inv, and PPNet.

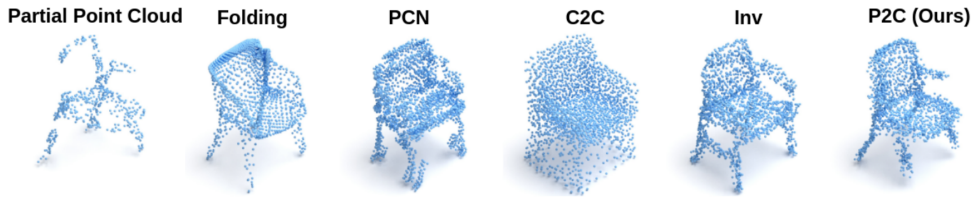


Figure 5.10: Diagram comparing the output of the Partial2Complete model with other state-of-the-art models.

Architecture Summary

The P2C network employs an encoder-decoder architecture to predict a complete point cloud from a partial input. Its core learning strategy, Patch-wise Self-supervised Learning, involves dividing the input point cloud into observable, masked, and latent patches. The network is trained to predict the complete point cloud, including both observable and latent patches, from the observable patches minus the masked ones. The architecture introduces the Region-Aware Chamfer Distance as a loss function and evaluation metric, improving upon traditional Chamfer Distance by evaluating point cloud correspondence based on local regions. Additionally, the Normal Consistency Constraint (NCC) is implemented to encourage the predicted shape to follow the local surface manifold of the complete point cloud by minimising the variance of normal direction similarity for nearby points.

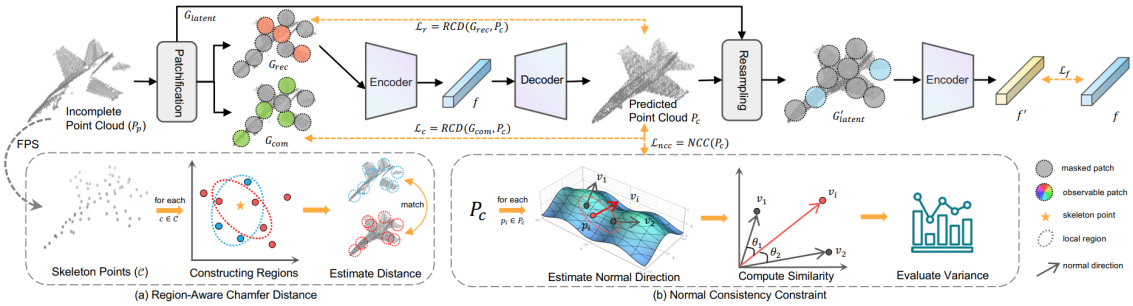


Figure 5.11: Caption

R&D Results

The network could not be implemented due to hardware limitations. Specifically, the team could not downgrade CUDA from 12.1 to the required 11.6 on the provided machine. Implementing alternative models, namely PCN [49], SVDFormer [50], and 3DQD [51] was unsuccessful due to the same reason. Point Cloud Completion was therefore dropped due to the lack of other functional alternatives.

6 Conclusion

6.1 Summary

The project builds upon Dr Kim's work [3] to tackle the problem of generating a VR scene with realistic visuals from a single omnidirectional image by employing a monocular depth estimation model (360MonoDepth [20]) to generate the depth map for the semantic scene completion model (EdgeNet360 [4]). To incorporate realistic acoustic properties, the pipeline uses a material recognition model [5] and then implements an algorithm to map the 2-dimensional material map to the 3-dimensional mesh to assign acoustic properties to each object in the VR scene. The 3D model with acoustic properties is then simulated in the Unity game engine.

Visually, while the pipeline does not achieve the precision of the stereo-matched ground truth depth map and exhibits notable visual noise in the form of displaced furniture, the overall shape of some objects is reasonably retained.

Acoustically, the VR scene realistically simulates audio for both stereo and monocular depth maps, achieving comparable results to previous stereo-dependent work with just a single RGB image.

6.2 Achievements

The project achieved several milestones, the main ones being:

1. The pipeline's backend functions autonomously, demanding no user intervention, automating formerly manual processes like material mapping. Through the GUI, the user inputs an image in three clicks to recreate the corresponding scene in VR. Previously, separate machines were utilised for distinct pipeline modules; now streamlined for seamless operation on a single machine.
2. The system successfully generates a 3D mesh from a single omnidirectional image using monocular depth estimation.
3. The system generates realistic audio for the reconstructed 3D scene in VR, attaining comparable results to those achieved with stereoscopic images in previous work.
4. The entire pipeline runs within 15 minutes, drastically reducing the time needed to recreate a 3D scene. This represents a noteworthy improvement, as the maximum duration previously required for the entire pipeline was 2 hours, reflecting a notable 92.5% reduction in running time.

6.3 Lessons Learnt

This project provided valuable experience and lessons to all team members, constituting a steep learning curve that presented distinct challenges. Confronting these challenges facilitated the development of diverse skills among the team. Key takeaways from the project include:

1. Research played a crucial role in depth estimation, semantic scene completion and material mapping, helping the team to evaluate diverse approaches and determine their suitability. This engagement in research significantly contributed to informed decision-making processes and enhanced our critical thinking skills.
2. Working on monocular depth estimation, semantic scene completion and audio testing involved collaboration and cooperation with PhDs. These instances provided us with academic research experience.
3. Working on a short-time-framed project underscored the importance of task prioritisation and resource allocation. Notably, the improvement of material recognition was deferred in favour of addressing a task characterised by higher complexity and uncertainty: material-to-mesh mapping. Additional resources were allocated to this prioritised task to ensure its successful completion.
4. Encountering issues related to dependency mismatches in Anaconda environments presented opportunities to build and refine troubleshooting skills.
5. Acquiring proficiency in Docker. Leveraging enhanced troubleshooting skills, the modification of the existing inefficient Docker setup for depth estimation was accomplished, resulting in a notable acceleration of the overall process.
6. Expanding our knowledge on how 3D reconstruction processes and computer graphics was broadened, contributing to an expanded knowledge base.
7. Gaining experience in CUDA C++ GPU acceleration programming.
8. Integrating software modules in a complex pipeline and automating it required us to communicate the interfacing requirements between each module. This experience contributed to the development of general scripting skills and effective technical communication.
9. For developing the audiovisual component of this project, it became necessary to become proficient in the fundamentals of Unity (including the C# programming language used in its scripting). Further knowledge was also required, and thus acquired, in more specialised areas such as: developing for VR with the OpenXR plugin; 3D modelling – working with Blender and .obj files; and audio processing fundamentals (as they relate to the simulation of spatialised sound).

7 Future Work

7.1 Monocular Depth Estimation

Several state-of-the-art monocular depth estimation models, including SliceFormer, BiFuse, 360MonoDepth, and UniFuse, were tried and tested to generate a suitable depth map for the semantic scene completion model EdgeNet360. Despite leveraging state-of-the-art models, the best-performing depth model (UniFuse) yielded suboptimal results when paired with EdgeNet360, in comparison to the ground truth EdgeNet360 output generated from the stereoscopic depth map. The challenge lies in discerning whether this discrepancy is attributed to the depth models utilised or inherent limitations within EdgeNet360. Investigating this issue stands as a potential avenue for future work. If the limitation is identified within monocular depth estimation, refining the normalisation process applied to 360MonoDepth becomes imperative to ensure its versatility across a spectrum of stereoscopic images. As such, any future work should focus on improving the output of the depth estimation model. To do this, the newly released DepthAnything [52] or BoostingMonocularDepth[53] would be good starting points. Alternatively, if the limitation is ascribed to EdgeNet360, future efforts should concentrate on adapting EdgeNet360 to integrate with the depth maps generated by 360MonoDepth seamlessly.

7.2 Semantic Scene Completion

A possible avenue to improve EdgeNet360's performance is to retrain the core EdgeNet model [54] with large omnidirectional datasets, like the Pano3D dataset [55]. SunCG [56] is another dataset that contains 45,000 computer-generated scenes that could be used in conjunction with Pano3D. This comprehensive retraining approach, incorporating both real and computer-generated high-quality indoor scenes, has the potential to improve EdgeNet360's performance by mitigating the risk of overfitting specific scene types.

Another alternative involves substituting EdgeNet's core model, EdgeNet, with a state-of-the-art model tailored for semantic scene completion in indoor environments. One possibility is CVSFormer [57], which needs to be trained on the Stanford2D3D dataset. Researchers in the future may also consider this approach if other novel models with superior capabilities are introduced to the field.

To tackle the anomalies (e.g., holes and deformities) in the voxel models generated by semantic scene completion models, a potential avenue for future work involves tailoring a generative model specifically for the pipeline's SSC model. This model

would take a 3D voxel model as input and output a completed voxel model without anomalies. The training data consists of the anomalous output generated by the SSC model in the pipeline as input, with the existing voxel models from the SSC model's dataset serving as the ground truth. The suggested semantic scene completion component is depicted in figure 7.1.

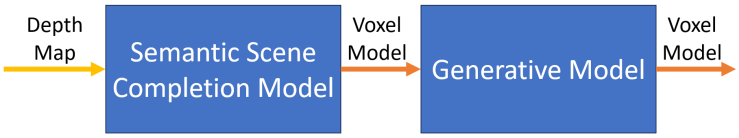


Figure 7.1: Diagram showing the suggested semantic scene completion component, which consists of the semantic scene completion model paired with a generative model that fixes the anomalies generated by the former.

7.3 Point Cloud Completion

While the team faced limitations in creating a proof of concept for point cloud completion due to CUDA limitations, a noteworthy suggestion for future work is to integrate a point cloud completion component between the semantic scene completion and the Unity integration components, as shown in figure 7.2. This placement intends to overcome the limitations of semantic scene completion, such as the anomalies in the generated voxels. In addition, semantic scene completion models are constrained by a predetermined granularity. Therefore, integrating a point cloud completion model would yield higher-quality 3D renders.

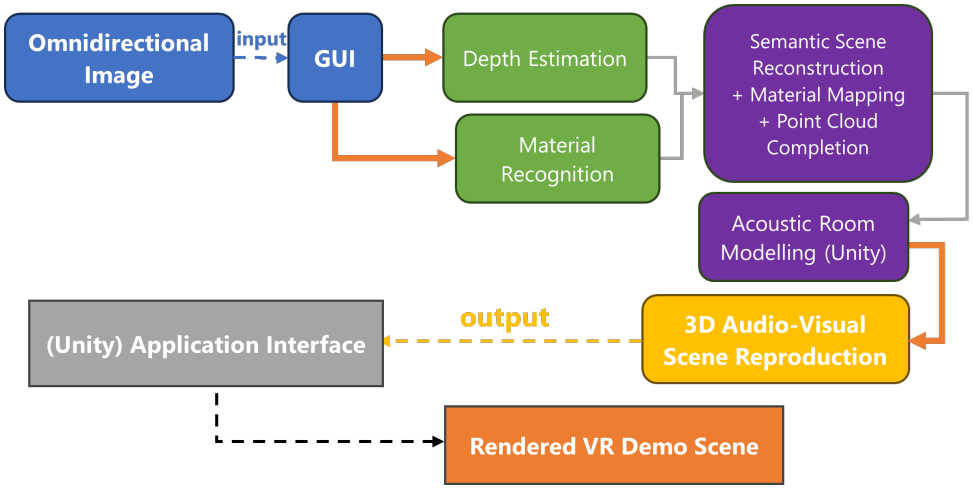


Figure 7.2: Diagram showing the new suggested pipeline, which includes point cloud completion.

The first model recommended for this purpose is the Partial2Complete model [48], which demonstrated promising results in its paper. The team would therefore recommend implementing this model on a CUDA 11.6-compatible machine. Additionally, another viable option for future exploration is a newly released model by NVIDIA [58], albeit its code was not yet publicly available during the project timeline; however, it is anticipated to be made available soon.

7.4 Material Recognition

The material recognition model, DBAT, performs well in recognising materials in omnidirectional images, being better at capturing the global context and larger objects than smaller objects and local materials on multiple occasions. Its effectiveness varies across different areas, particularly struggling with the top and bottom faces of the cube map projection. Due to the limited project duration and the complex and uncertain nature of the material-to-mesh mapping task, the project team accepted DBAT's performance as a limitation and prioritised the material-to-mesh mapping task. Therefore, the performance of the DBAT model could not be improved within the project's timeframe – leaving room for potential future enhancements, i.e., a technique to improve material recognition on the top and bottom faces/areas of the cube map projection by providing more context to those faces. An approach could be to copy the pixels of the neighbouring pixels of the top and bottom as they would be less likely to change. Another approach to improve DBAT's performance could be to adapt the architecture such that it can be trained on omnidirectional images for material recognition. However, an effective dataset would be required for this.

7.5 Material-to-Mesh Mapping

While the presented automatic mapping method is particularly effective and user-friendly when creating scenes in Unity, the previous manual mapping method provides a more accurate representation of scene materials due to the automatic mapping's current technology constraints rather than process defects. The accuracy of automatic material mapping is heavily reliant on the DBAT output. Additionally, the absence of additional contextual information, such as recognising shared materials across walls and ceilings, poses a challenge to the precision of automatic mapping. To bridge the gap between created VR scenes and real-world scenes, the semantic scene completion model's object identification and the material recognition model's material recognition performance both need to be significantly enhanced.

The object detection of EdgeNet360 failed on instances of identifying objects incorrectly, partially or mislabeling them. This severely affected the performance of the material mapping. EdgeNet360 uses a 3D CNN inspired by U-Net design for object detection in the SSC task. To improve object detection, attention mechanisms like self-attention can be incorporated within EdgeNet360. Attention mechanisms can help the model focus on important/informative features of the input data improving the object detection accuracy. This would involve modifying the architecture to include attention layers within the existing CNN layers.

Another approach could be to use Vision Transformers (ViTs), to either completely replace the 3D-CNN or a hybrid architecture combining the 3D-CNN and ViT can be used. The CNN could be initially used as a local feature extractor after which these features could be passed to the Vision Transformer. Through its attention mechanisms, it can capture global contexts/dependencies between the objects and the scene, hence leveraging the strengths of both. An alternative strategy involves exploring a different architecture for semantic scene completion. For instance, VoxFormer [42], as detailed in chapter 5, stands out as a potential candidate. In VoxFormer's case, datasets like Pano3D and SunCG would need to be converted to the

SemanticKITTI [43] format before initiating the training process for the VoxFormer model. Researchers can leverage this approach of modifying the dataset's format not only for VoxFormer but also for any state-of-the-art semantic scene completion model.

7.6 Acoustic Modelling

In the pursuit of generating more accurate reproductions of a room's acoustics, setting the material property parameters (e.g. absorption for different frequencies) would need to be more involved: our ad-hoc approach of referencing public data gave acceptable results in terms of RIR values, but further work would likely require experimentation to gain more broadly consistent values. Even working with Steam Audio itself for the audio component may need to be abandoned, as its functionality can be somewhat inscrutable; a novel implementation of a spatialised audio package may be required to gain more granular control over the acoustic modelling process.

Steam Audio's unsuitability for future work is also indicated by its performance limitations; as Valve's support of their publicly available projects is lacklustre [59], some GPU accelerated functions in the audio calculation are either absent or non-functional due to a lack of support for modern drivers and graphics cards.

7.7 Audiovisual Scene Reproduction

It is unfortunate that due to the limitations of Unity and the plugins chosen to complete this project, it was impossible to build this part of the pipeline into a standalone application without requiring the Unity Editor to run. Crucially, dynamic importation of assets from the file system (i.e., the 3D model generated by our pipeline) and changing the tagged Steam Audio Geometry of a scene must be performed in Unity's "Edit" mode; different, and perhaps more bespoke solutions to audiovisual rendering would thus need to be implemented in the pursuit of creating a more seamless pipeline from omnidirectional input to a VR scene.

For increasing the functionality of the audiovisual scene, multiple avenues could be explored in future work. Some examples would be mapping a textural image overlay from the pipeline's original input image over the scene mesh (which is currently merely coloured by the material of each sub-mesh); allowing for customisation of the audio within the scene at runtime (such as changing its behaviour, volume, adding multiple sources, etc.) through VR context menus; and perhaps allowing for "conferencing" with multiple users broadcasting their microphone input, to demonstrate better how the project might be useful for metaverse applications.

