Immersive and Efficient 3D Gaussian Splatting in VR

From Capture to Render to User

HOANG NGUYEN DEREK ZHU RICH CHEN

UNIVERSITY OF WASHINGTON

HOANGN@CS.WASHINGTON.EDU DEREKZ0@CS.WASHINGTON.EDU RC2002@CS.WASHINGTON.EDU

Fig. 1. This is a sample gaussian splat of Derek's bedroom. The structure from motion (SFM) metadata including camera field of views and image clips. This represents the software side of our pipeline, which will be explored more later in the paper.

This project explores a system for efficient and immersive 3D environment capture, reconstruction, and rendering, emphasizing collaborative data sharing and point cloud reuse for Gaussian Splatting, alongside immersive spatial audio reconstruction. Our approach introduces a novel chunking technique that divides environments into manageable blocks, enabling selective processing, storage, and retrieval of point cloud data. This facilitates the reuse of previously processed data and supports collaborative reconstruction, where users can contribute to a shared 3D representation by capturing and uploading data for specific chunks. We also investigate a hardware wearable device for capturing visual and multi-channel audio data, and implement spatial audio processing techniques to enhance the immersive experience. Our results demonstrate the potential of chunking to improve the scalability and efficiency of Gaussian Splatting for large-scale environments, and the importance of integrated audio for creating more compelling and realistic immersive experiences, paving the way for collaborative 3D scene and spatial audio reconstruction.

Author's address: Hoang Nguyen Derek Zhu Rich Chen

University of Washington

hoangn@cs.washington.edu derekz0@cs.washington.edu

rc2002@cs.washington.edu.

1 INTRODUCTION

Our motivation for the project was to build both hardware wearable and a software platform for users to interact with their "memories" both efficiently and immersively. In order to "capture" memories, we would need the wearable such that it would record a user's surroundings (visuals and audio). With the memory "recorded" we

University of Washington
hoangn@cs.washington.edu derekz0@cs.washington.edu rc2002@cs.washington.edu

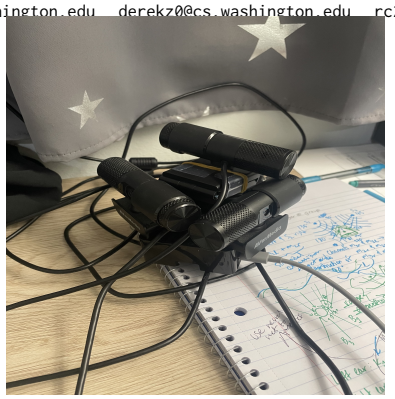

Fig. 2. This is a constructed wearable consisting of 3 cameras and 6 microphones. While bulky, inefficient, and difficult to power, it is able to somewhat capture 3 images per second and 6 channels worth of audio* (* issues being expanded upon in the paper). This represents the hardware side of our pipeline, which will also be explored more later in this paper.

could then use **3D Gaussian Splatting** and **Spatial Audio Reconstruction** to effectively rebuild an immersive environment to be explored in VR. The use cases for such a project is varied, from uses cases for individuals suffering from Alzheimer's or dementia to uses in architecture or engineering. Most of all it would just be a super cool form of entertainment, to be able to relive some of your favorite memories.

While not many products or companies have looked deeply into commercializing the whole pipeline that we undertook, we definitely utilized existing research in [3D Gaussian Splatting (ref the paper)] and spatial audio [ref the paper]. For the task of reconstruction, we decided to use existing tooling like **Jawset Postshot**, **Unreal RealityCapture** and **Colmap** to produce intermediary artifacts like point clouds (both sparse and dense) and camera registry files. Though we did not write a renderer ourselves, we used these tools to perform experiments on how to optimize and refine the reconstruction pipeline. As for the hardware, we were limited both in time and in finances, so though in a perfect world we could have obtained multiple 360 degree cameras, stereo cameras, and/or the Meta Orion lenses, unfortunately we had to make do with 3 AverMedia webcams. Though this posed a challenge, it was a good challenge and forced us to be more creative when thinking of solutions.

Specifically, for our project, we not only wanted to explore the pipeline for capture, reconstruction, and rendering, but also wanted to improve immersiveness through the use of **multiple set spatial reconstruction** and splat efficiency through **point cloud merging**, **chunking**, and **semantic segmentation** with varying levels of results for each of these. Though with more time, our results and explorations may be more in depth and detailed, we were forced to cut our time short and selectively choose what to spend our time on.

Overall we discovered a method for splatting with a cached point cloud [1], a systematic design for chunking and stitching of an environment for future splatting [2], and a method to do 3D spatial audio reconstruction [3]. Though our findings may not be completely novel and are built off of existing tooling, our results indicate that there are routes to explore in the path of caching for reconstruction.

