MULTIPHYSICS SYNTHESIS OF BLAST PHENOMENON

by

Surabhi Parab

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2024

ABSTRACT

SURABHI PARAB. Multiphysics Synthesis of Blast Phenomenon. (Under the direction of DR. ANDREW WILLIS)

This thesis develops an advanced simulation framework within the Gazebo environment to enhance the realism and efficiency of modeling blast phenomena. It tackles the challenge of synchronizing visual, acoustic, and pressure data to simulate explosions accurately. By integrating a suite of technologies-including a robotic operating system (ROS), an unmanned aircraft mission planner (QGroundControl), a vehicle autopilot simulator (PX4 Software In The Loop - SITL), and a flight control neural network training environment (OpenAI GymFC)-the framework facilitates precise emulation of real-time events: visual effects at the speed of light, acoustic propagation through seismic and air mediums, and dynamic pressure variations. The implementation utilizes a client-server architecture, enabling real-time adjustments and reducing latency, which enhances the simulation quality. The research introduces specialized plugins to model and manage different aspects of the blast, demonstrating significant improvement in both fidelity and operational efficiency. These enhancements make the simulation tool a valuable asset for training and analytical applications in safety and defense sectors, providing comprehensive insights into blast dynamics.

Keywords: Blast Simulation, Gazebo Simulation, Robot Operating System (ROS), Acoustic Simulation, Visual Simulation, Visual Rendering, Pressure Dynamics, Computational Fluid Dynamics (CFD), Physics-based

ACKNOWLEDGEMENTS

My Master's Thesis is supported by a Research Assistantship working on a project sponsored by Corvid Technologies, a Teaching Assistantship for the class 'Analytical Foundations of Electrical and Computer Engineering' under Dr. Yawo Amengonu, from the UNC Charlotte Graduate School's Graduate Assistant Support Plan (GASP) and the Electrical and Computer Engineering Department of William States Lee College of Engineering at the University of North Carolina Charlotte.

I would like to begin by expressing my sincere appreciation to my advisor, Dr. Andrew Willis, for believing in my potential, and for his invaluable expertise and guidance throughout my MS in Electrical Engineering studies. I am grateful for the knowledge he imparted and his assistance in the preparation of this thesis. His insightful discussions and direction were essential to the completion of this work.

I also appreciate Dr. James Conrad, a committee member, who has supported me since my first semester of my Master's. His guidance has been invaluable, closely following my progress and challenges throughout my studies. I am also thankful to Dr. Artur Wolek, another committee member, for his constant feedback during my thesis work.

I am deeply thankful to Dr. Jincheng Zhang and Chris Beam, my colleagues in the 'Machine Vision Laboratory'. Their support was essential in helping me adapt to the lab's operational methods and deeply enhanced my grasp of the involved technologies.

Additionally, I am deeply grateful to Dr. Dipankar Maity for his ongoing feedback throughout my thesis, even though he was not on my committee.

Finally, I must acknowledge my mother, Leena Yeshwantrao, for her unwavering support. Her commitment has enabled me to progress from undergraduate studies to a Master's degree and now towards a Ph.D., marking a first in our family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	An acronym for Application Programming Interface
CFD	An acronym for Computational Fluid Dynamics
GANs	An acronym for Generative Adversarial Networks
ROS	An acronym for Robot Operating System
SDF	An acronym for Simulation Description Format
SITL	An acronym for Software In The Loop
TNT	An acronym for Trinitrotoluene
UAV	An acronym for Unmanned Aerial Vehicle

CHAPTER 1: INTRODUCTION

Simulating the intricate dynamics of a blast phenomenon in real time presents formidable computational challenges. Fast time simulation is a computational technique that accelerates the simulation of system dynamics or processes, enabling rapid analysis and decision-making beyond real-time constraints. This research leverages a fast time simulation to a certain extent. A perceptual dataset is specifically designed to synchronize the playback of visual data for a blast. Images are played back at the moment they would be perceived, mimicking the speed of light and capturing the visual essence of the blast. Concurrently, acoustic signals are played back in two stages: first, at the time when pressure waves would travel through the ground at a accelerated rate due to seismic P waves, and subsequently, at the moment they would be heard traveling through the air at the speed of sound. This approach facilitates a more realistic and computationally efficient simulation of blast events, adhering to the underlying physical phenomena they represent. Notably, there is currently no open-source software available to simulate blasts, explosions, and the perceptual, seismic, acoustic and wind/pressure environmental phenomena they generate.

To streamline the execution of simulation-based experiments, our simulation solution integrates four technologies: (1) Gazebo and enhancements [1], [2], (2) the Robot Operating System (ROS) [3], (3) QGroundControl [4] and (4) PX4 Software In The Loop (SITL) firmware [5]. It can be time consuming to issue commands separately to get all applications running and to execute a mission. A script was written that automatically runs a complete flight mission which saves time by not requiring manual intervention to arm, initiate calibration, issue takeoff commands, execute the mission, and send landing commands to the vehicle. Furthermore, this ROS setup is replaced with a new OpenAI GymFC framework [6] [7] [8].

1.1 Problem Statement

In the field of defense simulations, accurately predicting and visualizing the impact of blast phenomena poses significant computational challenges. Many existing models struggle to effectively synchronize visual and acoustic data with the dynamics of real-time physical phenomena, which can render the simulations less realistic and limit their practical application. This research addresses these gaps by developing an integrated simulation framework in Gazebo that enhances the realism and computational efficiency of blast event simulations, even when not processed in real-time. Specifically, the project focuses on leveraging advanced computational models to index and playback pre-staged visual and acoustic data at moments that mimic real-time events. This method ensures more accurate synchronization of sensory data with the physical dynamics of blasts, improving the quality and utility of the simulations for training and analysis purposes.