8 Project Management

8.1 Project Management Framework

At the beginning of the academic year, the team initiated cultivating a collaborative and cohesive work environment. Recognising the significance of establishing interpersonal connections, the team dedicated the first meeting to team building. This inaugural meeting served as a platform for open discourse, allowing each member to articulate their strengths, academic and work experiences and project aspirations. As discussions unfolded, roles within the team were deliberated upon, setting the stage for a harmonious distribution of responsibilities.

According to the Drexler-Sibbet Team Performance Model [60], the first week's meetings were key to going through the first four stages:

1. Orientation: each team member expressing their skills and interests.
2. Trust Building: building interpersonal relations.
3. Goal Clarification: discussing the specification.
4. Commitment: laying out an initial strategy.

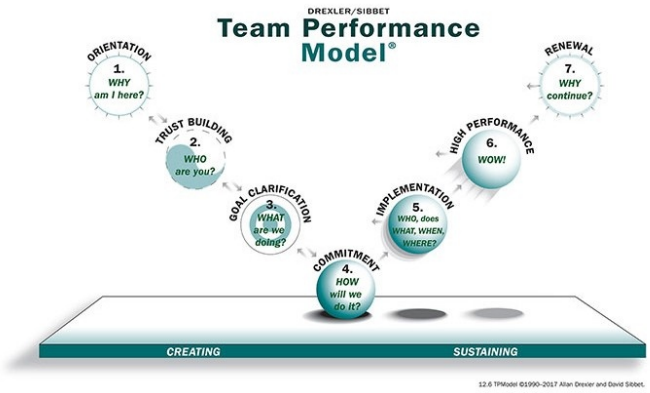


Figure 8.1: The Drexler-Sibbet Team Performance Model, reproduced from [60].

This strategic approach not only fostered a sense of unity and prepared the team for the remaining three stages (implementation, high performance, and renewal), but also laid the groundwork for the following McKinsey 7-S project management framework [61].

8.1.1 Strategy

Realising the large scale and the high technicity of the project, requiring the implementation of multiple state-of-the-art computer vision models as well as complex material mapping and audio-visual rendering algorithms, the team has collectively agreed on the importance of completing the core pipeline components as soon as possible with the Eisenhower matrix [62] in mind. The pipeline would be assembled at later stages, namely once two consecutive pipeline components are completed. For example, once the Monocular Depth Estimation and the Semantic Scene Completion components are marked as completed, the pipeline joining the two is built.

8.1.2 Structure

In accordance with the conversation about each team member's relevant experience, Joe was elected as the project manager having previous project management experience in a computer vision project in the industry. His responsibilities include project planning, team organisation, meeting coordination, communication management, risk management, stakeholder management, and deliverable completion insurance. Adam was assigned as the project officer, coordinating with members to produce minutes, reports, and documentation. All team members are developers.

8.1.3 Systems

The project follows the Agile development model [63] as it institutes iterative development — key to counter the high technical risks entailed by the complex pipeline and its technically demanding components — and ensures continuous integration and testing leading to faster delivery. Jira, a web-based project management platform, is used to keep track of each sprint's backlog and each member's progress.

Each sprint task is set with a description meeting the SMART goal format [64], with a strong emphasis on measurability, since the metrics will determine the completion status of that task. The SMART goal format ensures task clarity and atomicity, and the goal itself is tailored to each team member following the Locke goal-setting theory [65], thus maintaining engagement through challenge and commitment as well as fostering continuous improvement in project planning through feedback. In addition, the tasks are well-organised so that each team member is monotasking as much as possible to ensure full immersion in the task at hand, known as a state of flow [66].

8.1.4 Shared Values

The team of enthusiastic final-year students follows the motto "Move Fast and Break Things" [67], knowing the large importance of speedy innovation in this project and the potential failures along the way. With the awareness that the project is for a single semester, and to mitigate this approach's characteristic higher risk of errors, passion for innovation is observed with reason and realism using NASA's Technology Readiness Levels (TRL) [68]. For example, simpler algorithms inspired by principles and observations (TRL 1) could be implemented from scratch, while complex machine learning models could only be implemented if they meet TRL 6

(i.e., the system is validated in a simulated environment) as its implementation, training, and testing are more time-consuming.

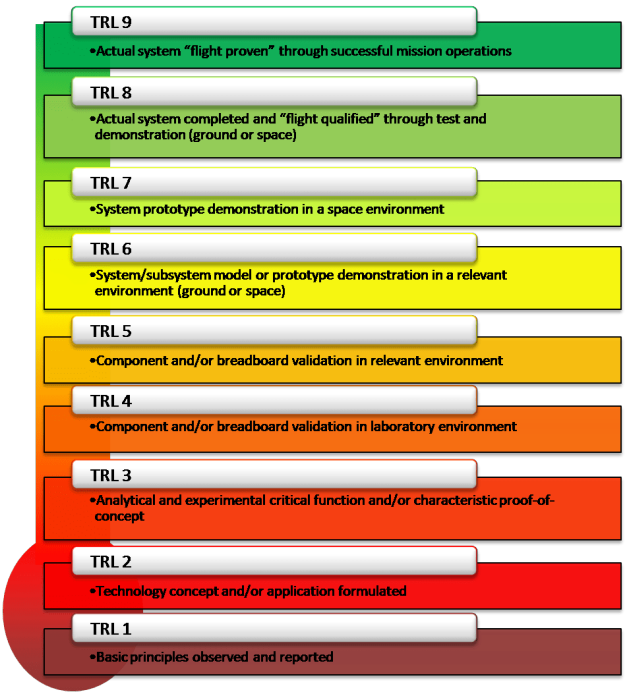


Figure 8.2: The NASA Technology Readiness Levels, reproduced from [68].

8.1.5 Style

The project manager holds the authority to enact changes and make decisions, thereby ensuring swift decisiveness and clear accountability. These decisions are informed by the team members' technical inputs through bi-daily sprint stand-ups, complemented by Dr Kim's weekly recommendations. The success of the "Move Fast and Break Things" approach with the mostly top-down management style is intricately tied to the conscientious application of Stephen Covey's seven habits of highly effective people [69] within the team. Sprint planning with the end in mind helped the team achieve quality and punctuality. Proactivity fostered innovation. Synergy yielded fast sprint deliveries. Regular stand-ups and sprint retrospectives allowed the team to sharpen its judgements and reinforce high expectations and a strong work ethic. End-of-sprint celebrations allowed the team to rest and bolster its cohesion. More on the leadership style will be discussed in the next section on sprint management.

8.1.6 Staff

The pipeline development was divided into three main specialisations:

1. Machine Learning Pipeline: the Monocular Depth Estimation, Semantic Scene Completion, and material recognition models. The tasks were allocated to Atharv, Joe, and Shubh according to their skills audit in Appendix B.1.

2. Material-to-Mesh Mapping Algorithm: mapping the materials from the material map outputted by the material recognition model onto the meshes out-

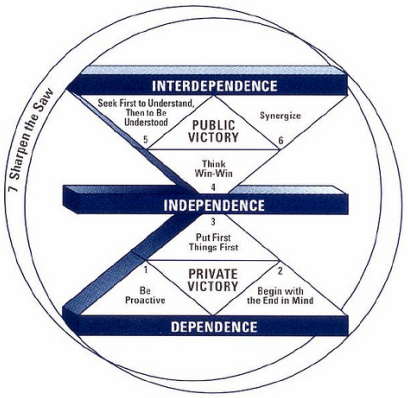


Figure 8.3: Diagram demonstrating the relationship between Covey's seven habits, adapted from [69].

putted by the Semantic Scene Completion model. This task was allocated to Sourish following his skills audit and his interest in exploring machine-learning technologies.

3. Unity Development: automatic scene generation from mesh data, acoustic modelling, user control. These tasks were allocated to Adam who is interested in Unity development.

8.1.7 Skills

Since the beginning of the project, the team had enough machine learning talent to work on the machine learning pipeline. On the other hand, the team did not have the full set of skills to work on material-to-mesh mapping and Unity development, with little support in these areas as previous work has manually performed these tasks. This meant that the team needed to dedicate a period of learning to acquire these necessary skills. The team's full skill audit is described in section 8.3.1.

8.2 Sprint Management

8.2.1 Sprint 1: Blitzkrieg

Scope

The first sprint is named "Blitzkrieg" as it aims to quickly develop critical project components. Its milestones are:

1. Machine Learning Components: Deliver a working prototype for the machine learning pipeline components, namely Monocular Depth Estimation, Semantic Scene Completion, and material recognition.

2. Unity Integration: Gain a comprehensive understanding of integrating machine learning pipeline outputs into Unity.

Management Style

The machine learning tasks were atomically divided and distributed among three team members. Since they consist of implementing existing machine learning models from online repositories, their corresponding deadlines were strictly observed to ensure the punctual delivery of the prototypes by the end of the sprint.

The Unity learning task had a coarser description (e.g., learning about scripting, system requirements, and input formats). Being a learning task, the set deadlines were loosely observed to allow team members to take time to learn and understand concepts, as well as explore alternatives.

Retrospective

At the end of the first sprint, the team has successfully achieved its goal of implementing primitive machine learning components and has captured a better idea of the integration into Unity. For the machine learning pipeline, the team has discerned the inefficacy of the RWTD model for Monocular Depth Estimation and started implementing the SliceFormer model. Also, the team now understands the next steps needed to make material recognition work for omnidirectional images. Finally, to make future sprints more manageable, each task will be accompanied by a description of its process and acceptance criteria ("If you can't measure it, you can't manage it." — Peter Drucker).

8.2.2 Sprint 2: Eureka

Scope

The second sprint, named "Eureka", aims to advance critical pipeline components through research and development (R&D) and the implementation of Unity modules. Its milestones are:

1. Machine Learning: Improve Machine Learning pipeline prototypes by exploring and implementing state-of-the-art techniques.
2. Material-to-Mesh Mapping: Research approaches to map materials onto the meshes.
3. Unity Locomotion: Achieve a working prototype of Unity locomotion.

Management Style

For this sprint, the tasks are coarsely set, as it is a research-based sprint. For example, since there is no improvement step after implementing EdgeNet360, the next task to attempt to improve Semantic Scene Completion is researching and implementing other open-source models. However, other research tasks, like developing a cube mapping algorithm to convert the omnidirectional image to a usable format for the material recognition model, are more specific as they are direct improvements on top of the first sprint's prototype.

For this sprint, deadlines were loosely enforced knowing that research is a continuous process whose duration could not be determined in advance. In addition, looser

deadlines fostered an innovative environment where creative approaches were proposed to solve two of the problems at hand, namely for Semantic Scene Completion and material-to-mesh mapping.

Retrospective

The team ended the two-week sprint with plenty of ideas for improvement, but little tangible changes to the prototypes apart from the obvious next iterations. It therefore announces the third sprint's executional nature with high expectations.

8.2.3 Sprint 3: We Conquer

Scope

In the mid-term presentation, one of the final slides showed the remaining challenges ahead. Its title was "We came, we saw…". The following three-week sprint's name completes this famous quote by Julius Caesar with "We Conquer", as the team is determined to solve the technical challenges on the component level. The sprint's milestones are:

1. Complete Material-to-Mesh Mapping.
2. Establish a Working Semantic Scene Completion Pipeline.
3. Develop Unity Modules: Create functional Unity modules that feature user locomotion and spatial audio.

Management Style

This sprint is managed with high expectations to deliver the new components while adhering to the set internal deadlines. To facilitate the management of this executional sprint, each task is atomically divided and is accompanied by a highly detailed description and acceptance criteria that include testing.

Retrospective

With all the machine-learning pipeline components finished and a single Unity module left to develop, the team is now ready to finish the product.

8.2.4 Sprint 4: Master Pipeline

Scope

The final sprint "Master Pipeline" aims to package to product and make it ready for delivery. Its milestones are:

1. Build Pipeline: Integrate all components in a single pipeline that takes in a single omnidirectional image and outputs a Unity VR scene.
2. Fine-Tune Machine Learning Models.
3. Complete Unity Development.

Management Style

As this stage is crucial in the delivery of the project, this sprint is managed with high expectations at every checkpoint. Every feature is required to be backed up by testing, and any new development must be supported by a backup plan. For example, fine-tuning a machine learning model does not require any backup plan as the earlier model version is already implemented in the pipeline. However, the development of the sound baking feature in Unity which improves the overall audio experience has an internal deadline where, in case sound baking is not fully functional, dynamic audio is implemented instead as it requires little development time.

Final Retrospective

The team is satisfied with the developed product and is ready for its delivery.

8.3 Strategic Management

8.3.1 Skills Audit

Across the three previously stated pipeline specialisations, i.e., machine learning, material-to-mesh mapping, and Unity development, three types of skills are identified: theoretical knowledge, practical knowledge, and soft skills.

For theoretical knowledge, the team started with strong foundations in machine learning, benefiting from the previous years' machine learning modules. In addition, some team members had foundational but sufficient theoretical backgrounds relevant to the material-to-mesh mapping and the Unity development, which progressively improved through continuous learning.

For practical knowledge, many team members have already gathered experience in machine learning development through coursework in previous modules, individual projects, and industry exposure. On the other hand, at the beginning of the project, the technique to be used for material-to-mesh mapping was unknown, and the team lacked significant Unity practical experience. These two practical skills were learnt throughout the semester.

For the soft skills, this project requires research, report writing, presentation, and basic graphic design. All team members were proficient in these skills since the project's onset.

Table 8.1 represents the team's skills audit at the beginning of the project. Table B.1 in the appendix contains the individual skills audit of all team members at the beginning of the project.

8.3.2 Risk Management

Risk Assessment

Risk assessment played a crucial role in determining the necessary completion speed for tasks during sprint planning to counter potential difficulties. Namely, recognising the high technical demands of implementing pipeline components and anticipating

Skills	Rating (1-5)
Theoretical Knowledge	
Machine Learning	5
Mathematics	5
Game Engines	2
Technical Skills	
PyTorch	5
GitHub Codebases	4
Open3D	2
Unity	1
Soft Skills	
Research	5
Report Writing	5
Presentation	5
Graphic Design	4

Table 8.1: Table showing the skills audit done at the beginning of the project, with skill ratings from 1 to 5 where 5 is the skill needed to achieve the project's goals.

other potential development issues, the team strategically allocated buffer times to each pipeline component. By completing a specific component earlier, the overall buffer time increases, providing the team with more time to address future challenges, thus reducing the risk of completing the entire pipeline. The table showing the list of risks, their likelihood, their impact, and their corresponding mitigation strategies are presented in table B.1 in the appendix.

Risk Mitigation

The initial challenge arose with the VR headset's incompatibility with the provided laptop. Prompt communication with the ECS Buyer team resulted in the acquisition of the necessary part within a week. Given that this issue was detected in the project's early stages (late October), the one-week delay had minimal impact on overall development progress. However, a subsequent connectivity-related issue emerged in late November due to a fabrication defect in the cable linking the headset to the laptop, which caused a three-week wait for a replacement under the warranty. Despite the three-week delay in obtaining a new cable from HP support, which temporarily hindered VR development, the team wisely utilised this time for report writing. Fortunately, the team's buffer time was enough to cover this delay.

The second significant challenge involved the technical difficulties encountered in setting up the Anaconda environments for many of the repositories, slowing down the development. In addition, the implementation of the material-to-mesh mapping algorithm lasted for more than a month due to its high difficulty. The swift completion of the material recognition task allowed for an extension of buffer time and the allocation of additional manpower to address this challenge, ultimately leading to a successful completion.

Finally, many team members have faced illness throughout the semester, freezing their work for up to two weeks, which has also pushed the expected delivery dates of some tasks. Illness was mitigated with task redistribution when possible, and

eventually an extension (further detail in section 8.4.5).

8.3.3 Time Management

The project timeline is estimated based on the skills audit and assessed risks. The Gantt chart (figure B.2) shows the project's main tasks, their dependencies, and expected times. Because of the events discussed in the previous section, the actual Gantt chart is different from the expected one, as shown in figure B.3.

On a smaller scale, Jira was used to manage task completion throughout the project and helped generate sprint reports that gave insight into the progress (Figure 8.4).

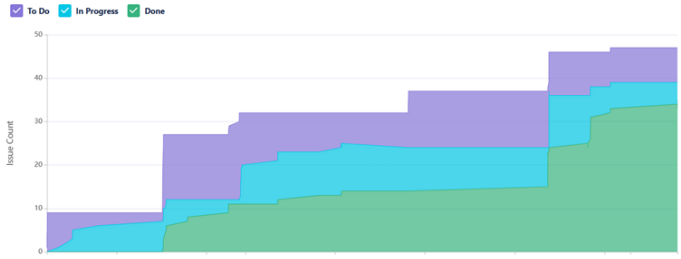


Figure 8.4: Cumulative flow diagram showing the tasks done, in progress, and to do at the end of sprint 3, generated by Jira.