1.1 Contributions

- We introduce a method for splatting a new scene (t+1) where t is the time of the environment, utilizing the point cloud of scene (t) and images from (t+1)
- We introduce a method for systematically chunking a capture into "blocks" and programatically determining all camera views associated with that block in order to cache and piece by piece later.
- We introduce a method to produce spatial audio captured from the environment
- We introduce a demo of a wearable piece of hardware that captures images and audio of the environment of the user wearing it.

2 RELATED WORK

Significant amounts of prior work were referenced to get an idea of the state-of-the-art technology in this area.

The core of our project involves Gaussian splatting, a technique with significant prior research. The key paper in the area is the seminal paper *3D Gaussian Splatting for Real-Time Radiance Field Rendering* [Kerbl et al. 2023] that first introduced the technique.

There is notable prior work on point clouds and their properties as well as alternative 3-dimensional representations. We referenced a significant amount of prior work in areas regarding 3D representations. This is shown through the research of papers such as *Real-time moving object detection and removal from 3D pointcloud data for humanoid navigation in dense GPS-denied environments* [Rath et al. 2020] and *OctoMap: an efficient probabilistic 3D mapping framework based on octrees* [Hornung et al. 2013].

Considerable prior work exists on understanding room acoustics and spatial room audio. For this, the paper *Past, Present, and Future of Spatial Audio and Room Acoustics* [Koyama et al. 2025] was a significant reference, highlighting the important developments in understanding how audio propagates within a room.

We also looked into spatial audio capture methods and ambisonics, especially with such prior work as *Binaural Reproduction of Higher Order Ambisonics A Real-Time Implementation and Perceptual Improvements* [Vennerød 2014], *Ambisonics Capture using Microphones on Head-worn Device of Arbitrary Geometry* [Bastine et al. 2022], and *DIFFBAS: An Advanced Binaural Audio Synthesis Model Focusing on Binaural Differences Recovery* [Li et al. 2024] to find effective methods to handle directional audio data.

There is significant prior work in machine learning for spatial data, as summarized in the 2022 paper *An overview of machine learning and other data-based methods for spatial audio capture, processing, and reproduction* [Cobos et al. 2022]. We referenced algorithms regarding signal capture and processing from this review.

3 METHOD

As our research and methodology spanned across both hardware and software spaces, we will be sure to detail both promising results we found and the failures we encountered in the process of refining our

application. Even though many of the rabbit holes we went down were not super promising, we believe that they may serve as "trailheads" for future work down the line. Specifically our methodology will cover: the hardware wearable, our splat chunking process, the ideas that went into our audio reconstructor, and finally our other attempts at improving the splatting pipeline (including semantic-segmentation and point cloud reuse).

3.1 Hardware Wearable

Its well known that to produce a good Gaussian splat, we need many images of the surrounding environment. Although it is feasible to constantly record with a phone for example, due to the one-way nature of recording and the lack of perspectives, it is very hard to produce a holistic reconstruction (in fact oftentimes using a stationary recording will result in a splat that is effectively just a frame). The idea is that we would like to develop some sort of wearable similar to the Orion glasses that can record our surroundings but from multiple views. Unfortunately, we are not Meta, nor do we have the resources to compete against them, but what we realized is that we can still get fairly far with 3 webcams and a Raspberry Pi.

The wearable itself is shockingly uncomplicated. 3 AverMedia webcams are plugged into a RPI4 in the position of an equilateral triangle (where each side is one camera). Althoughugh not completely 360 degrees, for our MVP run, we simply had the wearer strap it on their head and turn a little more when doing captures. As the user moves, we capture their surroundings, both from the webcam cameras but also with the microphones built into the webcam. These information would be saved and used later in our pipelline for splatting and audio reconstruction.

3.2 Chunking

A significant challenge lies in the effective reuse of point clouds or splats from prior environments. This engenders a non-trivial problem, namely the registration of arbitrary point clouds with potentially limited overlap, a problem that remains largely unresolved.

Existing literature offers limited direct solutions, as most point cloud registration algorithms presuppose substantial overlap between the datasets. While certain works, such as the hierarchical Gaussian splatting approach [Kerbl et al. 2024], explore merging splats across training segments, these methods typically rely on simple concatenation and assume minimal scene change. Related methodologies in robotics, for instance, those exemplified by [Yugay et al. 2024], similarly leverage the high degree of overlap inherent in point clouds acquired from sensors within close temporal proximity, often employing voxelization and Iterative Closest Point (ICP) for registration.

Our initial investigations involved a brute-force approach. This entailed determining the center of the dense point cloud (splat), iteratively zooming out, and capturing images at $360/k$ angular intervals to estimate rotation. Following this, a rudimentary rotation alignment was attempted, and the scene was projected onto 3-axis coordinates n times (corresponding to n angles) to explore the use of SIFT features for translation calculation. This methodology proved to be exceptionally inefficient and was constrained by a fixed bounding volume (due to the 2D plane projection), rendering it unsuitable for our purposes.