1.2 Structure of the Thesis

The structure of this thesis is organized into six chapters, each addressing distinct aspects of the simulation of blast phenomena. Below is an overview of each chapter, outlining the main focus and contributions of the research presented in this thesis.

Chapter 1 introduces the challenges of simulating blast phenomena with high realism and computational efficiency. It outlines the research objectives, emphasizing the need for improved synchronization of visual, acoustic, and pressure data within simulations. The chapter provides an overview of the thesis structure, setting the stage for detailed discussions in subsequent chapters.

Chapter 2 reviews existing literature related to blast simulation, identifying gaps in visual and acoustic simulation technologies and dynamic physical modeling. It contextualizes the current research within the broader field, highlighting how this work builds upon and enhances existing methodologies.

Chapter 3 is the Methodology chapter which is the core of the thesis, where the development and integration of the simulation framework are comprehensively described. It is divided into subsections that detail the design, implementation, and function of each plugin developed for the Gazebo simulator:

Camera Plugin: Explains how visual simulations of explosions are captured and synchronized. Microphone Plugin: Details the methods for simulating and synchronizing blast-related acoustic signals. Model Plugin: Describes the simulation of pressure and environmental impacts from blasts. World Plugin: Discusses the integration of all simulations to produce a cohesive and dynamic output.

This chapter not only elaborates on the technical aspects but also explains the rationale behind various design choices, underscoring their relevance to the research goals.

Chapter 4, the Results chapter, evaluates the simulation framework, presenting results from various test scenarios to demonstrate improvements in accuracy and efficiency. It assesses the performance of individual plugins and the integrated system, providing evidence to support the research hypothesis.

Chapter 5 discusses future enhancements and broader applications of the simulation framework. It suggests potential for advanced detection algorithms, integration of machine learning, and application extensions to other interdisciplinary fields, emphasizing the framework's adaptability and scalability.

Chapter 6 consolidates the key findings and contributions of the thesis. It briefly highlights the potential applications and future research directions to further improve and expand the simulation framework.

CHAPTER 2: RELATED WORKS

Contributions to multiphysics simulations include studies on interactive environments for household tasks by Fu et al. [9], accidental fire and explosion simulations by Henderson et al. [10], fluid dynamics courses by Bridson and Müller-Fischer [11], large scale smoke phenomena by Rasmussen et al. [12], and liquid splash modeling using neural networks by Um et al. [13].

Preliminary reviews that offered insights into the development of the acoustic models for this research included innovative approaches to earthquake detection using deep learning with fiber-optic distributed acoustic sensing by Hernandez et al. [14], advancements in earthquake detection at geothermal fields through distributed acoustic sensing and template matching by Li and Zhan [15], and the novel detection of earthquakes from balloons utilizing their acoustic signatures by Brissaud et al. [16]. These studies, while not directly aligned with the core objectives of this thesis, provided valuable contextual understanding and contributed to the foundational knowledge base.

The paper "Explosion Simulation Using Compressible Fluids" [17] is closely related to my work, as it explores the physical-based simulation of explosions, an approach that mirrors the methodologies employed in my research to simulate blast phenomena in Gazebo. By exploring the use of compressible Navier-Stokes equations for simulating the dynamics of explosions, this paper provides valuable insights into accurately modeling the complex interactions of gases and pressure waves during a blast. These methods can improve the accuracy and realism of my blast models, especially when it comes to modeling how explosions affect the surrounding area. This is in line with the goal of my project, which is to include true physical events into virtual simulations.

Karl Sims' seminal work on "Particle Animation and Rendering Using Data Parallel Computation" (1990) [18] offers critical methodologies that can be adapted for enhancing blast simulations within the Gazebo framework used in our project. Sims demonstrates the use of data parallel supercomputers to animate and render particle systems, covering a spectrum from kinematic motions to physically based simulations. This approach is particularly relevant for our simulation of dynamic blast phenomena, where rendering complex particle interactions such as debris and smoke with high fidelity is crucial. Additionally, Sims' implementation of advanced rendering features, including antialiasing and motion blur, can significantly improve the visual realism of our simulations. These techniques ensure not only enhanced visual accuracy but also improved computational efficiency, making them invaluable for real-time applications in safety and defense training simulations. Incorporating these data parallel computation strategies aligns with our objectives to achieve more realistic and computationally efficient simulations, highlighting the ongoing evolution of simulation technology in response to complex real-world phenomena.

The research presented by Martins, Buchanan, and Amanatides in "Visually Believable Explosions in Real Time" [19] primarily focuses on the graphical rendering of explosions to achieve real-time visual believability for interactive applications like video games. In contrast, my project extends beyond visual effects to include acoustic modeling, aiming for a comprehensive simulation of blast phenomena in Gazebo that integrates both visual and acoustic data. While Martins et al. optimize for immediate visual feedback and interaction, emphasizing speed and visual appeal, my work prioritizes physical accuracy and detailed simulation of sound and shock wave propagation. This approach not only enhances realism but also provides a more scientifically robust tool for applications in safety training and analysis, where accurate replication of physical phenomena is crucial. Thus, although both projects deal with the simulation of explosions, mine focuses on a more detailed and accurate representation, particularly suited for educational and analytical purposes in safety and defense sectors.