8.3.4 Resources Management

The team was provided with a high-performance laptop (Dell Inspiron 16 Plus with an NVIDIA GeForce RTX 4060 Graphics Card) and a Virtual Reality headset (HP Reverb G2). These resources are shared among team members throughout the development. Therefore, to organise the allocation of these shared resources, a shared calendar was created within the team to book time slots for each piece of equipment when needed (Figure 8.5). Once the time slot ends, the equipment is returned to the locker in Building 16, thus ensuring equipment accessibility and safety.

In addition, since an adapter was required to connect the VR headset to the provided laptop, the university procurement team was contacted to purchase the required part. Purchase details are provided in Appendix B.2.

8.3.5 Productivity Management

This project uses GitHub for version control. The code is also backed up on OneDrive as part of the risk mitigation strategies presented in table B.1. LaTeX is used for report writing, and Microsoft PowerPoint for presentations and poster creation.

8.4 Teamwork

With the project being multi-faceted with several different modules interfacing with each other, collaboration between team members was necessary for the project to succeed. There were several instances where different team members worked on

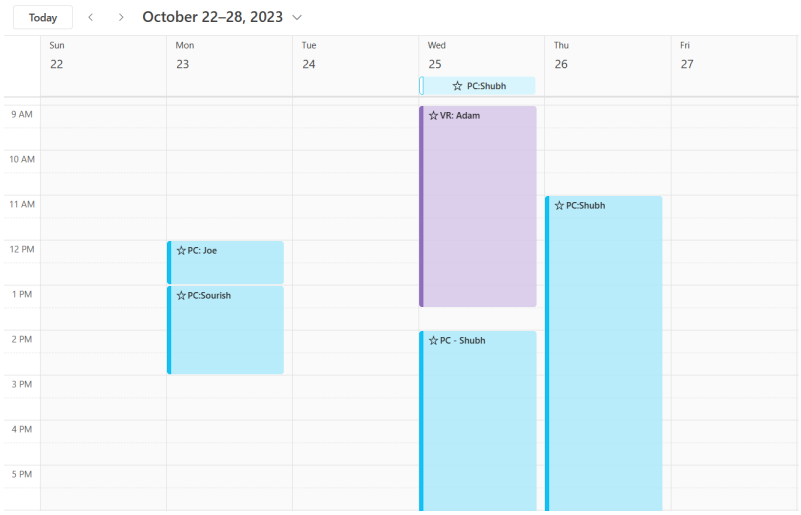


Figure 8.5: Calendar to book equipment time slots.

and across different modules to facilitate the interfacing and progress of the project. This section provides instances of teamwork and collaboration.

8.4.1 Agile Development

Agile methodology serves as the backbone of our project, realised through a steadfast commitment to the values outlined in the Agile Manifesto [70]. This section showcases how the team embraced and embodied these core values throughout the project lifecycle. The subsequent narratives delve into specific instances where these Agile values shaped our teamwork.

Individuals and Interactions over Processes and Tools

The team's human-centric approach is evident in our thrice-weekly meetings, acting as a skeleton throughout the project to provide direction. These sessions facilitated effective communication, enabling updates on each team member's progress, addressing impediments, and planning future tasks. This iterative process allowed us to allocate resources strategically, prioritising areas of complexity and ensuring a proactive approach to task assignment before the subsequent meeting or sprint.

Working Software over Comprehensive Documentation

Throughout the project, the focus was to deliver a working product. Rigorous research and development were integral components of our approach, as we continually assessed the viability of existing models and explored opportunities for enhancement. However, recognising the importance of timely delivery, certain aspects of research and development were strategically discontinued when deemed unfeasible. This decision-making approach was crucial in streamlining efforts and ensuring the successful delivery of a working product within our project timeline.

Customer Collaboration over Contract Negotiation

Our Agile approach prioritised customer collaboration through weekly meetings with Dr Kim who represents the customer. These sessions provided comprehensive up-

dates on project progress, impediments, and upcoming work, fostering transparent communication. Dr Kim's guidance played a crucial role in steering the project towards aligning with the customer's vision, exemplified by his input in incorporating features like smooth locomotion in Unity. This collaborative feedback loop ensured the final product met customer expectations. The meetings not only upheld Agile values but also aligned with Scrum pillars of transparency and regular inspection, providing continuous insights into our progress and facilitating necessary adjustments to achieve project goals.

Responding to Change over Following a Plan

While a plan was established at the project's onset, it was intentionally flexible, anticipating the need for adaptation as challenges arose. This flexibility was particularly crucial when unforeseen issues, such as a defect in the VR headset, occurred. Dr Kim's insightful feedback played a pivotal role in assessing the necessary adjustments to meet product expectations. An example of this adaptability was encountered when the initially provided depth estimation model, RWTD, failed to integrate with EdgeNet360, prompting the decision to overhaul the entire depth estimation model.

8.4.2 Collaboration

Incorporating Agile principles into our project, pair programming emerges as a vital collaboration tool. The instances in this section document the seamless integration of skills and agile problem-solving within the team.

- The task of material mapping was initially solely assigned to Sourish, but the team recognised the complexity and uncertainty of the module and assigned Shubh to the task as well. As anticipated, this task took a long time. However, the collaboration between the team members ensured that this complex task was completed.
- The task of importing the mesh into Unity called for Adam and Sourish to collaborate to ensure their modules could be interfaced. With Adam's knowledge of Unity and Sourish's Knowledge from working on the material mapping, relevant scripting and techniques ensured that the mesh was successfully imported.
- To convert 360MonoDepth's relative depth map into an absolute one, Joe supported Atharv in creating a script that defines the depth range thanks to his previous experience in depth maps.
- Adam collaborated with Atharv to get the depth estimation model running on the provided machine as there were several issues encountered with Docker. The joint effort of the team members helped overcome the dependency roadblock.
- To automate the whole pipeline Adam, Shubh, and Atharv wrote scripts for Unity automation, machine learning pipeline automation, and depth estimation automation. The members collaborated to combine these scripts effectively.

8.4.3 Communication

The role of project manager Joe proved instrumental in orchestrating effective communication. Serving as a central figure, Joe facilitated team-wide communication and actively engaged with team members as necessary. The communication framework is supported by three weekly meetings and a dedicated Discord server, allowing team members to reach out at any time.

8.4.4 Knowledge Transfer

Knowledge Transfer in Teamwork: Due to the technical landscape of the project pipeline, and since each team member is focused on their assigned component, the team understood the critical need for knowledge transfer. Therefore, during stand-up meetings, team collaboration extended beyond task updates to include in-depth explanations of relevant papers, general code discussions, and live demonstrations. This approach not only facilitated a shared understanding of individual contributions but also promoted a holistic view of the project.

Additionally, to foster continuous learning, each team member documented their work at the end of every sprint, creating a valuable reference for the team. This systematic knowledge transfer strategy became instrumental in enhancing overall team cohesion and project comprehension.

8.4.5 Crisis Management

Two weeks before submission, the project leader was severely affected by illness, impeding his contribution to the report and limiting his involvement to supervising the progress of other team members. The team has demonstrated strong resilience by following the mitigation strategy discussed in the risk assessment table (table B.1) and redistributed tasks to optimise the outcome of the final two weeks within the constrained manpower.

On the day of submission, the team secured an extension for the report until February 7 2024. Recognising the time constraints posed by other academic commitments, namely the individual reports and the final exams, the team decided to focus on the report after the GDP presentation. During this period, the team collaboratively refined the report's structure to align it with the intended vision, a vision that had been compromised earlier due to task prioritisation. Despite these challenges, the team expresses satisfaction with the final GDP report, underscoring the efficacy of crisis management, project planning strategies, and the cohesive teamwork exhibited throughout the project.

8.5 Reflection

The adopted project management approach significantly contributed to the project's success. The discipline instituted through Agile development, supported by the discussed 7-S framework, allowed the successful completion of all the client's requirements, despite three out of the seven identified events in table B.1 materialised.

Reflecting on the team dynamics throughout the project, it is evident that collaborative efforts were crucial to success. The team consistently demonstrated a high level of collaboration, fostering open communication and idea sharing. Development speed and punctuality emerged as a commendable trait within the team, ensuring that tasks and milestones were met in a timely manner. The team's resilience in overcoming unexpected challenges and crises demonstrates a collective determination to achieve project goals. Positive stakeholder communications were maintained, underscoring commitment to excellence.

8.6 Recommendations

This project's takeaways can be condensed for future endeavours in a couple of recommendations:

1. Examine equipment once received. Early defect detection can save precious weeks of work later in the project.
2. Once the general project principles are understood, build buffer time by implementing modules early on. Future improvements are built on top of the initial findings.

Bibliography

[1] G. Moore, "The effect of sound on the user experience of playing a video game," *Abgerufen von https://web. wpi. edu/Pubs/E-project/Available/E-project-042513-120731/unrestricted/Moore_MQP_Final. pdf*, 2013.

[2] Z. Cao, E. Magalhães, and G. Bernardes, "Sound design impacts user experience and attention in serious game," in *Serious Games* (M. Haahr, A. Rojas-Salazar, and S. Göbel, eds.), (Cham), pp. 95–110, Springer Nature Switzerland, 2023.

[3] H. Kim, L. Remaggi, A. Dourado, T. d. Campos, P. J. B. Jackson, and A. Hilton, "Immersive audio-visual scene reproduction using semantic scene reconstruction from 360 cameras," *Virtual Reality*, vol. 26, pp. 823–838, Sep 2022.

[4] A. Dourado, H. Kim, T. E. de Campos, and A. Hilton, "Semantic scene completion from a single 360-degree image and depth map," in 15^{th} *International Conference on Computer Vision Theory and Applications (VISAPP) - part of VISIGRAPP*, vol. 5: VISAPP, pp. 36–46, February 27-29 2020.

[5] Y. Heng, S. Dasmahapatra, and H. Kim, "Dbat: Dynamic backward attention transformer for material segmentation with cross-resolution patches," 2023.

[6] "Cylindrical projection."

[7] J. P. Snyder, *Flattening the earth: Two Thousand Years of map projections*. University of Chicago Press, 1998.

[8] G. Amoruso, *Handbook of Research on Visual Computing and Emerging Geometrical Design tools*. Information Science Reference, 2016.

[9] I. Hussain and O.-J. Kwon, "Evaluation of 360° image projection formats; comparing format conversion distortion using objective quality metrics," *Journal of Imaging*, vol. 7, no. 8, 2021.

[10] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," 2016.

[11] W. T, K. M, J. H, F. M, L. JJ, and M. J., "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *The International Journal of Robotics Research. 2015;34(4-5):598-626. doi:10*, p. 1177/0278364914551008, 2015.

[12] Google, "Resonance Audio - — resonance-audio.github.io." `https://resonance-audio.github.io/resonance-audio/`. [Accessed 07-02-2024].

[13] Valve, "GitHub - ValveSoftware/steam-audio: Steam Audio — github.com." https://github.com/ValveSoftware/steam-audio. [Accessed 05-02-2024].

[14] N. A. AG, "Reverberation Time." https://www.nti-audio.com/en/applications/room-building-acoustics/reverberation-time. [Accessed 07-02-2024].

[15] J. Mulcahy, "RT60 Graph." https://www.roomeqwizard.com/help/help_en-GB/html/graph_rt60.html. [Accessed 07-02-2024].

[16] J. M. V. Vidal, "e-REdING. Biblioteca de la Escuela Superior de Ingenieros de Sevilla. — biblus.us.es." https://biblus.us.es/bibing/proyectos/abreproy/70661/. [Accessed 07-02-2024].

[17] E. Labs, "Hyper-Reality Metaverse Research Laboratory." https://www.etri.re.kr/eng/sub6/sub6_0101.etri?departCode=170. [Accessed 07-02-2024].

[18] V. D. Community, "Source 2 - Valve Developer Community — developer.valvesoftware.com." https://developer.valvesoftware.com/wiki/Source_2. [Accessed 07-02-2024].

[19] Unity, "Unity Real-Time Development Platform." https://unity.com/. [Accessed 07-02-2024].

[20] M. Rey-Area, M. Yuan, and C. Richardt, "360monodepth: High-resolution 360deg *monoculardepthestimation*," 2022.

[21] G. Schwartz and K. Nishino, "Recognizing material properties from images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 1981–1995, 2020.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] https://www.theobjects.com/dragonfly/dfhelp/4-0/Content/09_Quantification/Connectivity.htm. [Accessed 12-01-2024].

[24] HP, "HP Reverb G2 VR Headset — hp.com." https://www.hp.com/gb-en/vr/reverb-g2-vr-headset.html. [Accessed 05-02-2024].

[25] Valve, "User&x2019;s Guide &x2014; Steam Audio Unity Integration documentation — valvesoftware.github.io." https://valvesoftware.github.io/steam-audio/doc/unity/guide.html. [Accessed 05-02-2024].

[26] A. P. Bureau, "Absorption Coefficients." https://www.acoustic.ua/st/web_absorption_data_eng.pdf. [Accessed 05-02-2024].

[27] A. Joshi, V. Deshmukh, N. Joshi, and P. Rane, "Studies on foliar sound absorption capacities of some urban trees by impedance tube method," vol. 23, 2013.

[28] Y. H. Kim, J. Y. Hong, P. J. Lee, and J. Y. Jeon, "Acoustical properties of vegetation including ground surfaces for scale model reproduction," in *Proceedings of Forum Acusticum*, pp. 903–907, European Acoustics Association, 2011.

[29] A. Engineering, "Digital Debunking: Could Simulation Help Food Manufacturers Achieve the Perfect Cereal Crunch? — altair.com." https://altair.com/newsroom/articles/Digital-Debunking-Could-Simulation-Help-Food-Manufacturers-Achieve-the-Perfe 2021. [Accessed 05-02-2024].

[30] CableMatters, "Foldable USB-C to 8K DisplayPort Adapter in Black — cablematters.com." https://www.cablematters.com/pc-1524-122-foldable-usb-c-to-8k-displayport-adapter-in-black.aspx. [Accessed 05-02-2024].

[31] Khronos, "OpenXR - High-performance access to AR and VR —collectively known as XR— platforms and devices — khronos.org." https://www.khronos.org/openxr/. [Accessed 05-02-2024].

[32] U. Store, "Runtime OBJ Importer — Modeling — Unity Asset Store — assetstore.unity.com." https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547. [Accessed 05-02-2024].

[33] G. Schwartz and K. Nishino, "Material recognition from local appearance in global context," 2017.

[34] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," 2015.

[35] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," 2017.

[36] A. Dourado, H. Kim, T. de Campos, and A. Hilton, "Semantic scene completion from a single 360-degree image and depth map," pp. 36–46, 01 2020.

[37] Audacity Team, "Audacity." https://www.audacityteam.org/. [Accessed 07-02-2024].

[38] "GitHub - MinisculeDust/RWTD — github.com." https://github.com/MinisculeDust/RWTD. [Accessed 01-02-2024].

[39] Y. Wu, Y. Heng, M. Niranjan, and H. Kim, "Sliceformer: Deep dense depth estimation from a single indoor omnidirectional image using a slice-based transformer," *Virtual Reality*, Oct 2023.

[40] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, "Bifuse: Monocular 360 depth estimation via bi-projection fusion," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[41] H. Jiang, Z. Sheng, S. Zhu, Z. Dong, and R. Huang, "Unifuse: Unidirectional fusion for 360° panorama depth estimation," *IEEE Robotics and Automation Letters*, 2021.

[42] Y. Li, Z. Yu, C. Choy, C. Xiao, J. M. Alvarez, S. Fidler, C. Feng, and A. Anandkumar, "Voxformer: Sparse voxel transformer for camera-based 3d semantic scene completion," 2023.

[43] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," 2019.

[44] A.-Q. Cao and R. de Charette, "Monoscene: Monocular 3d semantic scene completion," 2022.

[45] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," 2016.

[46] X. Yan, J. Gao, J. Li, R. Zhang, Z. Li, R. Huang, and S. Cui, "Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion," 2020.