We also evaluated the global registration algorithm provided by the Open3D package, which is predicated on the RANSAC algorithm [Nousias et al. 2023]. Although we increased point cloud overlap through voxelization, this approach still yielded unsatisfactory results. The core issue stems from the algorithm's reliance on matching distributions of neighboring points. Consequently, point clouds with similar distributions but spatial adjacency tend to merge into a single dense cloud, rather than remaining as distinct, neighboring entities.

Fig. 3. Results of merging two corners of a room using Open3D's global registration. The bottom images show the original point clouds, while the top images illustrate the voxelized point clouds after merging.

Our refined strategy adopts a different perspective. Instead of attempting to computationally derive rotation, translation, and scale, we propose to leverage information gathered directly from the user. Rotation data is obtained from the wearable's IMU sensor, while scale and translation are inferred from GPS data and user-provided height. Given the wearable's fixed position on the user's head, we can reasonably approximate the camera position as coinciding with the user's head position. This, in conjunction with GPS data, enables the computation of an approximate location and orientation for the reconstructed scene. This facilitates the construction of a global real-world representation, serving as a cache for our splatting pipeline, bypassing the point cloud reconstruction phase (for a majority of users), and enabling the creation of rich scenes with significantly reduced data acquisition, as users can share data. For example, if colleagues have already generated a point cloud and camera pose estimation for a classroom, a user would only need to capture data for their specific area of interest.

University of Washington
hoangn@cs.washington.edu derekz0@cs.washington.edu rc2002@cs.washington.edu

It is important to note that we do not cache the splat itself, as we rely on third-party software for training, and our limited exploration within the project timeframe did not identify open-source software that permits post-training modification of image data.

3.3 Multiset Spatial Audio

Each AverMedia webcam on our wearable has 2 microphones, a left and a right one. With 3 webcams, we have 6 channels of audio to essentially record from, and with sound being represented as a wave, we could effectively capture sound coming from any direction.

Fig. 4. A crude depiction of the wearable's microphones. Each microphone pair is denoted with m_x with specific left and right microphones labeled as m_{xl} and m_{xr} respectively

Essentially with these 6 stereo microphones we want to capture sound spatially, determine sound source position based on the microphone array's geometry, interpolate and combine the audio data in order to reconstruct a 3D sound field, and finally to render the ambisonic audio for immersive experience in the virtual environment.

Fig. 5. A very simplified example of how we would use the microphones to "detect" where sound is coming from.

A significant challenge included in this reconstruction is the issue of time-of-arrival differences (ToA) between microphones when determining sound direction. Furthermore, in the diagram I make the assumption that only 2 microphones would receive the highest intensity sound waves in order to determine the origin of the sound. However very noisy environments or environments a multitude of audio sources will heavily impact the reliability of this assumption. Effectively what we are trying to do here is SLAM but only with audio cues.

The research for this took more time than I would have liked, but in the end we were able to lock down a very simple and hacky method of processing the audio:

3.3.1 Acoustic Signal Processing. So even before we collect or process any audio, we must synchronize all microphones to record at the same sample rate. As the webcams automatically have software to apply high-pass filtering for noise reduction, we don't have to worry about that. Once this is complete we started collecting audio.

3.3.2 Direction of Arrival Estimation. A major issue with our approach is that we want to choose the two microphones getting the most direct input, but to do this we need to compute the **time difference of arrival** between all microphone pairs which is basically just determining when each microphone started picking up a certain sound. To do this we chose to use an industry standard: **generalized cross-correlation with phase transforms** defined by this equation:

$$R_{ij}(\tau) = \int_{-\infty}^{\infty} X_i(f) X_j^*(f) e^{j2\pi f\tau}\, df \tag{1}$$

where:

- $R_{ij}(\tau)$: Cross-correlation function between signals X_i and X_j ebign left and right respectively.
- $X_i(f)$: Frequency-domain representation of signal i.
- $X_j^*(f)$: Complex conjugate of the frequency-domain representation of signal j.
- $e^{j2\pi f\tau}$: Complex exponential term that introduces a time shift τ.
- τ : Time shift parameter.
- f : Frequency variable.
- df : Differential element in the integration over all frequencies.

We essentially must use this to solve for the sound source position by triangulating multiple ToA estimates. We also chose to compute the interaural level difference and interaural phase difference for the left-right stereo pairs. The interaural level difference helps to determine the lateral position of the sound while the interaural phase difference refines the "distance" of how far away the sound is. These values are used as weighted constants along with the multiple ToA estimates in order to solve for the sound source coordinates (represented as x, y, and z). We use least-square minimization to solve this.