The paper "Physics Based Real-Time Explosion Simulation" (2012) [20] offers insights into real-time physics-based simulation techniques, which contrast with the methodologies used in this thesis. While the paper focuses on real-time dynamics for digital home applications, this research develops a high-fidelity simulation framework in Gazebo that is not processed in real-time but instead uses pre-staged data for visual and acoustic synchronization. This approach allows for detailed, accurate simulations ideal for analysis and training within safety and defense sectors, without the computational overhead demanded by real-time processing.

The study detailed in "Animating Explosions" by Yngve et al. (2000) [21] presents a computational fluid dynamics approach primarily aimed at generating visually compelling explosion effects for the entertainment industry. While Yngve et al. emphasize the visual aspects of explosions, utilizing computational models to simulate shock wave propagation and interactions with objects, their approach lacks the integration of acoustic effects and the precise synchronization of visual, acoustic, and pressure data that are critical to this research. The methodology developed in this thesis not only seeks to achieve visual realism but also extends to practical training applications by incorporating a multidimensional data integration framework that enhances both the fidelity and utility of simulations for educational and analytical purposes in safety-critical settings. This distinction underscores the broader applicative potential of the simulation framework developed herein, contrasting with the aesthetic focus of the referenced work.

The paper "Animating Suspended Particle Explosions" by Feldman, O'Brien, and Arikan [22] introduces a novel method for animating explosions, focusing on visible effects like fire and smoke rather than the blast wave itself. Utilizing an incompressible fluid model, it adjusts fluid divergence to simulate the expansion of combustion gases and movement of particulate matter. This approach is significant to this thesis as it offers a complementary perspective on explosion simulations by emphasizing secondary visible effects, enhancing the scope and realism of explosion modeling in computer graphics.

The paper "Fluid Animation with Dynamic Meshes" by Klingner et al. [23] presents a method for fluid simulation that uses dynamically changing unstructured tetrahedral meshes. This approach adapts the mesh to fluid boundaries and areas of high interest, which parallels this thesis's focus on high-fidelity blast phenomena simulations.

The research in the paper, "A Vortex Particle Method for Smoke, Water and Explosions" [24], uses a hybrid method combining vortex particle and grid-based techniques, similar to my integration of multiple simulation plugins. However, this paper primarily enhances visual realism, while the thesis research also integrates acoustic and pressure data for a comprehensive simulation approach.

The paper "Visual Simulation of Smoke" by Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen [25] proposes an innovative method for simulating smoke in computer graphics, utilizing the inviscid Euler equations and a novel vorticity confinement approach. Their technique efficiently handles smoke's interactions with dynamic objects and enhances realism on coarse computational grids, making it suitable for real-time graphics applications. This research is relevant to my thesis in the context of optimizing fluid simulations for enhanced visual and computational efficiency. Another similar study is mentioned in [26].

The paper "Blast wave kinematics: theory, experiments, and applications" [27] offers a comprehensive analysis of the propagation dynamics of blast waves from various types of explosions. The authors develop a theoretical framework that extends the standard strong-shock solutions to describe the decay of blast waves into acoustic waves. This is accomplished by introducing dimensionless coordinates that facilitate the direct comparison and visualization of data from different explosions. Through experiments and theoretical modeling, the paper outlines the derivation of general expressions for the Mach number of the shock front, providing essential insights for applications in blast engineering and explosion analysis. These methodologies enhance understanding of the critical parameters that govern the behavior of blast waves, contributing significantly to the fields of safety, military, and engineering.

CHAPTER 3: METHODOLOGY

3.1 Overview

This research utilizes an integrated simulation framework integrating both traditional and innovative technologies to enhance the analysis of blast phenomena. The core framework comprises four primary technologies: Gazebo with specific enhancements, the Robot Operating System (ROS), QGroundControl, and PX4 Software In The Loop (SITL) firmware. Automation scripts facilitate seamless execution of full flight missions, reducing manual interventions and improving operational efficiency.

Gazebo allows users to choose between four different physics engines: (1) the Open Dynamics Engine (ODE) [28], (2) Bullet [29], (3) the Dynamic Animation and Robotics Toolkit (DART) [30], from Georgia Tech, and (4) Simbody [31], from Stanford University. Configuration of the engine including the maximum allowable time step for integrating the environmental dynamics is provided in a Gazebo world file.

Transitioning from the conventional ROS framework to the OpenAI GymFC framework offers improved control and configurability within the simulation environment. The GymFC framework is a critical enabling technology for RL and control tasks as it provides a very high speed interface directly to the simulator physics engine. This allows rapid experimentation for control algorithm development but is an absolute requirement for satisfying the *episode* data generation needs of Reinforcement Learning control algorithm training. A study of different frameworks has been discussed in the paper: [32].