[47] "Large Scale Parsing — buildingparser.stanford.edu." `http://buildingparser.stanford.edu/dataset.html`. [Accessed 31-01-2024].

[48] R. Cui, S. Qiu, S. Anwar, J. Liu, C. Xing, J. Zhang, and N. Barnes, "P2c: Self-supervised point cloud completion from single partial clouds," 2023.

[49] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," 2019.

[50] Z. Zhu, H. Chen, X. He, W. Wang, J. Qin, and M. Wei, "Svdformer: Complementing point cloud via self-view augmentation and self-structure dual-generator," 2023.

[51] Y. Li, Y. Dou, X. Chen, B. Ni, Y. Sun, Y. Liu, and F. Wang, "3dqd: Generalized deep 3d shape prior via part-discretized diffusion process," 2023.

[52] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," 2024.

[53] S. M. H. Miangoleh, S. Dille, L. Mai, S. Paris, and Y. Aksoy, "Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging," 2021.

[54] A. Dourado, T. E. De Campos, H. Kim, and A. Hilton, "Edgenet: Semantic scene completion from a single rgb- d image," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, Jan. 2021.

[55] G. Albanis, N. Zioulis, P. Drakoulis, V. Gkitsas, V. Sterzentsenko, F. Alvarez, D. Zarpalas, and P. Daras, "Pano3d: A holistic benchmark and a solid baseline for 360° depth estimation," *CoRR*, vol. abs/2109.02749, 2021.

[56] "Semantic Scene Completion — sscnet.cs.princeton.edu." `https://sscnet.cs.princeton.edu/`. [Accessed 31-01-2024].

[57] H. Dong, E. Ma, L. Wang, M. Wang, W. Xie, Q. Guo, P. Li, L. Liang, K. Yang, and D. Lin, "Cvsformer: Cross-view synthesis transformer for semantic scene completion," 2023.

[58] Y. Kasten, O. Rahamim, and G. Chechik, "Point-cloud completion with pre-trained text-to-image diffusion models," 2023.

[59] Valve, "Valve Time - Valve Developer Community." `https://developer.valvesoftware.com/wiki/Valve_Time`. [Accessed 07-02-2024].

[60] T. G. C. International, "A Model Depicting Team Development Stages — The Grove Consultants — thegrove.com." https://www.thegrove.com/methodology/team-performance-model. [Accessed 10-01-2024].

[61] M. . Company, "Enduring Ideas: The 7-S Framework — mckinsey.com." https://www.mckinsey.com/capabilities/strategy-and-corporate-finance/our-insights/enduring-ideas-the-7-s-framework, 2008. [Accessed 10-01-2024].

[62] J. Clear, "How to be More Productive and Eliminate Time Wasting Activities by Using the "Eisenhower Box" — jamesclear.com." https://jamesclear.com/eisenhower-box. [Accessed 10-01-2024].

[63] Atlassian, "What is Agile? — Atlassian — atlassian.com." https://www.atlassian.com/agile. [Accessed 10-01-2024].

[64] K. Boogaard, "How to write SMART goals (with examples) — atlassian.com." https://www.atlassian.com/blog/productivity/how-to-write-smart-goals. [Accessed 10-01-2024].

[65] E. Locke, K. Shaw, L. Saari, and G. Latham, "Goal setting and task performance: 1969–1980," *Psychological Bulletin*, vol. 90, pp. 125–152, 07 1981.

[66] D. Kahneman, *Thinking, fast and slow*. New York: Farrar, Straus and Giroux, 2011.

[67] MasterClass, "'move fast and break things': Pros and cons of the concept." https://www.masterclass.com/articles/move-fast-and-break-things, 2022. [Accessed 10-01-2024].

[68] NASA, "Technology Readiness Levels - NASA — nasa.gov." https://www.nasa.gov/directorates/somd/space-communications-navigation-program/technology-readiness-levels/, 2023. [Accessed 10-01-2024].

[69] S. R. Covey, *The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change*. New York: Free Press, 15 ed., Novembre 2004. ISBN-10: 0743269519 ISBN-13: 978-0743269513.

[70] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," 2001.

Appendices

Appendix A

3D Audio-Visual Scene Reproduction

A.1 Component Hierarchy

Order	Component	Description
1	**Model Importer (Game Object)**	An object holding the scripts to perform scenery import.
1.1	Import Scenery (Script)	`ImportScenery.cs`: A script to import scenery meshes from a .obj file generated by the pipeline.
2	**Directional Light (Game Object)**	Provides lighting to the scenery. Placed at coordinates $(x:0, y:5, z:0)$.
3	**Audio Source (Game Object)**	The Game Object defining the source of the audio.
3.1, 3.2	Sphere (Mesh); Mesh Renderer	Renders the object as a sphere so that the user can see the audio source during runtime; the Mesh Renderer changes the material during interaction (red by default, blue whilst being interacted with).
3.3	Sphere Collider (Component)	Gives the Audio Source a collider hitbox, allowing for interactivity through collision detection (i.e.: grabbing and moving the source).
3.4	Audio Source (Component)	Sets the audio file to play from the Audio Source (controlled by `ChangeAudio.cs` and `ImportSound.cs`); determines whether sound play behaviour is manual or automatic (controlled by `PlaySoundOnClick.cs` and `SoundClick.cs` scripts); determines attenuation settings (logarithmic, as previously described in the Acoustic Room Modelling section under "Steam Audio Source").
3.5	Steam Audio Source (Component)	Determines Steam Audio settings for producing the sound and its acoustics, as defined previously in the Acoustic Room Modelling section under "Steam Audio Source".

3.6	Rigidbody (Component)	Allows for the source to be physically moved throughout the environment.
3.7	XR Grab Interactable (Component)	Provided by OpenXR; Allows for the source to be physically moved throughout the environment through VR interaction events.
3.8	Change Audio; Play Sound On Click (Scripts)	Scripts, as defined in the description for the Audio Source Component, that allow for the audio source file to be changed and the audio to be played on button press, respectively.
4	**Steam Audio Static Mesh (Game Object)**	Game Object dynamically created by Steam Audio at runtime to handle its audio pipeline. Has a reference to the serialised mesh of all acoustic geometry of the scene, to which it is exported to.
5	**LocomotionArea (Game Object)**	A Game Object defining the area in which the user can locomote.
5.1	Quad Instance (Mesh)	A mesh defined by four corner vertices, allowing for simple resizing of its area on scene import. Located at $(x : 0, y : 0.163, z : 0)$, to account for the thickness of the floor when the scene mesh is imported at $(x : 0, y : 0, z : 0)$.
5.2	Resize Locomotion Area (Script)	`ResizeLocomotionArea.cs`: Calculates the combined bounds of all sub-meshes upon scene import from a new .obj file; repositions the four defining vertices of the Game Object's quad mesh; then re-renders the mesh.
5.3, 5.4	Teleportation Area; Continuous Move Provider (Component)	Components provided by OpenXR that allow for teleport locomotion (bounded to the Mesh's vertex coordinates) and continuous movement (upon the Mesh's surface).
6	**XR Origin (Game Object)**	Game Object asset provided by OpenXR, serves as the "origin" for the VR environment (tracking, interaction, etc.). Located at $(x : 0, y : 0.163, z : 0)$, to account for the thickness of the floor when the scene mesh is imported at $(x : 0, y : 0, z : 0)$.
6.0.1	XR Origin; Input Action Manager (Components)	Provided by OpenXR, manages all inputs sent from the VR environment.
6.0.2	Ray Hider (Script)	`RayHider.cs`: Enables the line visual for teleportation and interaction only during such events.
6.1	XR Interaction Manager (Game Object)	Provided by OpenXR, manages all interactions sent from the VR environment.
6.1.1	XR Interaction Manager, Input Action Manager (Components)	Allows for the binding and mapping of customisable interaction events (such as controller buttons for activating teleportation).

6.2	Camera Offset (Game Object)	Sets the offset from the origin for the camera (headset) and controller tracking; with the Tracking Origin Mode set to Floor, this allows the scene to be rendered at the height of the user registered by the headset.
6.2.1	Locomotion System (Game Object)	Provided with OpenXR's XR Origin Asset, this maps the locomotion system to the XR Origin.
6.2.1.1, 6.2.1.2	Teleportation Provider, Snap Turn Provider (Component)	Allows for the mapping of the Teleportation locomotion and Snap Turn to controller/action inputs.
6.2.2	Main Camera (Game Object)	The main camera from which the visual output is rendered.
6.2.2.1	Camera (Component)	Renders visual output to the target headset (the displays of both eyes).
6.2.2.2, 6.2.2.3	Audio Listener; Steam Audio Listener (Component)	Enables Steam Audio-calculated audio to be heard by the user based on camera position, as defined previously in the Acoustic Room Modelling section under "Steam Audio Listener".
6.2.2.4	Tracked Pose Driver (Input System)	Provided by OpenXR, changes the camera's orientation and position from the tracked inputs from the OpenXR Default Input Actions (corresponding to the headset).
6.2.2.5	Keyboard Move (Script)	`KeyboardMove.cs`: Disabled in the final build, this script allowed for moving the camera with keyboard and mouse input to evaluate audio spatialisation.
6.2.3	Left Controller (Game Object)	Game Object corresponding to the representation and functions of the Left Controller.
6.2.3.1, 6.2.3.2	Action-Based Controller Manager; XR Controller (Components)	Provided by OpenXR, allows mapping of all controller input actions to specified functions such as locomotion.
6.2.3.3, 6.2.3.4	XR Ray Interactor, Line Renderer (Components)	Provided by OpenXR and customised to a projectile curve, terminating in a 2D circular visual on the ground to allow for selecting a location to teleport to during Teleport Locomotion.
6.2.3.5, 6.2.3.6	XR Interactor Line Visual; Sorting Group (Components)	Further customisation for the ray interactor's line visual, with a high sorting layer to ensure no occlusion of visuals.
6.2.4	Right Controller (Game Object)	
6.2.4.1 - 6.2.4.6	[Ditto Left Controller]	Same as with left controller, though with differing mapped bindings for enabling function such as snap-turn with the right analogue stick.
7	**KitchenDemoScene (Game Object)**	Holds the mesh of a LIDAR scan of a Kitchen scene, provided by our supervisor, Dr. Hansung Kim, for use when running the demo scene (overlaid on top of pipeline output mesh).

7.1	Load Demo (Script)	`LoadDemo.cs`: Loads a specific pipeline-generated .obj file corresponding to the kitchen scene from the project's Resources folder, and enables the LIDAR mesh overlay.
8	**ModelWrapper (Game Object)**	Wrapper Game Object to hold the meshes obtained from any input .obj file.
8.0.1	Assign Materials (Script)	`AssignMaterials.cs`: Iterates through all sub-meshes of the contained Game Object corresponding to the .obj file input, assigning Steam Audio Geometry accordingly, then calls Steam Audio's Export function the current scene mesh.
8.1	[.obj filename] (Game Object)	Game Object holding all sub-meshes, split by material, of the pipeline output.
8.1.n	[.obj material name] (Game Object)	Sub-mesh extracted from .obj file; a contiguous mesh of a certain material.
8.1.n.1	Mesh Renderer (Component)	Renders the mesh within the scene, with the material colour for each mesh corresponding to each material recognition output.
8.1.n.2	Steam Audio Geometry (Component)	As defined previously in the Acoustic Room Modelling section under "Steam Audio Geometry", sets the Steam Audio Material for the sub-mesh with the corresponding parameters.

Table A.1: Breakdown of Unity scene's hierarchical components.

A.2 blenderflip.py

```
import bpy
import pathlib
import sys
from _ctypes import ArgumentError

def generate_flipped_mesh(obj_file_path):
    # Delete the default cube object
    bpy.ops.object.select_all(action = 'DESELECT')
    bpy.data.objects['Cube'].select_set(True)
    bpy.ops.object.delete()

    # import mesh
    str_filepath = obj_file_path.as_posix()
    bpy.ops.wm.obj_import(filepath = str_filepath,
            import_vertex_groups = True)

    # flip normals
    bpy.ops.object.editmode_toggle()
    bpy.ops.mesh.select_all(action = 'SELECT')
    bpy.ops.mesh.flip_normals()

    # translate object to "floor" height
    for obj in bpy.context.selected_objects:
        mtx_w = obj.matrix_world
        z_diff = min((mtx_w @ v.co).z for v in obj.data.vertices)
        mtx_w.translation.z -= z_diff

    # Save the object
    output_file_path = obj_file_path.with_stem(
            "final_output_scene_mesh")
    str_filepath = output_file_path.as_posix()
    bpy.ops.wm.obj_export(filepath = str_filepath,
                          export_material_groups = True)

def get_filepath(input_str):
    candidate = pathlib.Path(*input_str).resolve()
    if candidate.exists():
        return candidate
    else:
        raise ArgumentError("File not found.")

if __name__ == "__main__":
    # get filepath to obj file
    if len(sys.argv) > 2:
        raise ArgumentError(
                "Supply one argument (folder path) or none.")
```

```
obj_file_path = get_filepath(sys.argv[1:])
generate_flipped_mesh(obj_file_path)
```

Appendix B

Project Management

Skill	Ratings (1-5)				
	Adam	Atharv	Joe	Shubh	Sourish
Machine Learning	5	5	5	5	4
Mathematics	5	5	5	5	5
Game Engines	2	2	0	0	0
PyTorch	5	5	5	5	3
GitHub ML Codebases	4	5	4	4	3
Open3D	2	0	0	0	3
Unity	1	1	0	0	0
Research	5	5	5	5	5
Report Writing	5	5	5	5	5
Presentation	5	5	5	5	5
Graphic Design	4	3	3	3	3

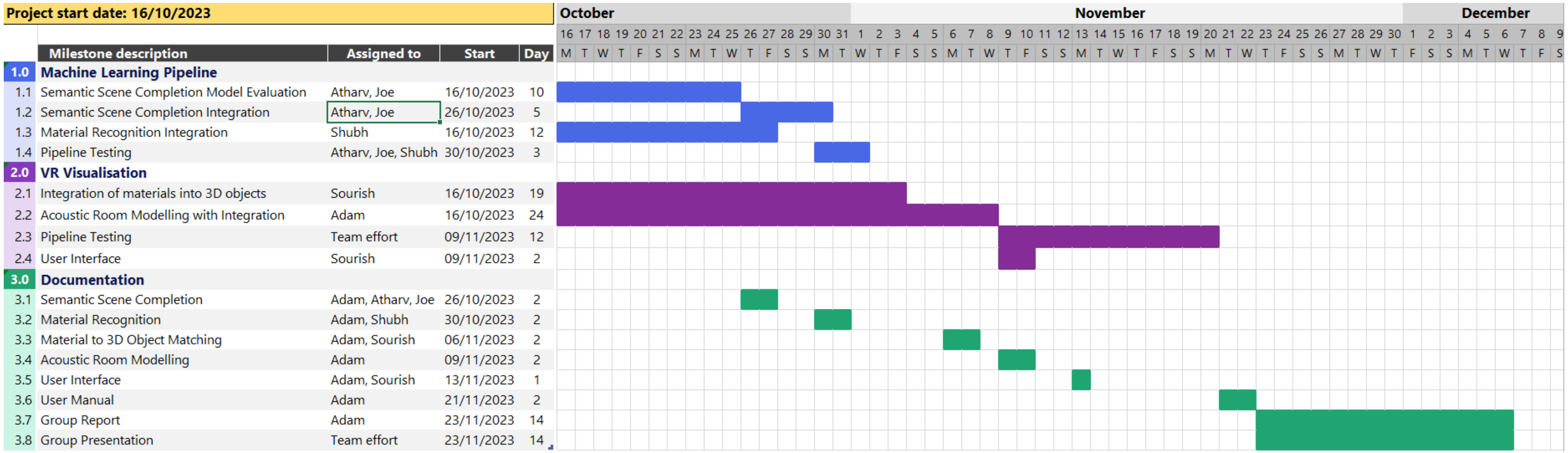
Table B.1: Table showing the skills audit done at the beginning of the project for each team member, with skill ratings from 1 to 5 where 5 is the skill needed to achieve the project's goals.

Item	Description	Quantity	Total Cost
USB C to DisplayPort 1.4 Adapter	VR headset adapter	1	£16.99
Total Cost			£16.99

Table B.2: Project bill of materials in Pounds Sterling (£).