3.3.3 Reconstruction. In order to do reconstruction (we want to produce some ambisonic sound field) we convert the microphone signals to first-order ambisonics using a process known as eigenbeam beamforming:

The values of B_0, B_x, B_y, and B_z are given by:

$$B_0 = \frac{2}{1}(p_1 + p_2 + p_3) \tag{2}$$

$$B_x = \frac{p_1 - p_3}{2} \tag{3}$$

$$B_y = \frac{p_2 - B_0}{2} \tag{4}$$

$$B_z = 0 \quad \text{(No height microphones in current setup)} \tag{5}$$

where:

- p_1, p_2, p_3 are the pressures measured at different microphone positions.
- B_0 is a combined pressure term.
- B_x and B_y are spatial pressure components.
- $B_z = 0$ because no height microphones are used in the current setup.

We also choose to use plane-wave decomposition in order to fully utilize all microphone pairs, though we weight the two main microphone channels at 0.8 and the other ones at 0.2 in order to better determine the ambisonics of the noise. This models possibly missing channels as a weighted sum of observed data.

Finally we would render the spatial audio field into Unreal, though I was unable to do this in time.

3.4 Other Explorations

To go from a set of images to a gaussian splat is a fast process, and often with enough compute it seems near instant. However our secondary mission was to find any possible methods to optimize the splatting process. The simplified pipeline looks like this: a series of images are send, some preprocessor uses structure from motion (SfM) to produce a sparse point cloud. The images are then used to produce a dense point cloud and finally the dense point cloud undergoes splatting resulting in a completed gaussian splat after X amount of training steps.

However let us take the CSE 493V lecture room and splat it without any people inside. This would involve taking a set of images of the empty classroom (regardless of what device was used for capture it). We go through the splatting pipeline, producing a sparse and dense point cloud of the environment before finally training and obtaining a splat of the empty classroom. We can treat this as a ground truth representation. But what if now we wanted to splat the classroom with people inside? We would still need to collect a set of images (now with people in the classroom). Due to the people in the classroom, images involved may occlude objects like chairs and tables that were already existing. Furthermore, would it not be inefficient to generate a whole new point cloud when the majority of the environment was already preprocessed when empty?

Thus one of our major experiments was to utilize intermediate point clouds (for example a sparse point cloud of the empty room) and use it with images of a non-empty room to reconstruct an environment of the non-empty room. This way we effectively "cache" a sparse cloud to be used for processing down the line, skipping the possibly expensive reprocessing of the sparse point cloud.

Another route we explored was the use of semantic segmentation where we use semantic segmentation on new objects introduced and simply "merge" them into the precomputed dense point cloud of the original. While much more complex with varying levels of success, it does prove to hold some value in the idea of "caching" intermediate steps to be reused in the future.

4 IMPLEMENTATION DETAILS

Put in all the specific implementation details here, including both hardware and software aspects. Even if you didn't build a piece of hardware, make sure to document what hardware you used, including your headset, computing environment, and major software libraries. If you implemented specific hardware devices, describe how you decided on the design parameters for the hardware. For example, if you built a VR headset, you'd apply the equations you previously introduced in Section 3 to decide on the values you used in your construction. If you implemented an algorithm, such as volume rendering, then you'd describe the function implementation details here (e.g., GitHub projects you built on, libraries you used, or specific aspects you found challenging and how you resolved them).

4.1 Hardware Wearable

Our implementation for the hardware wearable was not as straightforward as we originally thought. Using a Raspberry Pi 4 and 3 AverMedia webcams, the physical construction of the wearable was easy. However the software involved was much more complex and difficult to wrangle. I used python and some existing 3rd party libraries like FFMPEG and vidlib4 in order to access each webcam in a sort of round-robin queue and take a picture. Unfortunately due to the age of the Pi and the heavy processing being done, it was not possible to run both the audio and video capture simultaneously. However we were able to take pictures and save them along with processing both channels of the 3 microphones.

4.2 Unreal VR renderer

To achieve immersive visualization of the generated 3D Gaussian Splats, we employed the Unreal Engine, a powerful real-time 3D creation tool. Specifically, we utilized LumaAI in conjunction with the MetaXR plugin to render the splats on the Quest 3 VR headset. LumaAI provided tools or technology that facilitated the efficient and high-fidelity rendering of the complex 3D scenes represented by Gaussian Splats, enabling a more realistic and visually compelling experience within the virtual environment. The MetaXR plugin served as the crucial bridge between our Unreal Engine application and the Quest 3 hardware, ensuring seamless integration and optimal performance on the device. This integration handled essential aspects of VR rendering, such as stereoscopic display, motion tracking, and controller input. To further enhance the sense of presence and immersion for the user, we implemented custom code for user movement within the virtual environment, allowing for natural exploration of the reconstructed scenes.

4.3 Pipeline Improvements

Other explorations we took were involved in finding reuse-ability with the point clouds being produced. Ideas that were explored were: 1. Semantic Segmentation + Point Cloud Merging: here we would segment our output point cloud and simply merge people into an existing cached point cloud and then run the training. While straightforward, we would need to use ICP to align the clouds. We explored many different clustering techniques (thanks to SciKitLearn) clustering on position (x, y, z) and color (r, g, b). While segmentation results were good, we discovered a more effective result that didn't need to utilize merging point clouds.