The methodology is structured around four specialized plugins:

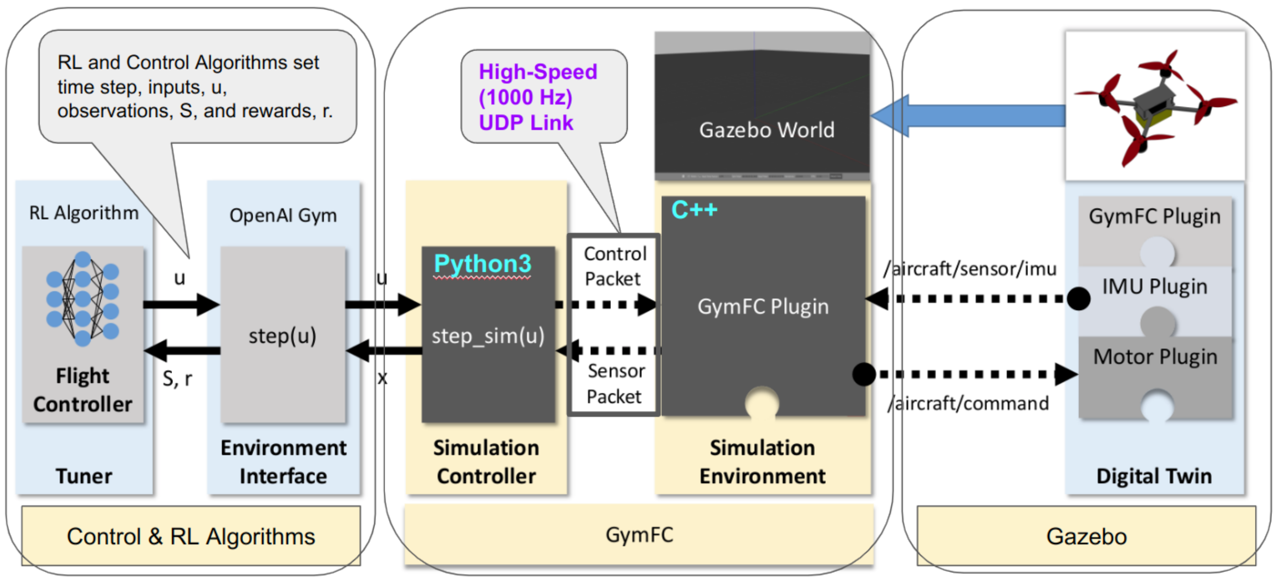

Figure 3.1: shows a flow diagram of the simulation framework

1. **Camera Plugin:** This plugin ensures accurate capture of the visual aspects of the blast, synchronizing visual data with real-time phenomena.

2. **Microphone Plugin:** It processes the acoustic signals from the blast, aligning auditory outputs with corresponding visual simulations.

3. **Model Plugin:** This component models the environmental impacts, specifically simulating the dynamics of pressure and forces within the wind caused by the explosion.

4. **World Plugin:** Serving as the integration hub, this plugin consolidates inputs from other plugins, ensuring cohesive and comprehensive simulation outputs.

Employing a client-server architecture, the world plugin functions as the server, with the other plugins acting as clients. This setup allows for the pre-staging of blast data by communicating blast events to clients well before the actual simulation time. This approach facilitates the playback of impulse response data at a simulation-configured rate of 50-100 Hz, crucial for managing local storage within the simulator for force generation and sensor telemetry generation plugins. This strategy significantly reduces latency in communication within the Gazebo simulation for short-duration events.

By integrating these plugins and utilizing a client-server architecture, the blast simulation project achieves a high level of realism and accuracy, enhancing our ability to understand and investigate blast phenomena thoroughly.

3.2 Camera Plugin

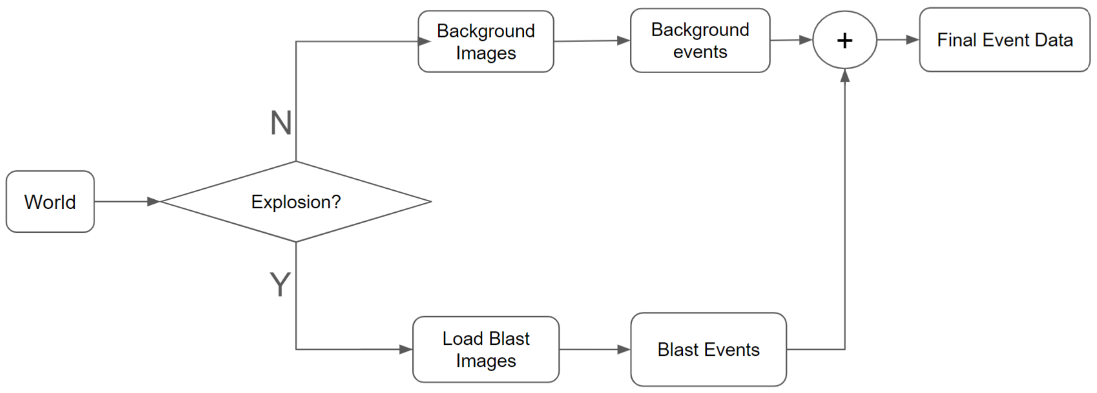

Figure 3.2: shows a flow diagram of the Camera plugin.

The camera plugin, a crucial component of the blast simulation project, is designed to capture and process the visual aspects of blast events. Implemented as a Gazebo sensor plugin, it interfaces with the simulation environment to provide realistic and synchronized visual data. Upon initialization, the plugin configures its parameters, such as image width, height, depth, and format, based on the specifications of the attached camera sensor. It utilizes the Gazebo API to access camera properties and subscribe to image frames. The plugin employs computer vision techniques, leveraging OpenCV for image processing tasks. It calculates the differences between consecutive frames to detect and highlight significant changes, simulating the dynamic visual impact of a blast. This is done with the help of an event camera [33] [34]. Furthermore, the plugin supports the integration of blast-specific visual effects. It can read and process a sequence of pre-generated images representing different stages of a blast, blending them with the real-time simulation to enhance realism. This feature allows for the depiction of complex blast phenomena, such as fire and post-blast smoke clouds, which are challenging to simulate in real-time. To facilitate data exchange and integration with other components of the simulation, the plugin publishes processed image frames and associated event data to designated topics within the Gazebo transport system. This ensures seamless communication and synchronization with other plugins, such as the microphone and model plugins, enabling a comprehensive and coherent simulation of blast events. In summary, the camera plugin plays a pivotal role in the blast simulation project by providing a dynamic visual representation of blast events. Its capabilities extend beyond simple image capture, incorporating advanced image processing and integration techniques to simulate the complex visual effects associated with blasts.