Risk	Probability Score	Impact	Impact Score	Score	Prevention Strategy	Mitigation Strategy
Difficulty understanding concepts.	1	Delays development and report writing.	2	2	Dedicate more time to learning concepts. Discuss them with the supervisor.	Involve more team resources to learning. Reschedule team efforts accordingly.
Hardware loss or damage.	1	Freezes development and causes progress loss.	5	5	Backup of progress and code on GitHub to enable transitioning to alternate hardware once available. Securement of equipment in Building 16's locker when not in use	Order new hardware. Rediscuss project scope with supervisor in the case of large progress loss.
Codebase or data loss	1	Loss of progress.	5	5	Backup of progress and code on GitHub and OneDrive.	Rediscuss project scope with supervisor in the case of large progress loss.
Lack of necessary skills during development.	5	Slows down/freezes development.	2	10	Knowledge transfer within the team. Keep supervisor updated on upcoming tasks.	Find a detour around the problem.
Team breakdown	1	Freezes development. Costs the team time to rebuild cohesion.	4	4	Ensure effective coordination and communication is maintained between team members and supervisor.	Organise communication sessions.
Illness	1	Delays development.	4	4	Build buffer time by finishing project tasks as soon as possible.	Contact special considerations if the impact is significant.
Licensing and liability conflict.	1	Forces team to rewrite code with another software (if available) or use budget.	3	3	Thoroughly vet all models and resources: all must be authorised or provided by the supervisor or otherwise open-source.	Rewrite code with another software (if available) or use budget to buy license.

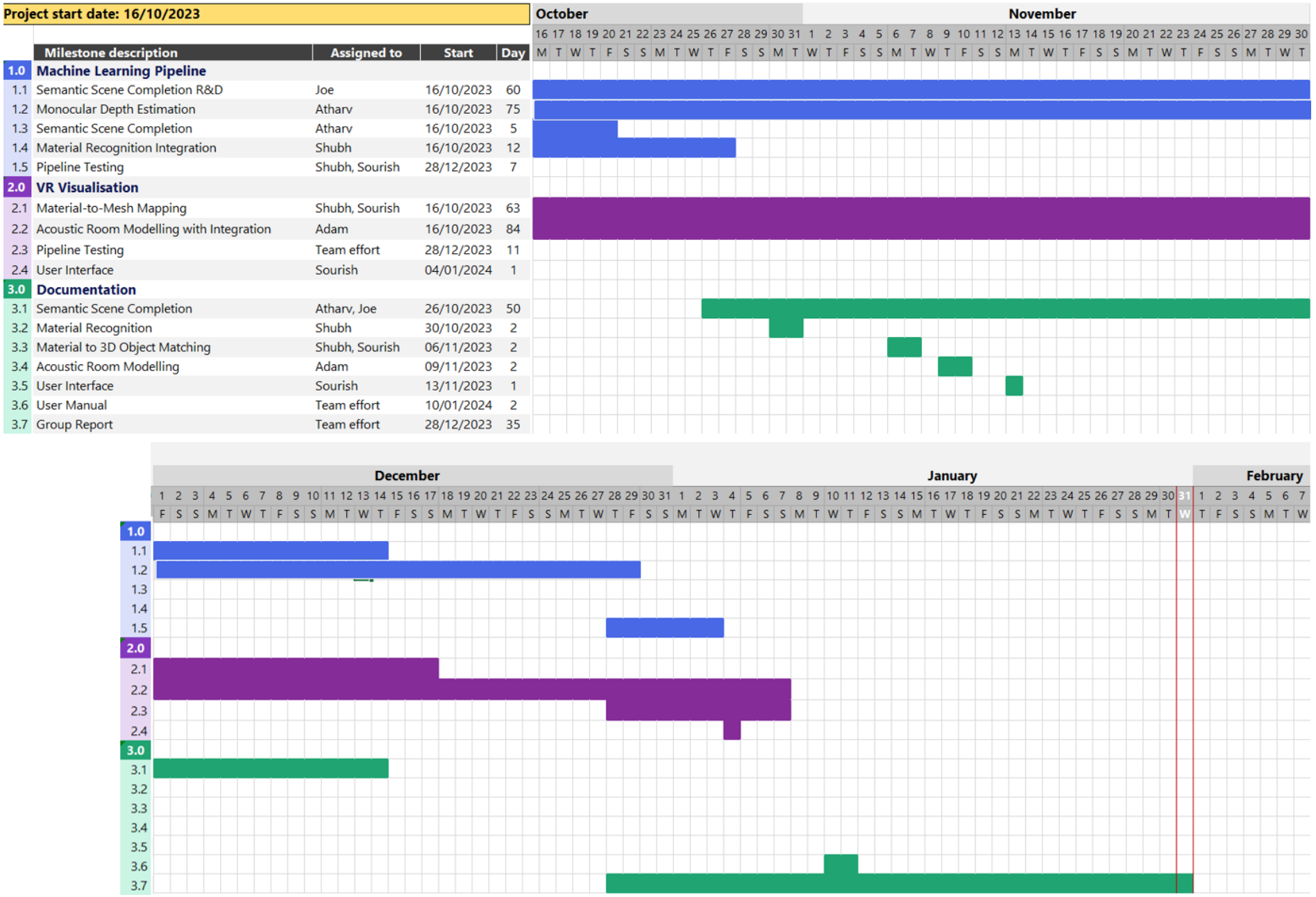
Figure B.1: The project's risk assessment table.



Project start date: 16/10/2023

	Milestone description	Assigned to	Start	Day
1.0	**Machine Learning Pipeline**			
1.1	Semantic Scene Completion Model Evaluation	Atharv, Joe	16/10/2023	10
1.2	Semantic Scene Completion Integration	Atharv, Joe	26/10/2023	5
1.3	Material Recognition Integration	Shubh	16/10/2023	12
1.4	Pipeline Testing	Atharv, Joe, Shubh	30/10/2023	3
2.0	**VR Visualisation**			
2.1	Integration of materials into 3D objects	Sourish	16/10/2023	19
2.2	Acoustic Room Modelling with Integration	Adam	16/10/2023	24
2.3	Pipeline Testing	Team effort	09/11/2023	12
2.4	User Interface	Sourish	09/11/2023	2
3.0	**Documentation**			
3.1	Semantic Scene Completion	Adam, Atharv, Joe	26/10/2023	2
3.2	Material Recognition	Adam, Shubh	30/10/2023	2
3.3	Material to 3D Object Matching	Adam, Sourish	06/11/2023	2
3.4	Acoustic Room Modelling	Adam	09/11/2023	2
3.5	User Interface	Adam, Sourish	13/11/2023	1
3.6	User Manual	Adam	21/11/2023	2
3.7	Group Report	Adam	23/11/2023	14
3.8	Group Presentation	Team effort	23/11/2023	14

Figure B.2: Gantt Chart representing the planned course of the semester.



	Milestone description	Assigned to	Start	Day
1.0	**Machine Learning Pipeline**			
1.1	Semantic Scene Completion R&D	Joe	16/10/2023	60
1.2	Monocular Depth Estimation	Atharv	16/10/2023	75
1.3	Semantic Scene Completion	Atharv	16/10/2023	5
1.4	Material Recognition Integration	Shubh	16/10/2023	12
1.5	Pipeline Testing	Shubh, Sourish	28/12/2023	7
2.0	**VR Visualisation**			
2.1	Material-to-Mesh Mapping	Shubh, Sourish	16/10/2023	63
2.2	Acoustic Room Modelling with Integration	Adam	16/10/2023	84
2.3	Pipeline Testing	Team effort	28/12/2023	11
2.4	User Interface	Sourish	04/01/2024	1
3.0	**Documentation**			
3.1	Semantic Scene Completion	Atharv, Joe	26/10/2023	50
3.2	Material Recognition	Shubh	30/10/2023	2
3.3	Material to 3D Object Matching	Shubh, Sourish	06/11/2023	2
3.4	Acoustic Room Modelling	Adam	09/11/2023	2
3.5	User Interface	Sourish	13/11/2023	1
3.6	User Manual	Team effort	10/01/2024	2
3.7	Group Report	Team effort	28/12/2023	35

Figure B.3: Gantt Chart representing the actual semester progress.

Appendix C

Sprint Documentation

Sprint Documentation

Sprint 1

Planning

TBD, refer to JIRA documentation and such

Meetings

Supervisor Meeting: 1

Team introductions

Supervisor gave laptop (4060 8GB; 32GB RAM), HP VR headset

- Common id: kproject and password: []

Presentation given by Dr. Kim

- Working on for 5 years
- VR/Metaverse: spatial audio not well developed, research in this area to improve human perception/immersion
- Spatial audio
    - Stereo/5.1ch audio not enough (not correct; difficult setup)
    - VR headset and binaural audio (6 DoF)
    - Gadget free system (difficult to implement)
    - Best way to reproduce: RIR (multi-order reflections from different materials in room); impractical as invasive, valid for static scenes at single point of measurement
- Goal overview: 360 image/video -> depth estimation/material recognition -> semantic scene reconstruction -> acoustic room modelling -> 3D audio-visual scene reproduction
    - Scene completion: "guess" occluded objects (e.g. table blocked by items on it)
- Single shot depth estimation: similarity gap between train and test will cause poor performance, use RWTD (reverse-gradient warming-up threshold discriminator) to mitigate
- Material estimation: limitations surmounted with DBAT
- SSC: edgenet used for semantic scene completion ("filling in holes")
- Acoustic modelling: synthesising RIRs using image source method for early reflection; ray-tracing for later earlier; statistical model based on Gaussian noise for late decay
- VR scene with spatial audio
    - Metadata OBJ for geometry; JSON for scene/acoustic information
    - VR platform: unity with steam audio
- Client: ETRI (SK); supervisor has project, forms an initial implementation for project – goal agreed with them by supervisor
    - Stretch goals are self-defined, no liaising with client necessary

    - Supervisor has already agreed on specifications with client
- Product management a vital part of this project – self-organised

Supervisor Meeting: 2

- Dr. Kim: has emailed PhD students for support in module implementation (contact should be minimised to ensure most research is by ourselves), gave links to datasets, etc.
- Discussion of team roles (as outlined in project brief)
- Handover of VR equipment and laptop (desk space provision received on Monday)
- Discussion of first sprint (2 weeks); parallel – high-level working pipeline;
    - Experimenting with models (e.g. VoxFormer)
    - Familiarisation with technical details/models
    - Dr. Kim spoke with team members regarding roles during sprint (recommendations of code/resources for each task)
        - Material recognition requires split and recombination of omnidirectional input due to model format requirements
        - Material recognition to semantic label for objects in 3D scene should be simple (one label per object)
        - Two contributions: research part (pipeline) and engineering (VR demo scene render)
- Next week's meeting will be online due to supervisor not being on campus
- Datasets, testing strategies
    - Best dataset: Stanford indoor scene dataset (edgenet 360 uses, depth estimation uses; common)
    - Testing: only Dr. Kim's dataset can be compared (audio ground truth comparisons), can provide datasets later for VR/audio testing – need ground truth/RIR (place virtual sound source and microphone in Unity/scene to match for comparison)
    - SSC: qualitative (check geometry for holes, etc.)
    - Material recognition: limited resources available (DBAT most recent)

Meetings Week 1:

- Mainly worked on completing project brief (defining deliverables, writing project specification, risk management, schedule/Gantt chart)

Meeting Week 2 (M1):

- Procured desk space and locker space for hardware/equipment (B16)
- Sprint planning meeting (sprint 1: blitzkrieg), as detailed in Jira (set up by Joe)
    - ~2 weeks, going over responsibilities for each member (exploring models, tools, etc. and creation of prototypes for each component of the pipeline)
- Continued discussions from supervisor meeting

Meeting Week 2 (M2):

- Sprint (blitzkrieg) standup meeting
    - Mostly setup (exploration) of each part of the pipeline on the part of each member

Meeting Week 3 (M2):

- Planning for Collation of documentation on pipeline efforts throughout sprint 1
    - Technical learnings, approaches
- Discussion of projection strategies (2D/cut to 3D/panoramic and vice versa)
- Exploration/research phases, initial prototypes for each phase of pipeline complete
- Sprint 2 plans broached informally

Retrospective

Adam:
automatic implementation
Unity 3D
R&D Completed

Sourish:
Voxel models done
Testing with Atharv's model
Communication positive

Atharv:
RWTD: Not perfectly suited
Pretrained network for SliceFormer
Working with EdgeNet360 (Linux or WSL or hybrid)
Time management: behind on model
GitHub
Testing model: dataset size
for phds: Sliceformer

Shubh:
material recognition working
contact PhDs
making script 360 -> 2D -> 360
Model resolution after inference

Joe:
Happy to hit targets and milestones
More Fine-grained tasks

Sprint 2

Planning

TBD, refer to JIRA documentation and such

Meetings

Meeting Week 4 (M1):

- Discussion with supervisor over sprint 1 achievements (getting prototypes for each stage of pipeline)
    - Discussion of results for material recognition (reliability, fragmentation, etc.)
        - Certainty/probability for material candidacy?
        - Extension of materials across detected objects (e.g. wall material to ceiling?)
        - Intrinsic/extrinsic parameters of camera used to capture omnidirectional image
            - Camera to use will have 4K capability, will use for texture mapping, reduce resolution for omnidirectional image to pipeline (which will also be downsampled)
- SSC:
    - Voxels should be coarse (2*2cm voxel size); unsure if parameter in pretrained model/code to change size
        - Voxels observed in results larger than 2cm (perhaps misrepresentation of scale?)
        - No ceiling, there is a parameter to add ceiling
        - SSC algorithm errors (gaps in table for example)
    - Smoothing, handling gaps (e.g. marching cube)
- Audio
    - Steam audio implementation discussion
- Buying equipment
    - Dongle
- Discussion of mathematic drawbacks/advantages of different projection types (cylindrical, spherical, cubic)

Meeting Week 4 (M2):

- Sprint retrospective (discussed as in sprint 1 documentation)
- Sprint 2 planning
    - Setting up of VR headset/environment
    - Validation of approach within windows for using WSL then getting output and running in windows
    - Sprint 1 documentation
    - Rectifying edgenet inconsistencies
    - Monocular depth estimation (decision whether to use sliceformer, midas)
    - Voxformer evaluation, decision on whether to use
    - Material recognition literature review, evaluation, enhancement

Meeting Week 4 (M3):

- Atharv has a working implementation of SliceFormer. He is going to investigate the inputs and outputs, which are not that clear at the moment. Also, since SliceFormer was trained on low-resolution images, he might ask to retrain the model on high-resolution images.
- I have collected a solid understanding of the SemanticKITTI and Stanford2D3D datasets, and conversion of formats seems possible.
- Sourish has laid out a plan to tackle the problem. He is also going to send an email to Hansung today to get his input.
- Shubh will be joining forces with Sourish starting from today.