2. Dynamic Oct-tree Point Clouds: While we did explore the use of oct-trees for dynamic environments that change a lot, we ultimately realized the rabbit hold would lead to a dead end and scrapped the code and idea.

3. Sparse Point Cloud + New Images: Intuitively in our attempt to solve the first exploration (segmentation + merging) we discovered that just utilizing the sparse point cloud of the original with images of the new scene would be just as effective AND efficient. We recorded this result but realized it still was not feasible if we wanted to scale and stitch tens if not hundreds of splats together...

University of Washington
hoangn@cs.washington.edu derekz0@cs.washington.edu rc2002@cs.washington.edu

4. Chunking Technique: We settled on this to explore after the results found about in point 3. Read below for Chunking technique.

4.4 Measurement

To reconstruct the initial point cloud and camera pose estimation of a scene, we utilized RealityCapture. We also implemented a custom editor, leveraging Open3D and RealityCapture export data, to manage scaling, rotation, and translation using GPS data and IMU sensor readings. To calibrate the sensor data, we measured the sensor output for a few frames of our input, selected one of these frames as a reference, and employed our assumptions regarding user height and head rotation. While this complete process was not integrated into our wearable device, we utilized IMU data streamed from our previous homework and estimated GPS (latitude, longitude, and altitude) data using online tools. Video for training was recorded by positioning the capture device in front of the user's head. Our editor automates calculations based on camera pose and sensor data to transform the initial point cloud accordingly, prior to uploading it into our database. Our editor also incorporates functionality for manual editing of the point cloud, enabling us to address minor errors.

4.5 Chunking Database

To implement our caching approach, we construct a global representation of the point cloud that aligns with real-world coordinates. To optimize storage, we store data only for areas likely to be of interest for splatting. Therefore, we employ a database that organizes point clouds based on a chunking technique. Currently, we store each chunk as a 1m x 1m x 1m representation of the recorded scene. Each chunk stores an associated point cloud, which can be updated by user uploads, along with associated camera and image details relevant to that specific chunk. Following transformation using data from our wearable device, the captured data is divided into chunks representing corresponding spatial blocks in the real world. Each chunk is assigned a global coordinate derived from local data collected by our wearable and GPS data. Point clouds are stored as relative distances from a reference point (the bottom-left corner of the chunk), with values ranging from 0 to 1, allowing for optimization techniques such as quantization or sphere mapping [Cigolle et al. 2014].

To associate camera poses with specific chunks, we utilize ray casting. For each estimated camera pose and its corresponding frame image, a ray is cast to determine its intersection with the chunks. We currently set the ray intersection check distance to 5 meters. This parameter can be modified depending on the setting and situation; for instance, outdoor scenes may necessitate a longer ray distance due to the sparser nature of their point clouds.

This database is characterized by write-heavy and read-heavy operations, with limited deletion functionality. Currently, point clouds from users are merged using concatenation and downsampling within each chunk. This process relies on the assumption that our wearable device's measurements are reasonably accurate. Minor discrepancies can be addressed with ICP due to sufficient overlap between point clouds within each chunk. However, this approach is less effective in dynamic scenes with significant movement, such as roads with cars and trucks. It performs adequately for reconstructing smaller objects to be added to a scene, as demonstrated in the table example in 5.2. Future work could incorporate segmentation to store only static objects (e.g., buildings, structures) and potentially employ a frequency count from users to identify objects that remain static over extended periods, leading to a consensus on the point cloud representation of those objects.

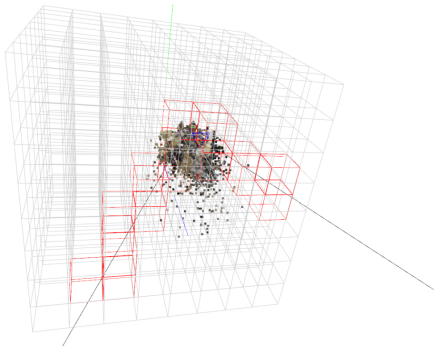

Fig. 6. Chunk intersection determination via ray casting. The figure depicts rays originating from two random camera pose estimations. Chunks identified as intersecting with the rays are marked in red.

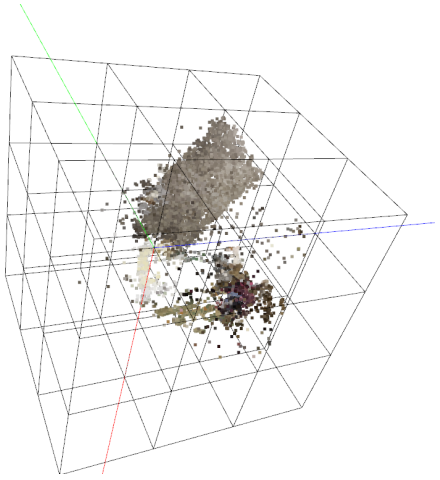

Fig. 7. Result of our database implementation: 3x3 chunk retrieval centered around the origin.