3.2.1 Perceptual Modeling of Blast Events

A model was developed to generate perceptual blast data within the Gazebo environment. This developed model is integrated as a component to the blast simulation Gazebo plugin.

3.2.2 Rendering a Blast Image

This study employs advanced perceptual simulation techniques to accurately depict blast phenomena. The simulation uses state-of-the-art open-source computer graphics rendering methods with Computational Fluid Dynamics (CFD) solvers to achieve realistic visualizations of explosions.

The core of our simulation framework utilizes the Mantaflow CFD solver, integrated within the Blender computer graphics software to simulate explosions and capture images from these simulations, which forms the basis for creating a comprehensive 3D blast image dataset.

Mantaflow, an open-source framework initiated in 2009 at the ETH Zürich Computer Graphics Laboratory and currently maintained by the Thuerey group at the Technical University of Munich (TUM), is utilized for its fluid simulation capabilities. It features a parallelized C++ solver core and a Python scene definition interface, enabling rapid prototyping of complex fluid dynamics scenarios. Mantaflow supports a range of Navier-Stokes solver variants and has been enhanced in recent research, including a 2018 study utilizing Generative Adversarial Networks (GANs) to accelerate CFD simulations [35]. The latest version of Blender (v4.1), incorporating the Mantaflow component, was compiled into a Python module to facilitate automated, script-driven simulation of explosions.

In the simulated environment, explosions are visualized using a nominal camera model, capturing images from all possible camera angles. This method ensures the creation of a detailed 3D blast image dataset. The captured images are indexed based on the sensor camera's intrinsic parameters and the relative pose between the blast and sensor. These images are then scaled according to the sensor's range, bearing, and camera parameters, which ensures that the simulation accurately represents the observed phenomena from various perspectives.

To optimize the simulation process, a Python script was developed to automate the generation of explosion scenarios. This script leverages the integrated capabilities of Mantaflow for simulating the fluid dynamics of explosions and Blender for rendering image sequences from multiple views of the explosion. The automation significantly enhances the efficiency of generating diverse and realistic visual representations of blast events, facilitating more comprehensive research and analysis.

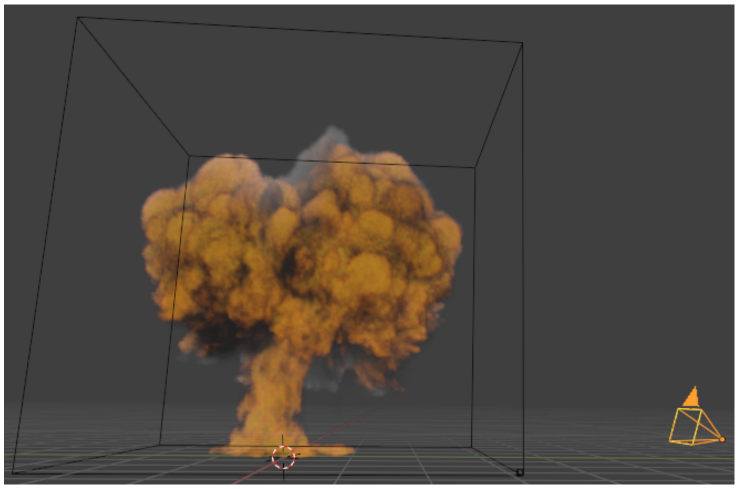

Figure 3.3: shows a blast generated using Blender

Figure 3.3 shows an example of one view of a script-generated explosion with the viewing camera frustum shown (in orange) at the bottom right corner. Blast images are generated having transparent backgrounds to allow the image data to be seamlessly inserted into observed images of the Gazebo environment using conventional cameras and event camera sensors [33]. Image sequences were taken at discrete locations on the upper hemisphere at a fixed range that allows the explosion image to entirely fall within the camera frame. During simulation, frames will be selected from the dataset using the sphere-vehicle relative geometry. The projection of the vehicle onto the unit sphere centered on the explosion will determine the spherical (θ, ϕ) coordinate from which to extract the image. The vehicle range and camera sensor orientation will determine the scale and projection location of the explosion into the image sensor. Sensor locations where the explosion has a non-zero value will be merged or occluded by the computed explosion pixel data.

3.2.3 Creating a Perceptual Dataset

A blast image dataset was generated by creating a script that uses Blender Python packages. This script leverages Blender, a 3D computer graphics software tool, to produce images of a blast from multiple camera viewpoints. A simplistic blast scene was developed using the Blender tool, featuring a blast captured from a single front-view camera [36]. The blast is designed with a transparent background, allowing it to be seamlessly superimposed onto various environments, such as Gazebo worlds.