Meeting Week 4 (M4):

- Adam:
    - All prerequisite resources for sprint 1 plan acquired, will collate by next meeting
    - VR headset installed and set up on home PC, will have to acquire laptop to replicate setup
    - Linux: VR headset only works with windows; created powershell script to test working with files within WSL and then from windows (successful)
- Atharv:
    - Sliceformer research (depth map generation)
        - Concern: doesn't take in just stereoscopic, also takes in ground truth depth map from dataset (couldn't find inference mode)
            - Might try with dummy files
    - Edgenet research
        - Extra dependency had to be compiled (extra cuda drivers) – cuda mismatch encountered
- Sourish
    - Kim advised to investigate depth map mapping to voxel
        - Truncated sine-distance function uses depth map to create a voxel representation of visible scene (can be matched with SSC output)
        - Found other papers for coordinate projection, mapping, alignment etc.
    - Will research on most feasible alignment method and attempt implementation of (somewhat) working model by end of sprint
- Joe
    - Research on "3D Sift" for solving material alignment problem
    - Voxformer: found method to create required format (requires raytracing)
        - Discussion with Hansung and Mona: say they are not hopeful on voxformer (trained on huge dataset, larger than Stanford, may be ineffective to retrain)
            - Alternative proposals: edgenet360 framework, built around edgenet core, assembles edgenet outputs from panoramic; can provide some models to replace edgenet core
            - Easiest: gap/hole filling; harder: object-based geometry approach, hard to generalise
    - Using another model to complete output model anomalies (comparison with ground truth)
    - Goal for this sprint: replace edgenet core model with others, evaluate results
- Sourish
    - Material mapping to scene: truncated sine function, looked at implementation; does pixel mapping from depth map to voxel coordinates, could add material array (voting); should be present in .obj file to import to unity

- Shubh

Meeting Week 5 (M1):

- Joe: on material recognition, leave improvement on side for this sprint (focus on merging labels with SSC mapping)
    - Experimentation of other models for core edgenet360 (edgenet), not much success found; hansung says mona gave advice on improving output with post-processing (e.g. gap filling, smoothing)
        - Couldn't find much literature on post-processing (e.g. interpolation); floated idea of half-shape transform
- Met yihong (phd student, here to clarify questions on SSC, depth mapping)
- Adam: VR (setup, plans for testing on laptop), linux to windows testing
- Atharv: sliceformer issue (two inputs, ground truth solved with dummy file)
    - Depth file format to png using temporary workaround, had to scale up
        - Asked further questions on conversion to yihong
    - Format required for ssc is a concern (mismatch from depth map output)
    - Showed results (output not resembling input); solution through bit depth of png (16-bit vs 8-bit)
    - Hansung gave some explanations and advice for improving output, gave explanation of his approach (stereoscopic matching, not applicable)

Meeting Week 5 (M2):

- Atharv: depth map format file to png
    - Sliceformer: dependency forgot to be removed, but can workaround with dummy file (has to be any valid with actual data), has to scale down to required resolution; output seems small, somewhat random resolution
    - Retrain depth model on pano3D
    - Found alternate depth map model online (large when cloning); will investigate
- Adam
    - Locomotion: teleport, have to create valid bounding plane
    - Will setup VR on laptop
- Shubh
    - Firewall issues with anaconda
    - Backup of laptop data (buy backup harddrive)
- Sourish:
    - Talked more on alignment (material mapping)
- Joe:
    - Not many pretrained models for edgenet core replacement, most old
    - Will take over from atharv for rectifying edgenet inconsistencies (e.g. generative model)
    - Spoke on point cloud approach vs voxels for interpolation

Meeting Week 5 (M3):

- Adam
    - Progress stalled due to VR issue; floated several ideas of resolving
        - Purchase of replacement cable expensive
        - Will try other methods to solve issue before trying warranty

        - RMA possible, may take long
        - Have own VR set, might go weekend to get it
- Atharv
    - Monocular depth estimation: docker environment difficult to run successfully
    - Tried alternate model: bifuse; results promising
- Shubh
    - Found NVIDIA diffusion model for shape completion, SDS-Complete
- Joe
    - Found 3DQD diffusion model for shape completion, will attempt to test to see if beneficial to SSC
- Sourish
    - Setup edgenet on own laptop, attempted to change parameters, edit
- Discussion of acquisition of SSD to hold datasets (large in size, 40 each)
- Anaconda dependency discussion, installation strategies

Meeting Week 5 (M4):

- Sourish: found CUDA file with pixel mappings; running into compile errors on WSL
- Adam: helped Atharv with depth mapping (360 monodepth); promising results until preprocessing stage
    - Will go get VR headset if feeling better (sick)
    - Will post documentation by end of day
- Joe: planning for next sprint (might help with unity)
    - Looking at point cloud models, realised incompatibility with depth maps
    - Looking for alternate ways to enhance depth maps
    - Presentation, talking on delivery, results-focused
- Shubh: material recognition mapping experiments, trying to map directly materials to voxels during edgenet process; running into errors with labelling

Meeting Week 6 (M1):

- Material recognition mapping continued
    - Tried to piggyback off voxel mapping; issues encountered in assignment of material array
    - Hansung outlined two methods for implementing material mapping (3d mesh to 2d image origin, or vice versa)
    - Joe talked on our panoramic projection proposal (alignment)
    - From experience, hansung claims that we can assume after importation 3d mesh orientation will be invariant (i.e. same alignment)
- 360 monodepth identified, implemented, some problems with 3D mesh created
    - Hansung said phd student (yihong) currently retraining on higher resolution for possible monodepth
    - Hansung said to provide code for scaling depth maps properly; will have to consider parameters (“real”, then scale to trained depth for inference)
    - Colour mapping to correct grayscale (can’t just RGB, must assign from colour spectra or from actual depth back to grayscale)
    - Consider: if relative depth, may have to find way to convert to absolute depth (scaled to expected input format)
- Vr headset problems:
    - Will have to contact support
    - Stopgap with own headset

Sprint 2 Restrospective:

Shubh:
Not as good as the first one for his part
Hard to know what to do
SSC: found models
VR: headset issue
Topics dragged along during meetings

Adam:
Didn't achieve quite as much as Blitzkrieg
Upset about not achieving as much
Factors: VR headset issue, illness
Happy to help Atharv
Promise on SSC pipeline
Meetings were fine
Joe is alright
Intercommunication between teams: Solution: Discord #support channel
Not enough awareness on Sourish's work

Sourish:
Not feeling that he has achieved what was meant to be achieved
Good team work with Shubh
Meetings are informative

Atharv:
Sprint was more research-based
Due to external factors: VR headset, SliceFormer low-res, WSL issue
Meetings tend to drag on: flagging when topics take some time

Joe:
Link up modules
Set stricter deadlines, due to a shift from the R&D nature of sprints to a more executive one.

Sprint 3

Planning

TBD, refer to JIRA documentation and such

Meetings

Meeting Week 6 (M2):

- Sprint retrospective, to be inserted here:
  - Creation of support channel in discord (separate threads)
- Sprint 3 Planning, task refinement, assignment (as seen on JIRA)
  - Material mapping: split into two concurrent approaches (3D to 2D using CUDA array; investigation into alternate 3D projection or alignment techniques)
- Planning of presentation

Meeting Week 6 (M2.5):

- Group presentation planning

Meeting Week 6 (M3):

- Atharv: busy with other module, will complete work over weekend
- Adam: VR headset secured, will work on VR env over weekend; starting collation of sprint 2 documentation
- Sourish: progress made with material mapping, but GPU memory errors detected (could be TensorFlow error)
- Shubh: open3D investigation; func for voxel grid from point cloud could be used

Meeting Week 7 (M1):

- Presentation feedback (Hansung says good, will show to customer when publicly available)
- Adam: Vive secured, development will continue (problems with HP reverb support)
- Sourish: material mapping; separate voxel grid (assigning to material value)
- Hansung spoke on conversion from voxels to mesh (need to separate meshes; by default edgenet creates single scene mesh rather than separate objects)
  - Shubh explained some of implementation/approach
  - GPU error was inconsistent; shubh did not encounter (material map appears to be successful, but outputs mesh? Need to sort this out)
  - Hansung explained interior of .obj file for our reference (voxels, normals, faces, etc. held in textual/numerical format within); proposed some way to separate mesh by object

Meeting Week 7 (M2):

- Discussion of particulars of material mapping implementation (and how it will integrate with Unity)

- Review of progress on all sprint activities
- Joe: laptop broken, on return will test open3D application for hole-filling (object completion); will finish jira documentation for current sprint; will try and separate mesh object detection (e.g. tables -> table 1, table 2, allowing for separate material assignments)

Meeting Week 7 (M3):

- Continuing progress on work mentioned Monday

Meeting Week 7 (M4):

- Ditto
- Adam: VR camera working, now need to develop locomotion system (should be done over weekend); HP support will send VR cable for Reverb headset

Meeting Week 8 (M1):

- Hansung spoke on alignment of mapping (segmentation, materials, etc.), and how these will interface with Unity
- Shubh spoke on material array implementation considerations (thread concerns with CUDA programming, how voxels are mapped – commonality used to determine)
    - Logic sound: works in python; when using CUDA/C++ issues still arise (difficult to determine source of error)
- Discussed the downsampling problem for voxels
- Adam: VR; head tracking works (steam audio listener applied, solved floor), controllers not detecting but locomotion code ready to go when this issue is resolved
- Atharv: more dev in 360monodepth; WSL dependency errors delayed progress somewhat; tested yihong's new model (spurious data output on inference, though works on given test set)
    - Hansung highlighted that likely issue is incompatibility of depth map format/values
    - Hansung worked through with Atharv trying to help identify focus areas to resolve issues

Meeting Week 8 (M2):

- Sprint 3 extended to next week
- Adam: elaborated more on VR progress (only controller detection remaining; will collate Sprint 1 + 2 reports when all information collated from other team members)
- Atharv: yihong new model generates worse depth map at higher resolution
    - Need to rescale values to adapt to coordinate range for depth map (~absolute)
- Joe: setting up environment on own laptop (gpu not on own laptop; wasn't running on cpu mode); did some research on more point cloud completion approaches (SVDFormer)
- Sourish and Shubh: explained more on current mapping issues (CUDA assignment issue, occluded voxel considerations); with this solved, the rest should fall into place

Meeting Week 8 (M4):

- Sourish/shubh: Material mapping done (dictionary maps, mostly correct); mesh doesn't have labels

- Adam: will try recreating Unity scene from scratch for VR first, then adding steam; cable arrived, will test

Meeting Week 9 (M1):

- Atharv: worked on conversion from 360md depth map to absolute depthmap values
    - Some variation/inconsistency in ordering of colours (e.g. blue/red for closest)
        - Can determine this through top/bottom pixel colour (ceiling or floor colour will change to closest colour assignment as approaches image edge)
    - Tried midas on jupyter midas to do isolated testing; ran into crashing errors on final line (output display)
- Sourish: Object separation attempts (run edgenet 360 on alpha cluster); trying to create .obj file with separate objects/meshes for each material – alpha cluster issues, contacted support
    - Joe recommended smaller version created locally
- Joe: on laptop; setting up partial->complete repository for point-cloud completion; NVIDIA user variable in environment throwing error, (no CUDA-HOME); narrowing down solution
- Adam: VR cable works; dev on HP headset resumed
    - TP locomotion and VR scene complete
    - Fixed steam audio to allow for spatialisation

Meeting Week 9 (M2):

Meeting Week 9 (M3):

- Sprint retrospective TBD
- Writing out what tasks remain for next sprint(s) TBD

Meeting Week 10 (M1):

- Hansung: spoke on some evaluation strategies to include in report (e.g. comparison to actual data from audio measurements in the rooms scanned); coordination with PhD student for data (e.g. clarinet)
- Adam:
    - Tasks
        - Make/finish user manual
            - Remember: shubh updated his sprint 1 para
        - Collate documentation on sprints
        - Replace sound source with given echo chamber sounds (speech/clarinet)
        - For final demo: overlay LIDAR/texture
    - Showed VR demo again (had headphones for spatialisation)
    - Added smooth loco; added better controller models
- Shubh:
    - Worked more on combination, automation of pipeline
- Joe
    - Floated idea of demo allowing for examiners to experience spatialisation; hansung proposed doing in 15 mins after (questions, etc.)
    - CWK deadlines affecting work (accounted for in risk management); progress steady

    - CUDA versioning issue still providing issues; work done for a lot of what would proceed after (after resolution, would progress quickly)
    - 2TB harddrive requested a while ago, no reply yet from university requisitioning team
- Atharv: resolved kernel crash issue in jupyter notebook
    - Solved colour to grayscale map; scaling – must test with 360 monodepth

Meeting Week 10 (M2):

- Atharv: spoke on how CUDA issues could be resolved (creating new environment), proposed to Joe how to solve his own
- Adam: will start on report during holiday, require sprint 3 documentation paragraphs
    - For remaining week, will focus on adding automation to unity
    - Need list of materials from sourish/shubh
    - LIDAR: manual overlay for transparency
- Spoke on scheduling over next week and beginning of holiday

Meeting Week 10 (M3):

- Adam: require sprint 3 filling in; will upload unity project for perusal
- Joe: will shift focus to report
- Sourish/shubh: mesh separation by material progressing well
    - Asked adam whether possible to split in unity; will investigate
- Atharv: 360monodepth building on laptop; now outputs grayscale
- Report planning (timescale/schedule)

Sprint 3 Restrospective:

- Adam:
    - Happy with results
    - Environment works
    - Unlucky with time
- Sourish:
    - Good teamwork
    - Good throughput
- Atharv:
    - .exr to .png -> satisfied
    - 360Monodepth -> disappointed. Not much time could be invested because of intelligent agents
- Shubh:
    - better than previous sprint -> Material mapping working
    - Could have done more
    - Get mesh file splitting working
- Joe:
    - Technical progress could have been better. Held back by NVIDIA CUDA issues.

Meeting Week 14 (M1-Final meeting with supervisor):

Hansung's feedback:

Adam:

- -Focus on the report but if you can get the unity scene to properly save between scenes, then try to do it.
- -An omnidirection field of reception for audio should be fine and then just use a mono audio channel, can change setting in unity to change "this shape"
- -For Room Input Response, use sweep and gunshot audios, Speaking and music audios for demo
- -For materials you can't find acoustic properties of, match it to the next closest thing you can find

Atharv:

- -It's good to have options for ground truth and 360monodepth -Talk about approached in report
- -good tech details, good eval, and good proj management for report. good criticism about what was planned and what happened, and talk about learning outcomes.

-Add Appendix

-Add User Manual

Content of the presentation: Similar to the previous one, but with some kind of experimental result like a video, like 30 sec or 1 min. Would you be okay if we give the VR headset for people to try individually during demonstration (after showing the second examiner)? It's a good idea to do it after, not during, and also in a different room.

Poster: Big A0 poster, title, motivation, goals, some technical details, outcomes, 28px text size

Appendix D

User Manual

D.1 Video Tutorial

Please find the pipeline video tutorial at the following link: https://drive.google.com/file/d/1xv0wRaHHgQdnxnMK1-g_A3mq_oKdDDTw/view?usp=sharing

Please find the Unity video tutorial on the following link: https://drive.google.com/file/d/1jhwR5SVirXHkpy0qZN7Ht4kOQtGDC6Be/view?usp=sharing

D.2 User Manual

Manual

VR Equipment and Laptop Setup

The following outlines the setup of the specific use-case of running the project on the University-provided laptop; the project can run on any VR-capable PC, but the physical setup will vary by platform.

HP Reverb G2 VR Headset Setup:

- Connect the power cable to a wall outlet, after ensuring the power output cable from the hub is connected to the power adapter.
- Connect the input cable (*DisplayPort*) to the port on the headset behind the mounted magnetic gasket (which must be removed to connect, then reattached afterward).
- Connect the display cable output (*DisplayPort*) to the *DisplayPort-to-USBC* adapter, then connect this to the laptop's *USBC* port.
- Connect the data cable (*USBC*) to the *USBC-to-USBA* adapter, then connect this to the laptop's *USBA* port.
- Press the button on the square hub into which all cables are routed to power on the headset.
- Ensure the controllers are charged (*AA* batteries can be accessed through sliding down on the panel near the thumb grip area). Turn them on through holding down the **Windows** button underneath the analogue sticks.
- Launch the **Windows Mixed Reality** application on the computer; you are ready when the application's default scenery renders to the headset, and you can see both controllers tracked to where you physically hold them.

System Requirements

ML Pipeline Requirements

- You must have approximately **200GB** of free storage
- You must have an active internet connection to run Monocular Depth Estimation
- You must have a system with an `NVIDIA` graphics card due to the *CUDA* dependency
- All the individual components of the pipeline require different *Anaconda* environments to run, except for *360MonoDepth* which uses a Docker container. The environment requirements for the different modules can be found and set up using the instructions provided on their repositories:
    - *360MonoDepth* - https://github.com/manurare/360monodepth

    - *DBAT* - https://github.com/heng-yuwen/Dynamic-Backward-Attention-Transformer/tree/main
    - *EdgeNet 360* - https://github.com/SimonLisowski/Mesh-Generation
      Note: The Docker container should be set up using the modified 360MonoDepth code on the main project repository of this due to the modifications made to make it compatible with the pipeline.
- While not a requirement, it is recommended to run this on a newer machine with an SSD

Unity Requirements

- You must run the program using the *Unity Hub* (into which the *Unity* project folder: `Project/AV-VR/Unity/Project/AVVR` is imported as a "project on disk"), which must have installed the *Unity Editor* version **2022.3.11f1**
- An NVIDIA graphics card is required to enable *Steam Audio* functionality
- A VR headset is required to output the scene's render: the *HP Reverb G2* is recommended, but the project is designed to run platform-independently
- The VR environment of the headset must be running (i.e. *Windows Mixed Reality*, *SteamVR*, etc.) before pressing play when the scene is loaded

Generating a scene

Generating a scene is an effortless task. This section provides guidance for running the pipeline for 3D scene reconstruction for VR from a single omnidirectional input image. The whole pipeline includes namely depth estimation using *360MonoDepth*, Material Recognition using *Dynamic Backward Attention Transformer*, Semantic scene completion and Material mapping using *EdgeNet360*, with a separate final import of the scene in *Unity* which is outlined in the next section of this manual.