The database query system retrieves a set of images, camera poses, and point clouds based on GPS data (indicating the chunk at the query's center) and a chunk distance (specifying how many neighboring chunks to retrieve). This will create a comprehensive dataset suitable for jump-starting the 3DGS training phase. This approach assumes that users employ our wearable device, resulting in similar frames and rotations, enabling the substitution of image datasets. Our pipeline is designed as follows: Assuming a database with a near-accurate representation of a scene (e.g., a room), constructed

from numerous user contributions, a new user seeking to reconstruct the room would only need to record the relevant portions where detailed accuracy is desired. Our system would then output the relevant point clouds and camera poses for 3DGS training to generate the scene. This provides users with a rapid method for obtaining a 3D representation of their desired capture. Concurrently, our system executes the COLMAP pipeline to reconstruct point clouds and integrate that data, further enhancing the database for subsequent users.

Our results are promising. Using an NVIDIA GeForce RTX 4070 Laptop GPU and Postshot, a complex scene that previously required 201 images for reconstruction can now be constructed with a smaller set of images. New details in the scene emerge in approximately 2-3 minutes.

4.6 Audio

The implementation and research for audio was done in python with significant effort in audio signal preprocessing and synchronization of the six microphones. Since the webcams operate independently, their audio streams must be aligned using cross-correlation techniques. This involves using generalized Cross-Correlation with Phase Transform (GCC-PHAT) which we implemented from scratch in order to do processing on the PI to estimate Time Differences of Arrival (TDoA) between each microphone pair. We had to force the system to continuously buffer audio data in real-time, calculating cross-correlations for overlapping segments, and identifing peaks in the delay estimates (intensity checking). These delays are then used in an Iterative Least-Squares solver selected from scikitLearn, which refines the estimated (x, y, z) position of the sound source relative to the microphone array. I also attempted to run EMA and Kalman filters to filter out significantly noisy inputs but they did not work as well and changes were negligible.

Once the source position is established (as well as it can through the ILS), beamforming techniques reconstruct the spatial audio field by summing the six microphone signals with dynamic delay compensation. The compensation algorithm I took inspiration from was one designed by Wedge in one of their white papers where we conduct the delay compensation on the intensity checks noted from the preprocessing step before. We chose to use the delayed-sum-beamformer in order to align the wavefronts of the source audio based on the computed TDoAs, creating a focused spatial representation of the sound. To transform the audio into Ambisonic B-format, the microphone signals are mapped into First-Order Ambisonic components, ensuring rotational invariance and spatial coherence. If any spatial information is missing due to microphone sparsity, we use plane-wave decomp to interpolate additional signals. These combined approaches generate a 360-degree spatial sound representation that can be processed into multiple audio output formats in the actual spatial audio environment (we were able to add this attenuation into Unreal). We also tried to translate the ambisonic field to HRTF to build the binaural audio but ran into issues with how Unreal XR would project that binaural audio so we hit the roadblock there. Hopefully that will be fixed in future work on this topic.

5 EVALUATION OF RESULTS

5.1 Hardware Wearable

Fig. 8. Our wearable (though we removed it from the hat for easier testing)

Our wearable performed as well as you would expect 3 webcams jerry-rigged to a raspberry pi would work. It was clunky, hacky, and a pain to wear. But to its credit, it worked. We were able to capture images from each camera and use them for splatting. We were also able to use the audio collected to do experiments and testing in order to produce the spatial audio fields. As an MVP it functioned fairly well.

5.2 Improved Pipeline

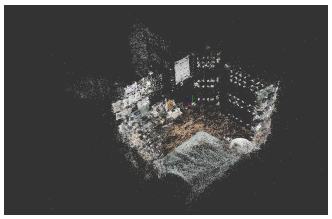

Fig. 9. This is a dense point cloud we use as a "cache" and objects like a draped shirt and a poster are added into it and aligned through ICP. However this proved to be extremely complex and was somewhat inefficient (merging took 1 minute while alignment took 8 minutes)

One of our methods we looked into was semantic segmentation of the point clouds involved with the gaussian splatting. As stated in the section above, we looked through a variety of clustering methods with DBScan and GMMs producing the best results for semantic segmentation. We would then select these clouds and attempt to merge them into another point cloud and train the new splat from there, however we ran into issues where methods of using iterative closes point (ICP) for alignment took extremely long which led us to explore other avenues in the "caching" idea. As referred by the image above, we wanted to see if we could use saved point clouds to save some time recomputing point clouds and use existing ones. Our