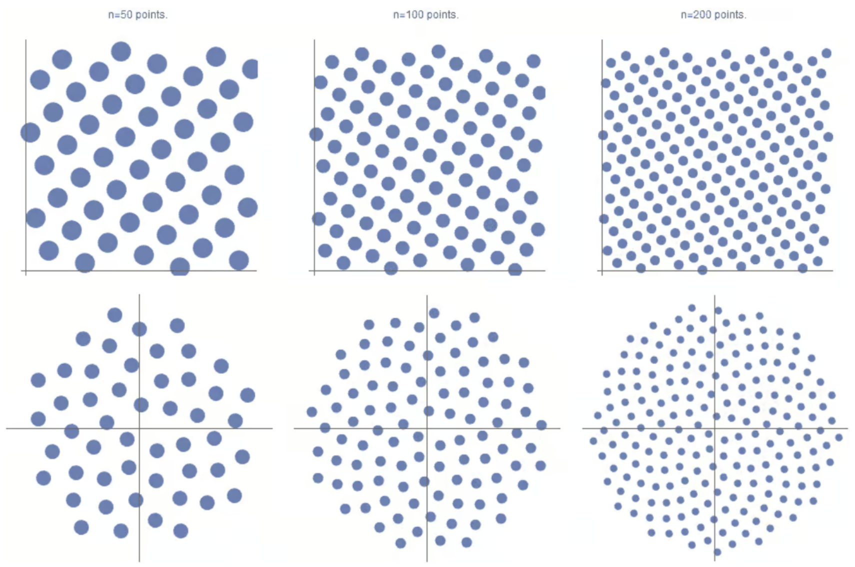

Figure 3.4: shows the Fibonacci lattice (top row) and the Fibonacci spiral (bottom row) for various values of n

Figure 3.4 shows the viewpoint sampling strategy adopted to uniformly sample explosion image sequences across the surface of the upper hemisphere. Conventional spherical coordinates yield highly asymmetric surface density samplings which would incur more error at elevations close to the equator and less at other elevations. To address anisotropic behavior due to sampling, a Fibonacci sphere sampling pattern is adopted which approximately solves the problem of computing an equidistant sampling across a spherical surface. The Fibonacci sphere sampling strategy combines number theory associated with the Fibonacci sequence and its extension to higher dimensions, referred to as the Fibonacci lattice. The Fibonacci lattice sampling approach is then merged with the numerical pattern given by the Golden Spiral, a logarithmic spiral that grows by a factor of the golden ratio for every quarter turn it makes, to generate the canonical implementation of the Fibonacci sphere sampling pattern. Recent refinements for this approach have been incorporated that offset the points slightly away from the poles resulting in improvement in the sampling configuration by up to 8% in terms of packing distance, as demonstrated by Hardin et al. [37] and discussed in other works[38]

3.2.4 Algorithm for the Camera Plugin

As shown in Algorithm 1, the GazeboBlast3DCameraPlugin is designed to integrate with a camera sensor within the Gazebo simulation environment to process and respond to simulated blast effects visually. Upon loading, the plugin initializes and configures itself based on parameters specified in the SDF (Simulation Description Format). It then connects to the camera, retrieves frame dimensions, and sets up necessary publishers and subscribers for communication. During each simulation update, the plugin checks for new frames from the camera. If a blast event is active, it blends blast images into the current camera frame to simulate visual effects of explosions, such as changes in visibility or direct impact visualization. The algorithm effectively handles the reception of new frames, processes them by potentially overlaying blast effects, and then outputs the modified frames for further use in the simulation or for visualization purposes.

3.3 Microphone Plugin

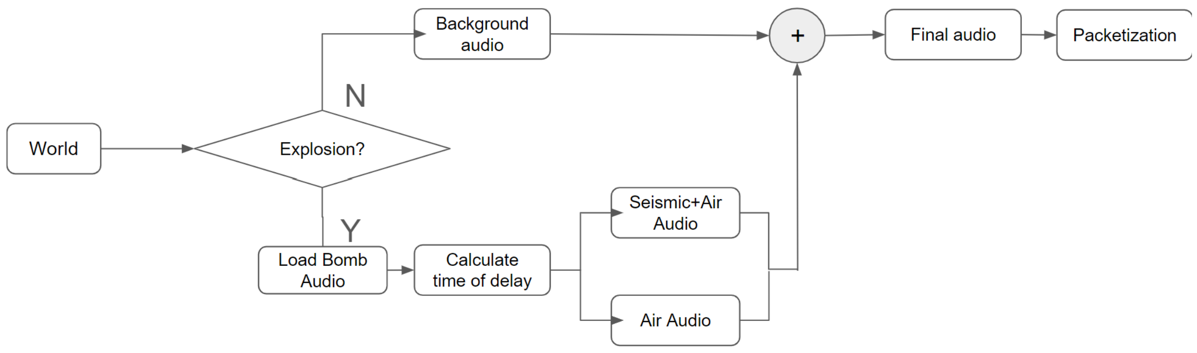

Figure 3.5: shows a flow diagram of the Microphone plugin

Algorithm 1 Load and operate the GazeboBlast3DCameraPlugin

```
 1: procedure Load(sensor, sdf)
 2:     if sensor is NULL then
 3:         print "Invalid sensor pointer."
 4:         return
 5:     end if
 6:     Initialize sensor and world references
 7:     Read parameters from sdf
 8:     Setup camera properties: width, height, depth
 9:     Configure topic subscriptions and publications
10:     Load and configure blast images if applicable
11: end procedure

12: procedure OnUpdate
13:     Get current time
14:     if interval elapsed then
15:         Capture image from camera
16:         Process and send image based on mode
17:     end if
18: end procedure

19: procedure OnNewFrameCamera(image)
20:     Convert image to appropriate format
21:     if explosion triggered then
22:         Blend explosion imagery into current frame
23:     end if
24:     Publish modified image
25: end procedure

26: procedure BlendEventOutput(roll)
27:     Perform event-driven image blending
28:     Update event list based on image differences
29:     Publish event messages
30: end procedure

31: procedure PublishRGBMessage(image)
32:     Format and publish RGB image data
33: end procedure
```