The pipeline provides three distinct ways to run the pipeline:

1. *Using any RGB image with our* Monocular Depth Estimation *module*
2. *Using any RGB image and its ground truth depth maps from the* Stanford2D3D Dataset Area 3
3. *Using any RGB image and its corresponding depth map (already produced using stereo matching)*

To run the pipeline, the `GUI.py` script provided in the scripts folder should be run. The User Interface provides the user with the choice to run the pipeline using the options outlined above. This is demonstrated in *Figure 1*.

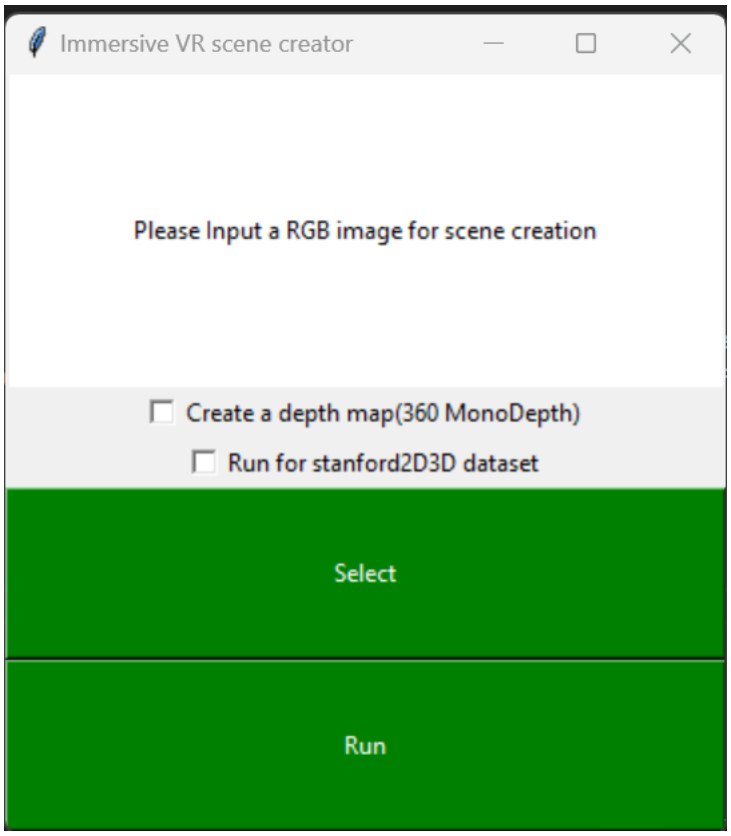


Figure 1 GUI providing the options to run the 3d reconstruction pipeline for VR

The scripts folder contains all the necessary scripts and python files used for automating the pipeline.

Running with the Monocular Depth Estimation Module

Running this component is simple. As a pre-requisite, when 360MonoDepth has been set up and built with Docker initially, you must manually edit all the .ps1 scripts except for masterscript.ps1 and ensure that the $imageNameOrId variable is set to the image ID (and not container ID) of the associated 360MonoDepth Docker image (can be retrieved using the Docker Desktop application). This step is only required at the very initial set up stage.

To run the pipeline with our Monocular Depth Estimation module, run `GUI.py` and make sure the **"create a depth map"** option is checked.

Clicking the green **Select** button will then provide the user with a dialogue prompt to navigate to and select the RGB image on which they wish to apply Monocular Depth Estimation.

Clicking the **Run** button next will run the Machine Learning Pipeline without the need for the user to manually intervene. The main part of the User Interface displays the status of the pipeline, whereas, the command line displays exactly what is being executed.

The output depth map is saved in the `Data/Input` folder within the *EdgeNet360* directory ready to be passed into that module. The *`splt_img.py`* file runs and splits the RGB image

and saves the cube face images in the `material_recognition/Dynamic-Backward-Attention-Transformer/split_outputs` folder.

The output material recognised cube face images from *DBAT* are saved in the `material_recognition/Dynamic-Backward-Attention Transformer/output/split_output"` folder. The omnidirectional material map image is then saved in the `Input` folder of `Edgenet360`.

Check the command prompt whilst the scene is creating. The command prompt displays the material assigned to each objects for verification, as seen in *Figure 2*. Once the command prompt displays the line seen in *Figure 3*, the pipeline execution is finished, press any key. *EdgeNet360* creates the necessary output `.obj` and `.mtl` files for the voxel grid and the mesh saving them in the `edgenet360/Output` folder.

Restart the widget to create another scene, to rerun the pipeline.

Once the mesh has been created, please refer to the next section for the procedure of generating and rendering an acoustically-modelled VR scene from it.

Running with Stanford2D-3D

To run the pipeline using the images from the *Stanford2D3D* dataset and their corresponding ground truth depth maps, firstly download the *Stanford2D3D* dataset and run the *Ground Truth Extraction* on the images in the **Stanford** folder as instructed in the *EdgeNet360* repository.

Then all you need to do is run the `GUI.py` python script and check the option: Run for *`stanford2D3D`* dataset and provide the input RGB image using the select image option, insert which area the image is in, as seen in *Figure 2* and select **Run**. The output files i.e. the material recognised images, cube face images and the `.obj` & `.mtl` files for the voxel grid and mesh will be saved in the same directories as explained before. The material image is saved in the appropriate folder by the scripting process so that it can be provided to *EdgeNet360* along with the depth map. The final mesh scene is saved in `edgenet360\output`.

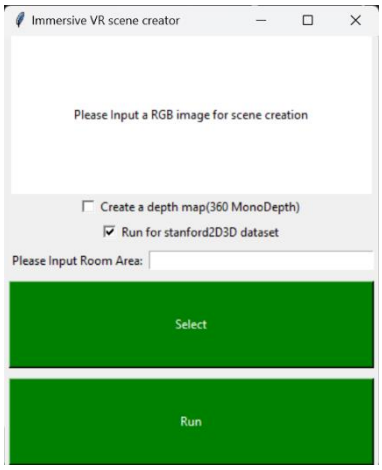
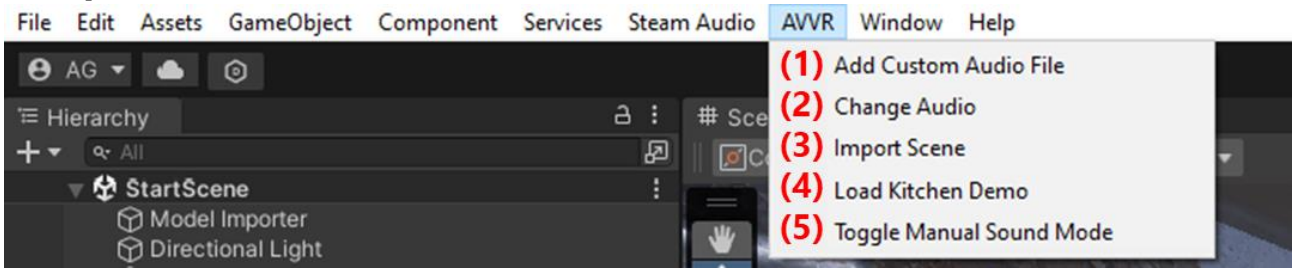


Figure 2: The GUI input for running the Stanford2D3D dataset. Insert Image and area of the Stanford dataset, e.g. area_3

Running a scene

Once a mesh has been created by the ML pipeline process outlined previously, it is time to render a Virtual Reality scene to experience the acoustic modelling that can be generated with it.

Setup



*The **AVVR** custom menu and its items.*

1. Run the **AVVR** project from the *Unity Hub* application to launch the Unity environment, this should launch into *"Edit"* mode.
2. Select the **AVVR > Import Scene (3)** context menu in the top bar of the Unity window to import the *.obj* file generated in the previous section. A file selector will open, starting in the default output folder of the ML pipeline; any .obj file on the PC can be selected, however. Once selected, the scene presented will dynamically change to allocate the new mesh.
3. By default, the audio that will play in the VR scene is a clip of some music played on a clarinet.
    - If desired, this can be changed by selecting the audio source in the project hierarchy (**AVVR > Change Audio (2)**, on the top context menu) and selecting the desired audio under the *"AudioClip"* field of the popup window. Custom audio files can be added to this list by using the **AVVR > Add Custom Audio File (1)** menu item.

    - The default behaviour of the sound can be changed; with the **AVVR > Toggle Manual Sound Mode (5)** menu item, this can be switched between playing automatically or manually with a controller button press (mapping listed in the following *Play Mode* section of this manual)
4. Ensure the Windows Mixed Reality application (or the corresponding environment for the chosen VR headset, such as the SteamVR application) is running, then press the play button at the top of the window; this will render the scene to the PC's connected headset.

Play Mode

Upon pressing play, the spatialised audio (with reverberations for the room applied to the mesh generated by the ML pipeline) will be playing throughout the scene, which you can navigate throughout to experience.

VR Controls

- **Left or Right Controller Trigger (Back Button):**
    - ***Teleport:*** A selection sphere will appear on the ground; release the trigger to instantly teleport your position to its location.
    - ***Move Sound Source:*** Point a controller at the Audio Source (a red sphere at the middle of the room), and press the trigger to pick it up. The source will stay in attached to your hand until you let go of the trigger, allowing you to move it to any position within the room.
- **Left Controller Analogue Stick:**
    - ***Smooth Movement:*** Moving the stick in any direction will allow yourself to move smoothly along the ground within the scene (relative to where you are facing).
- **Right Controller Analogue Stick:**
    - ***Quick (i.e. "Snap") Turn:*** As an alternative to physically orienting yourself in the direction you wish to face, you may also position the stick left or right to quickly turn yourself (facing 45 degrees to the left or right from your previous gaze, respectively).
- **Left or Right Controller Buttons (X/Y):**
    - ***Play Sound:*** With **AVVR > Toggle Manual Sound Mode (5)** switched on, this will execute the playback of the sound source.

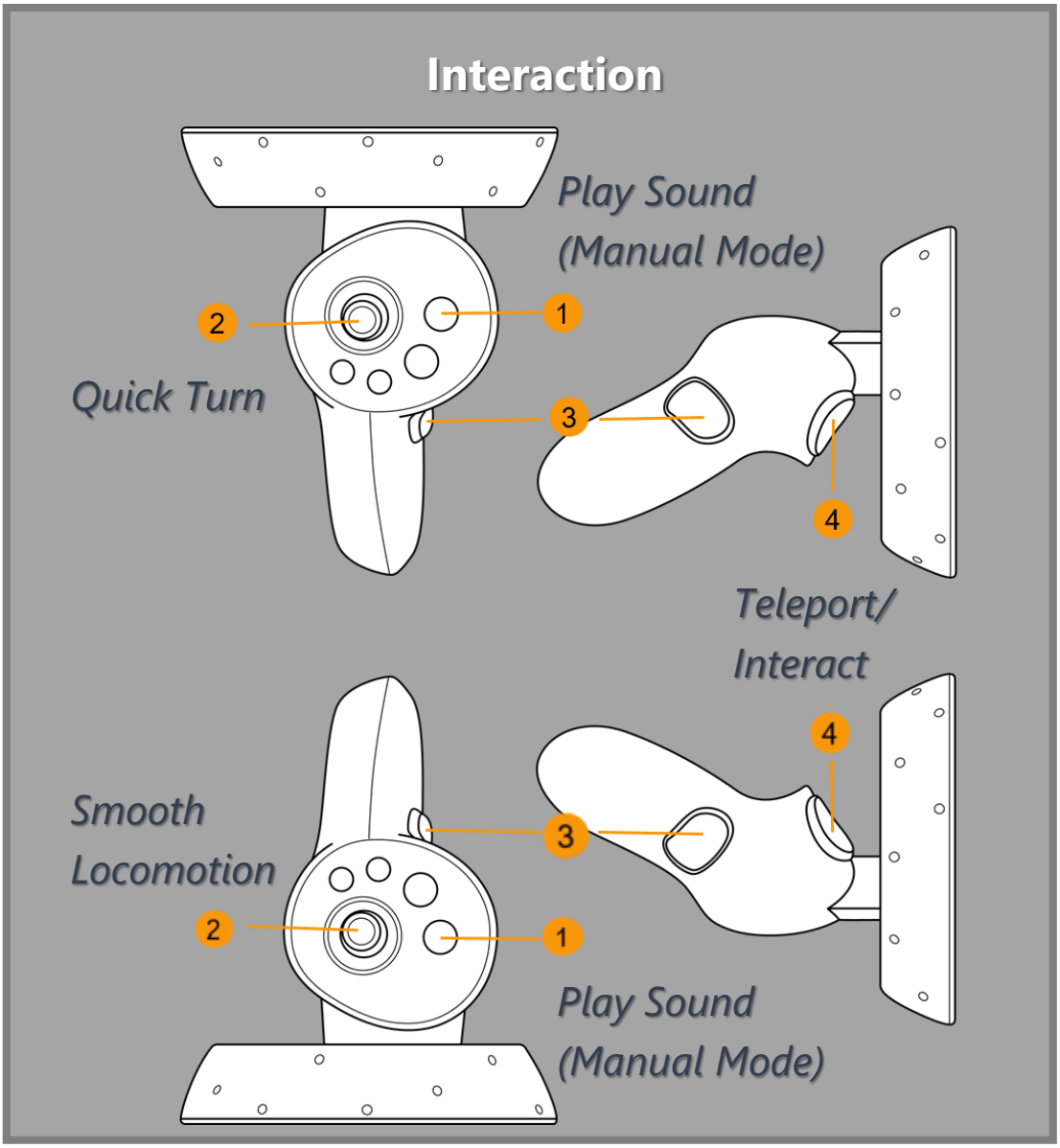
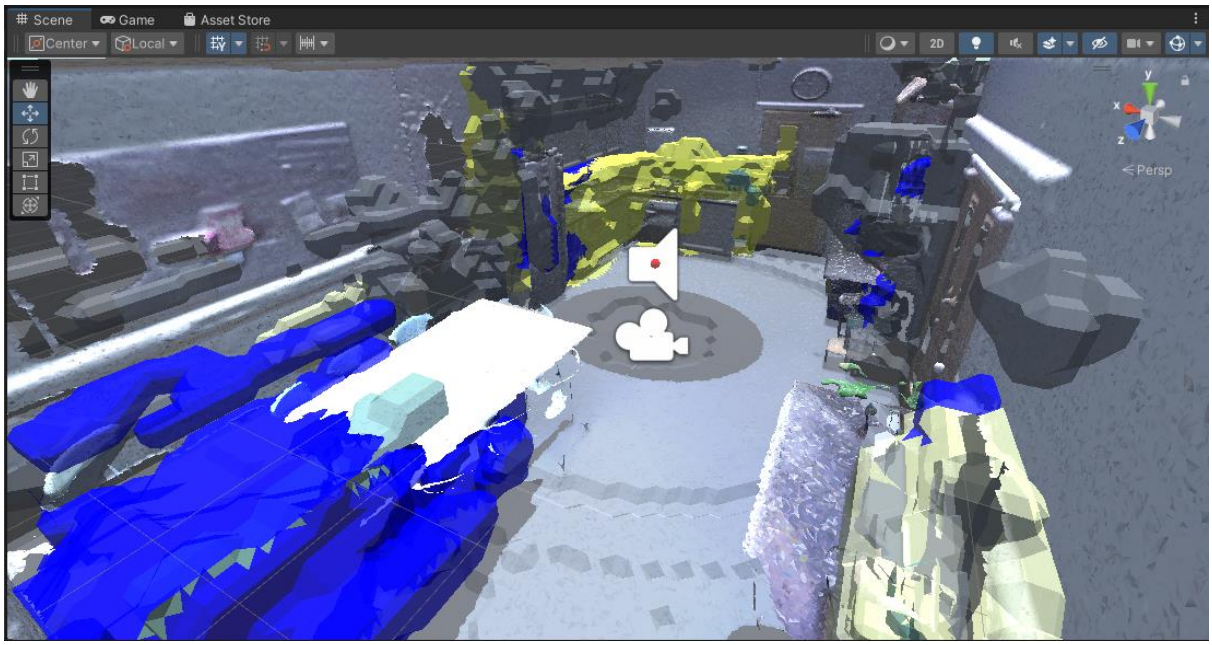


Diagram of controller inputs for the HP Reverb G2.