University of Washington
hoangn@cs.washington.edu derekz0@cs.washington.edu rc2002@cs.washington.edu

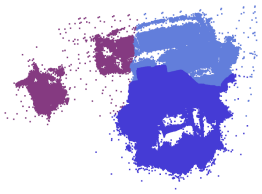

Fig. 10. One of our testing point clouds known as "table", where the use of a GMM classified 3 "parts" of the point cloud. The goal of the test is to see if given a point cloud of an empty table (time t), we could find some way of making a splat with objects on the table (time t+1)

test was with image folders involving an empty table and the same table but with random objects on it. What we ended up discovering was that if we used the sparse or dense point cloud precomputed for the empty table we could use it and some new images of the table with objects to produce a new splat without needing to recompute a new point cloud. The results are showing here: This is a promising

Fig. 11. Dense point cloud associated with a empty table (notice how there are no blue or gray objects on the table's surface)

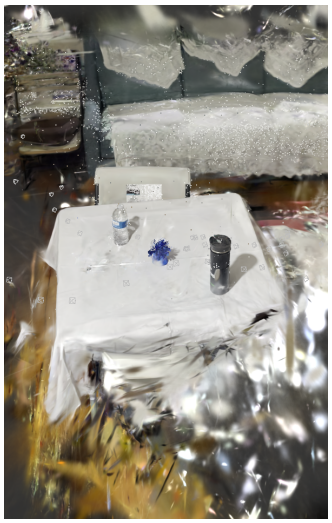

Fig. 12. Resultant splat

result, as it means that for environments subject to lots of change, we can in fact do some minor optimizations to improve the pipeline.

6 DISCUSSION OF BENEFITS AND LIMITATIONS

Honestly, the most promising and novel part of our research was ultimately in our attempts at improving the caching methods involved with computing splats in general. Just like how KV-Caching fundamentally changed the nature of scalable LLM systems, I think some of the results we found by using semantic segmentation of point clouds, point cloud merging, and a sparse point cloud caching are routes to pursue in the future to make splatting global environments easier and more scalable.

Our chunking technique is also a form of caching. By dividing the environment into blocks, we can process and store each block separately. Then, if we only need to render a specific part of the environment, we can just load the corresponding chunks as input to Postshot, rather than the entire environment. This allows us to reuse previously processed data. This approach offers significant advantages in terms of efficiency and scalability. Instead of having to process and store an entire environment as one large dataset, we break it down into smaller, manageable units. Each chunk, representing a 1m x 1m x 1m space, stores associated point clouds, camera information, and image details relevant to that specific block. This means that when a user wants to explore a specific area, we only need to retrieve and render the chunks relevant to that area, effectively "caching" and reusing the pre-processed data and avoiding unnecessary computation or data loading. This is particularly beneficial for large-scale environments, as it allows for selective loading and rendering, reducing the computational load and memory footprint. Furthermore, the chunking approach facilitates collaborative reconstruction, as different users can contribute to the creation of a shared 3D representation by capturing and uploading data for different chunks.

We encountered a number of limitations however; the wearable was clunky, it was hacky, and worst of all it wasn't consistently performant, it was a struggle to implement ambisonic attenuation into Unreal, and some of our attempts at improving the splatting pipeline itself was less than satisfactory. Hardware definitely stands to be a limitation on making splatting dynamically at scale.

7 FUTURE WORK

Our research lays a foundation for several promising avenues of future work, each with the potential to significantly enhance the capture, reconstruction, and immersive experience of 3D environments.

Enhanced Wearable Device: A primary direction involves refining the hardware wearable. Future iterations should focus on:

- **Improved Capture Capabilities:** Integrating higher-resolution cameras with wider fields of view or even 360-degree capture capabilities to minimize occlusion and maximize scene coverage in a single pass. Incorporating depth sensors or LiDAR could further enhance the accuracy and completeness of the captured 3D data.
- **Robustness and Reliability:** Addressing the performance inconsistencies of the current prototype by utilizing more robust hardware components, implementing real-time error detection and correction mechanisms, and optimizing power consumption for extended operation.

Refinements to the Splatting Pipeline: Several improvements can be made to the software pipeline:

- **Advanced Point Cloud Caching and Management:** Expanding upon our initial exploration of point cloud caching. This includes developing more sophisticated strategies for merging point clouds from different captures, handling dynamic scene elements (e.g., moving objects), and implementing efficient data structures for storing and retrieving cached point clouds at various levels of detail. Researching efficient methods to update and refine cached point clouds over time is crucial.
- **Semantic Segmentation Integration:** Fully integrating semantic segmentation into the pipeline to enable more intelligent object-based manipulation and editing of splats. This could involve using segmentation to isolate objects for removal, replacement, or modification, as well as to improve the accuracy of point cloud merging by aligning objects based on their semantic labels.
- **Splat Optimization and Compression:** Investigating techniques to optimize the representation of Gaussian Splats for efficient storage, transmission, and rendering. This could involve exploring methods for splat pruning, adaptive splat density, or compression algorithms specifically designed for Gaussian Splat data.