The microphone plugin is designed to capture and process the acoustic signals associated with the blast scenario. As a Gazebo model plugin, it is responsible for simulating the auditory experience of a blast, including the initial explosion sound and subsequent environmental effects. This plugin simulates the propagation of sound waves from the blast source to the microphone sensor. It considers both the direct air path and the seismic path, accounting for the differences in propagation speed and attenuation. The plugin uses a simple model to calculate the time delays and amplitude reductions for the sound reaching the microphone through these paths. To enhance realism, the microphone plugin uses pre-recorded audio files for the background environment and the explosion sound. These audio files are loaded at the start of the simulation and played back as needed. The background audio is continuously looped to provide a consistent ambient sound, while the explosion audio is triggered by specific events in the simulation, such as the detection of an explosion. Furthermore, the plugin also includes low-pass filtering for the seismic component of the explosion sound, simulating the effect of the ground absorbing higher frequencies. This filtered seismic audio is then combined with the direct air blast sound to create a composite audio signal that is published to a designated topic within the Gazebo transport system. In summary, the microphone plugin provides a realistic auditory simulation of blast events, complementing the visual simulation provided by the camera plugin. It captures the complex acoustic phenomena associated with explosions, including the direct blast sound, ground-transmitted vibrations, and ambient environmental noise. This comprehensive auditory simulation enhances the overall realism of the blast simulation project and aids in the analysis and understanding of blast effects.

In the development of the microphone plugin for the blast simulation project, an effective approach to audio streaming is employed, mirroring the continuous yet discrete nature of the universe's ambient sound. This method involves the segmentation of audio data into small, manageable packets, which are then transmitted sequentially to simulate a continuous audio stream. The determination of an optimal packet size is critical for efficient data transmission and processing, to accommodate the dynamic nature of the simulation environment. The plugin is designed to operate at an update rate of 100 Hz, with an audio sample rate of 200 Hz, ensuring that at least two samples are included in each packet. This configuration provides a simulation experience that closely mimics real-world audio streaming services, which utilize buffers at both ends of the transmission to manage latency and ensure smooth playback. In the context of the blast simulation, the use of a buffer system is instrumental in detecting and compensating for latency, thereby maintaining the temporal integrity of the audio stream and enhancing the overall realism of the simulation.

3.3.1 Acoustic Modeling of Blasts

The audio data model consists of two acoustic energy sources: (1) ambient acoustics and (2) blast acoustics. Ambient acoustics sources persist across all time and are intended to mimic the acoustic signals generated within the given simulation environment in the absence of a blast.

3.3.1.1 Ambient Acoustic Model

In the design of the microphone plugin for this project, consideration is given to the natural attenuation of background ambient sound with respect to height or altitude, specifically to support the development of low-latency acoustic blast detection algorithms. This phenomenon is modeled using a circular buffer that dynamically adjusts the intensity of the background noise relative to the ground plane. As a result, the simulated auditory experience accurately reflects real-world behavior, where ambient sound becomes less pronounced as a UAV ascends, and conversely, more pronounced as it descends closer to the ground. This approach not only ensures a realistic representation of the acoustic environment but also enhances the overall immersion and accuracy of the simulation, crucial for effective blast detection in varied operational scenarios.

3.3.1.2 Blast Acoustic Model

Blast acoustics seek to capture multi-modal acoustic realizations of blasts in the environment. As shown in figure 3.6, acoustic energy resulting from blasts propagate to scene objects using two models: (1) free space propagation and (1) seismic/free-space propagation. In free space propagation, the sound waves travel through the air, while in seismic space propagation, the sound waves travel through solid mediums such as the ground. Due to the higher speed of sound in solids, seismic space propagation allows the sound waves to travel faster through the soil compared to free space propagation. This dual-mode propagation model ensures a comprehensive simulation of the acoustic signature of a blast, capturing both the aerial and through-ground transmitted components which offer low-latency opportunities for detection for downstream blast control algorithms.

In the blast acoustic model, the attenuation of the acoustic signal is governed by Stokes' law [39] (see Eqn. 3.1), which defines the rate of sound attenuation [40], α, based on the dynamic viscosity (η), density (ρ) of the medium, the speed of sound (V), and the angular frequency of the wave (ω). In Eqn. 3.2, the signal's amplitude at a specific observation point in space, $A(\mathbf{p}_s)$, is determined using an exponential decay function, where the initial amplitude (A_0) decreases as the sound travels through the medium. The decay is influenced by the distance between the source ($\mathbf{p}_b$), where the blast originates, and the observation point ($\mathbf{p}_s$), where the amplitude is measured. This distance, denoted as $\|\mathbf{p}_s - \mathbf{p}_b\|$, crucially impacts how much the amplitude diminishes, encapsulating the attenuation effects over spatial extents. This comprehensive model ensures the accurate simulation of acoustic energy from blasts, capturing how waves propagate through both air and solid mediums. The model thus offers reliable low-latency detection opportunities for downstream algorithms, facilitating enhanced responsiveness in acoustic sensing applications.