Demo Scene

Additional functionality exists under the **AVVR > Load Kitchen Demo (4)** menu item; this will load a pre-prepared scene generated of a kitchen scene, with the automatic overlay of a LIDAR scan of said kitchen with the pipeline output set to transparent. This scene is useful for helping demonstrate the real-life scene corresponding to the mesh our pipeline generates, and is more visually appealing than the scenes obtained from the default pipeline mesh import.



Unity Edit Mode view of the Demo Scene.

NB: A video demonstrating this part of the pipeline is present in the Project Archive, named "VR Demo.mp4".

Appendix E

Project Brief

School of Electronics and Computer Science

ELEC6200 MEng Group Design Project

Project Specification and Plan

GDP Group 6: Immersive Audio-Visual Virtual Reality Scene Reproduction

Academic Supervisor:	Dr. Hansung Kim
Team Members:	Joe Boutros, Adam Gafar, Shubh Harshad, Sourish Mukherjee, Atharv Sankholkar
External Partner:	Dr. Sung-Uk Jung Head of *Hyper-Reality Metaverse Research Laboratory, Electronics and Telecommunications Research Institute*

Project Specification

Traditionally, in the advancement of Virtual Reality fidelity for applications such as the metaverse, research and development have long neglected the incorporation of realistic audio in its simulated environments – a vital component in user immersion. Utilising research provided by our supervisor (Dr. Hansung Kim, outlined in his seminal 2021 paper[1]), and the adoption and reimplementation of suitable pretrained models, the objective of this project is the creation of an end-to-end pipeline that takes an omnidirectional image input and renders a real-time 3D environment (likely in a game engine such as *Unity*), with simulated acoustic properties derived from the materials detected therein. The audio solution is to be implemented utilising spatial audio packages such as *Steam Audio* by *Valve*. As Dr. Kim's approach used a stereo camera setup, a cost reduction element exists in that the input for this project is to instead be obtained from a single 360-degree photo (produced by a consumer-grade camera).

Correspondence with the external client on the part of the team is limited due to time differences and their workload; as this project is based on much of Dr. Kim's research, liaison and ratification of the core deliverables for the project were performed between them in advance – these are listed in the next section of this brief.

In terms of the final result, the expectation is to produce a proof-of-concept system to run on a supplied machine (a *Dell Inspiron 16 Plus* with an *NVIDIA GeForce RTX 4060* Graphics Card), and to be experienced with a supplied VR headset (*HP Reverb G2)*. As the product is not customer-facing, the software may not necessarily be elaborate, but is required to be of simple operation ("a few clicks" at most) to achieve the desired outputs. Inference, or running of the system, has no strict time limit; the recommendation is that the run-time is tractable (no more than a couple hours), with emphasis on seamless integration of pipeline components and real-time rendering performance for the VR scene.

With regards to testing strategies, the performance of different models is to be qualitatively compared when selecting which to use in each component (and, of course, evaluated on test data sets when training them). For semantic scene reconstruction, comparisons could also be made to physical measurements of scanned room dimensions. During later stages, the plan is to consult with Dr. Kim and PhD students working under him to develop more quantitative methods. As for acoustic modelling, comparison to ground truths is difficult (as this requires Room Impulse Response measurements); either comparison with results and RIRs from the

[1] Kim, H. et al., 2021. *Immersive audio-visual scene reproduction using semantic scene reconstruction from 360 cameras.* Virtual Reality, 30 October, 26(3), p. 823–838.

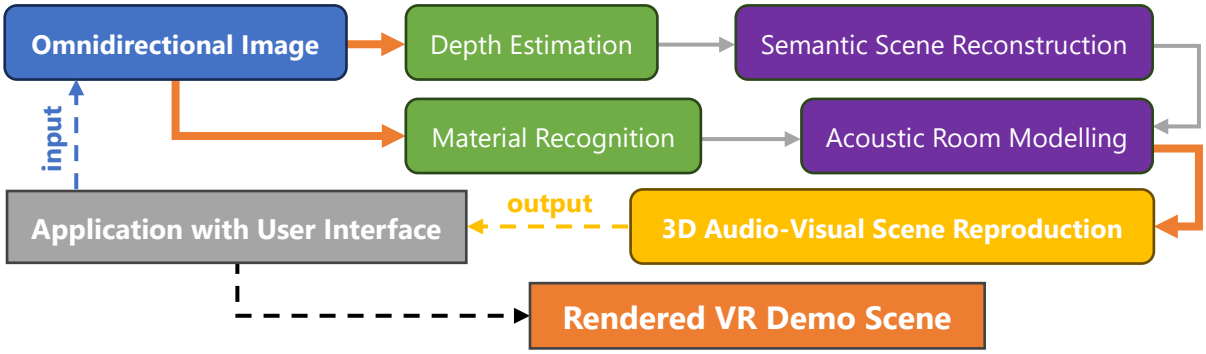
2021 paper would be attempted, or perhaps anonymous participants could be given a survey to evaluate simulated acoustics versus physical recordings for selected environments.

Project Plan

Deliverables Table:

Objectives	Stretch Objectives
Re-implementation, adaptation, and testing of required components (including evaluation of alternate approaches/models):	User customisation of selected parts of pipeline/scene (e.g. material labels, 3D model/scene tweaks)
Monocular depth estimation	Ambient environmental audio from Generative AI
Semantic scene completion	Use of the VR headset as the source of the original omnidirectional image
Material recognition	
3D model importation to game engine; acoustic room modelling/simulation	
Rendering and demonstration of VR scene	
Linking of components to and implementation of pipeline (omnidirectional image input from single camera; outputting real-time VR scene with spatial audio effect)	
Implementation of pipeline demo system with simple UI	

System Diagram:



Risk Analysis Table (Scores / 5: **P**robability, **S**everity; ***** Product):

Risk	P	S	*	Mitigation strategy
Loss of hardware or damage; loss of data or codebase	1	4	**4**	Backup of progress and code (*GitHub*) to allow for transition to alternate hardware; securement (e.g. locker) of equipment when not in use.
Scope management issues	2	3	**6**	Restriction to focus on main deliverables in event of scope creep.
Delays due to implementation difficulties, team member disposition, etc.	3	4	**12**	Implementing slip in time plans; renegotiation/focusing of deliverables after consultation with supervisor/client; consultation with PhD students under Dr. Kim for assistance
Licensing and liability conflict	1	5	**5**	Thorough vetting of all models, resources, etc.: all must be authorised/provided by supervisor or otherwise open-source.
Teamwork and coordination breakdown	1	4	**4**	Ensure effective coordination and communication is maintained between team members and supervisor.

Project Management

Gantt Chart of Planned Project Schedule:

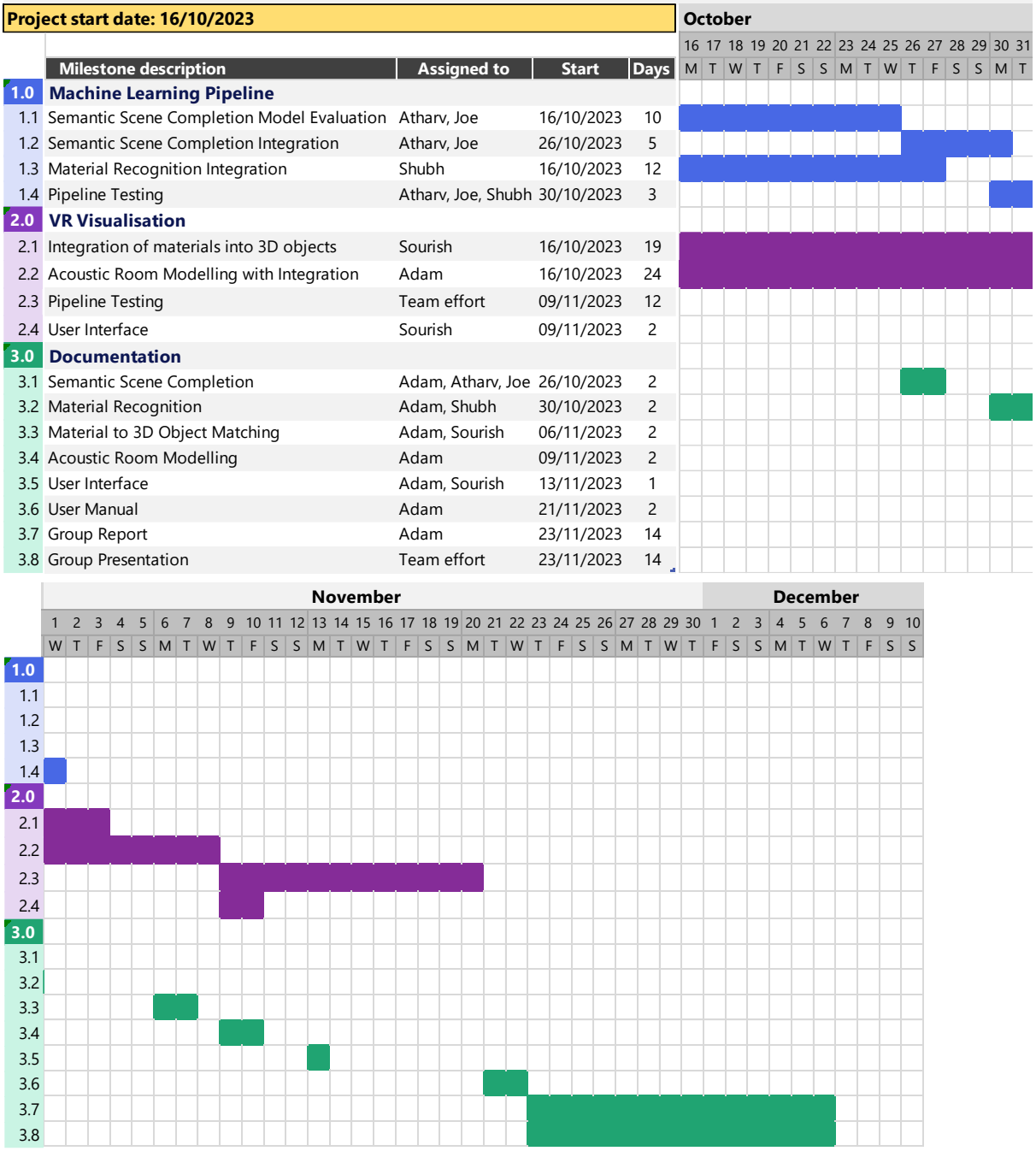
Immersive Audio-Visual Virtual Reality Scene Reproduction

Group 6

Adam Gafar, Atharv Sankholkar, Joe Boutros, Shubh Harshad, Sourish Mukherjee

Project start date: 16/10/2023

	Milestone description	Assigned to	Start	Days
1.0	**Machine Learning Pipeline**			
1.1	Semantic Scene Completion Model Evaluation	Atharv, Joe	16/10/2023	10
1.2	Semantic Scene Completion Integration	Atharv, Joe	26/10/2023	5
1.3	Material Recognition Integration	Shubh	16/10/2023	12
1.4	Pipeline Testing	Atharv, Joe, Shubh	30/10/2023	3
2.0	**VR Visualisation**			
2.1	Integration of materials into 3D objects	Sourish	16/10/2023	19
2.2	Acoustic Room Modelling with Integration	Adam	16/10/2023	24
2.3	Pipeline Testing	Team effort	09/11/2023	12
2.4	User Interface	Sourish	09/11/2023	2
3.0	**Documentation**			
3.1	Semantic Scene Completion	Adam, Atharv, Joe	26/10/2023	2
3.2	Material Recognition	Adam, Shubh	30/10/2023	2
3.3	Material to 3D Object Matching	Adam, Sourish	06/11/2023	2
3.4	Acoustic Room Modelling	Adam	09/11/2023	2
3.5	User Interface	Adam, Sourish	13/11/2023	1
3.6	User Manual	Adam	21/11/2023	2
3.7	Group Report	Adam	23/11/2023	14
3.8	Group Presentation	Team effort	23/11/2023	14



Project Management Roles:

Team members are to contribute to the project as shown in the above chart. With regard to project management roles, Joe has been elected as the overall project manager, assuming responsibilities such as team organisation, meeting coordination, communication management, and ensuring the completion of deliverables. Adam has been assigned as the project officer, coordinating with members to produce minutes, reports, and documentation.

Appendix F

Directory Tree

```
./
├── 360monodepth
│   ├── Dockerfile
│   ├── LICENSE
│   ├── README.md
│   ├── code
│   ├── data
│   ├── imgs
│   └── weights
├── Unity
│   ├── Example Meshes
│   ├── Project
│   └── Scripting (WSL)
├── edgenet360
│   ├── Data
│   ├── Mesh-Generation
│   ├── Output
│   ├── README.md
│   ├── enhance360.py
│   ├── eval_stanford.py
│   ├── infer360.py
│   ├── infer360_stanford.py
│   ├── lib_edgenet360
│   ├── log
│   ├── preproc360_stanford.py
│   ├── preproc360_stanford_batch.py
│   ├── read_pointcloud_mat.py
│   ├── replace.py
│   ├── src
│   └── weights
├── material_recognition
│   └── Dynamic-Backward-Attention-Transformer
│       ├── Dockerfile
│       ├── README.md
│       ├── __pycache__
│       ├── checkpoints
│       ├── combine_img.py
│       ├── data
│       ├── datasets
│       ├── environment.yml
│       ├── output
│       ├── path_.txt
│       ├── split_img - Copy.py
│       ├── split_img.py
│       ├── split_output
│       ├── test_indoor.zip
│       ├── torchtools
│       ├── train_sota.py
│       └── ui.py
├── scripts
│   ├── 360monodepthexecution
│   ├── GUI.py
│   ├── blenderFlip.py
│   ├── combined.bat
│   ├── combined_ogV.bat
│   ├── combined_stanford.bat
│   ├── edgenet.bat
│   └── material.bat

29 directories, 29 files
```

Appendix G

Authorship

G.1 Adam Gafar

G.1.1 Main Body

- **Background:** 2.5, 2.6
- **Methodology:** 3.4, 3.5, 3.6
- **Results:** 4.3, 4.4, 4.5
- **Conclusion:** 6.2, 6.3 (with all team members)
- **Future Work:** 7.6, 7.7

G.1.2 Appendices

- **Appendix A.1** (Component Hierarchy)
- **Appendix A.2** (`blenderflip.py`, with Sourish)
- **Appendix C** (Sprint Documentation)
- **Appendix D.1** (Unity Demo Video)
- **Appendix D.2** (User Manual: VR and Unity sections)
- **Appendix E** (Project Brief)

G.2 Atharv Sankholkar

G.2.1 Main Body

- **Background:** 2.1, 2.2
- **Methodology:** 3.1, 3.7
- **Results:** 4.2.1
- **Research & Development:** 4.2.1
- **Conclusion:** 6.3 (with all team members)
- **Future Work:** 7.1, 7.2 (with Joe)

G.2.2 Appendices

- **Appendix D.2** (User Manual: ML sections, with Shubh and Sourish)
- **Appendix F** (Directory Tree)

G.3 Joe Boutros

G.3.1 Main Body

- **Introduction:** 1.1, 1.2, 1.3.1, 1.3.2
- **Research & Development:** 5.3, 5.4
- **Conclusion:** 6.1, 6.2
- **Future Work:** 7.2 (with Atharv), 7.3
- **Project Management:** 8.4.1, 8.4.2, 8.4.3 (with Shubh); all other content

G.3.2 Appendices

- **Appendix B** (Project Management)

G.4 Shubh Harshad

G.4.1 Main Body

- **Background:** 2.3, 2.4
- **Methodology:** 3.2, 3.3, 3.7
- **Results:** 4.2.2, 4.2.3
- **Conclusion:** 6.2, 6.3 (with all team members)
- **Future Work:** 7.4, 7.5
- **Project Management:** 8.4.1, 8.4.2, 8.4.3

G.4.2 Appendices

- **Appendix D.2** (User Manual: ML sections, with Atharv and Sourish)

G.5 Sourish Mukherjee

G.5.1 Main Body

- **Background:** 2.4
- **Methodology:** 3.3, 3.7
- **Results:** 4.2.3
- **Conclusion:** 6.3 (with all team members)

G.5.2 Appendices

- **Appendix A.2** (`blenderflip.py`, with Adam)
- **Appendix D.2** (User Manual: ML sections, with Atharv and Shubh)