Scalability and Real-world Deployment: Addressing the challenges of scaling the system for real-world deployment:

- **Distributed Processing:** Investigating the use of distributed computing or cloud-based processing to handle the computational demands of large-scale splatting and storage.
- **Data Management and Storage:** Developing robust and scalable data management systems for storing and retrieving large collections of Gaussian Splats and associated metadata. This includes addressing issues of data redundancy, consistency, and security. * **Collaborative Capture and Reconstruction:** Designing workflows and tools to support collaborative capture and reconstruction of 3D environments, enabling multiple users to contribute to the creation of a shared 3D representation.

By pursuing these future research directions, we can move closer to realizing the vision of capturing and reliving memories in a truly immersive and efficient manner.

8 CONCLUSION

In this project, we explored the pipeline for capturing, reconstructing, and rendering immersive 3D environments using Gaussian Splatting and spatial audio. We developed a hardware wearable for capturing visual and audio data , investigated methods for optimizing the splatting process through point cloud caching and chunking, and explored techniques for spatial audio reconstruction. While our results demonstrate promising directions for future research, particularly in the area of caching for reconstruction, we also encountered limitations in hardware performance and the complexities of integrating various components of the pipeline. Ultimately, our work contributes to the ongoing efforts to create more efficient and scalable methods for capturing and reliving immersive experiences, and suggests that further research into point cloud caching, among other areas, could yield significant advancements in the field.

ACKNOWLEDGMENTS

We would like to acknowledge and thank one Long Nguyenle for his support and work in audio processing.

REFERENCES

Amy Bastine, Lachlan Birnie, Thushara D. Abhayapala, Prasanga Samarasinghe, and Vladimir Tourbabin. 2022. Ambisonics Capture using Microphones on Head-worn Device of Arbitrary Geometry. In *2022 30th European Signal Processing Conference (EUSIPCO)*. 309–313. https://doi.org/10.23919/EUSIPCO55093.2022.9909803

Zina H. Cigolle, Sam Donow, Daniel Evangelakos, Michael Mara, Morgan McGuire, and Quirin Meyer. 2014. A Survey of Efficient Representations for Independent Unit Vectors. *Journal of Computer Graphics Techniques (JCGT)* 3, 2 (17 April 2014), 1–30. http://jcgt.org/published/0003/02/01/

Maximo Cobos, Jens Ahrens, Konrad Kowalczyk, and Archontis Politis. 2022. An overview of machine learning and other data-based methods for spatial audio capture, processing, and reproduction. *EURASIP Journal on Audio, Speech, and Music Processing* 2022, 1 (16 5 2022). https://doi.org/10.1186/s13636-022-00242-x

Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. 2013. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Auton. Robots* 34, 3 (April 2013), 189–206. https://doi.org/10.1007/s10514-012-9321-0

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. arXiv:2308.04079 [cs.GR] https://arxiv.org/abs/2308.04079

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. 2024. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM Transactions on Graphics* 43, 4 (July 2024). https://repo-sam.inria.fr/fungraph/hierarchical-3d-gaussians/

Shoichi Koyama, Enzo De Sena, Prasanga Samarasinghe, Mark R. P. Thomas, and Fabio Antonacci. 2025. Past, Present, and Future of Spatial Audio and Room Acoustics. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5. https://doi.org/10.1109/icassp49660.2025.10890366

Yusen Li, Ying Shen, and Dongqing Wang. 2024. DIFFBAS: An Advanced Binaural Audio Synthesis Model Focusing on Binaural Differences Recovery. *Applied Sciences* 14, 8 (2024). https://doi.org/10.3390/app14083385

George Nousias, Konstantinos Delibasis, and Ilias Maglogiannis. 2023. *H*-RANSAC, an algorithmic variant for Homography image transform from featureless point sets: application to video-based football analytics. arXiv:2310.04912 [cs.CV] https://arxiv.org/abs/2310.04912

Prabin Rath, Alejandro Ramirez-Serrano, and Dilip Pratihar. 2020. Real-time moving object detection and removal from 3D pointcloud data for humanoid navigation in dense GPS-denied environments. *Engineering Reports* 2 (10 2020). https://doi.org/10.1002/eng2.12275

Jakob Vennerød. 2014. *Binaural Reproduction of Higher Order Ambisonics A Real-Time Implementation and Perceptual Improvements*. Ph. D. Dissertation. https://doi.org/10.13140/RG.2.1.4624.4007

Vladimir Yugay, Theo Gevers, and Martin R. Oswald. 2024. MAGiC-SLAM: Multi-Agent Gaussian Globally Consistent SLAM. arXiv:2411.16785 [cs.CV] https://arxiv.org/abs/2411.16785