$$\alpha = \frac{2\eta\omega^2}{3\rho V^3} \tag{3.1}$$

$$A(\mathbf{p}_s) = A_0 e^{-\alpha||\mathbf{p}_s - \mathbf{p}_b||} \tag{3.2}$$

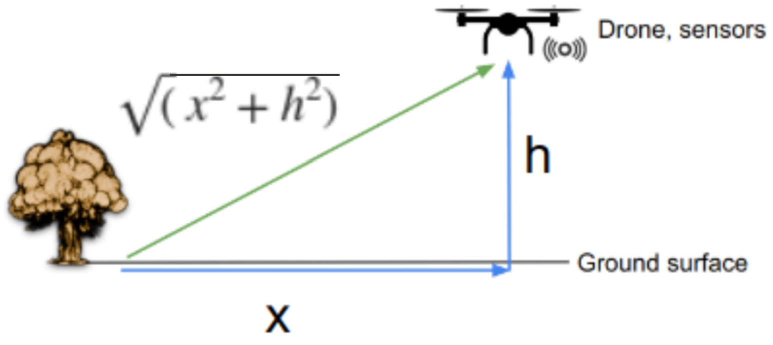

Figure 3.6: shows sound propagation after a blast

Figure 3.6 illustrates the dual-pathway propagation of sound from the epicenter of a ground-level blast to the sensors aboard on a drone. The diagram presents two distinct trajectories for the acoustic energy emanating from the blast. The hypotenuse $\left(\sqrt{x^2+h^2}\right)$ represents the direct path through which sound waves travel through the air (free space propagation), moving diagonally from the blast location to the drone. This airborne route reflects the typical propagation of sound through free space, where the waves spread out spherically from the source, gradually attenuating with distance and experiencing minimal dispersion in the absence of obstructions. In parallel, a second pathway is characterized by seismic propagation, where sound energy initially travels horizontally (ground propagation), represented as x, through the soil - a medium through which sound travels faster due to its higher density compared to air. Upon reaching the vertical alignment with the drone, the seismic waves transition to the vertical free space propagation, represented as h, ascending through the air column to reach the drone's sensors. This ground-to-air route is significant as it captures the transmission of vibrations from solid ground to the drone, providing a unique acoustic signature compared to the direct air path.

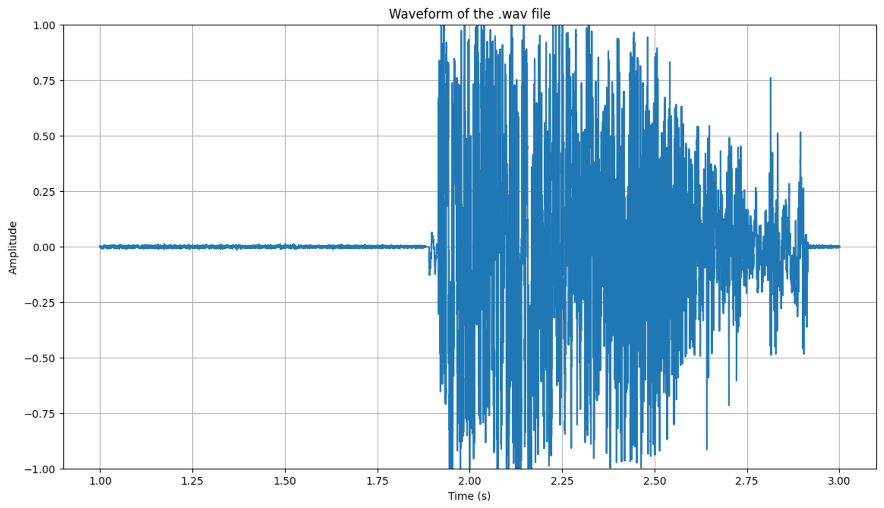

Figure 3.7: shows a recorded audio signal post packetization

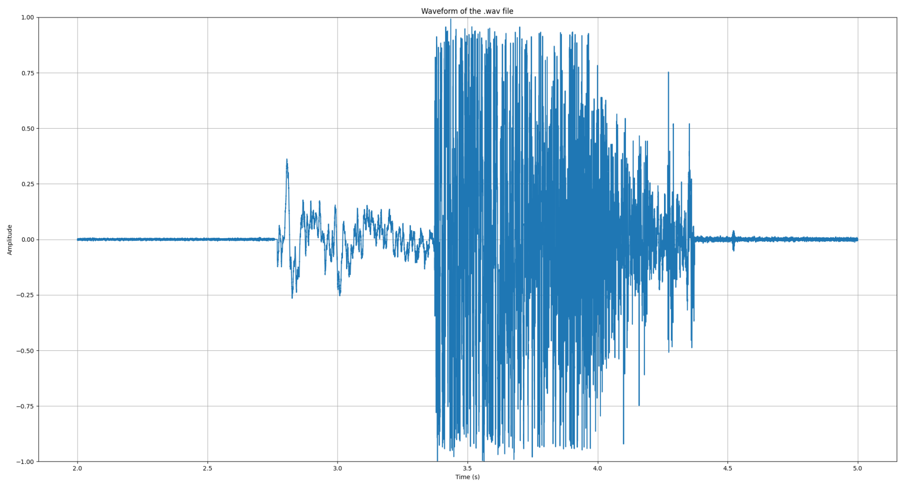

Figure 3.8: shows a recorded audio signal post packetization, where a 0.6 second delay is introduced to the air signal to clearly distinguish it from the seismic signal, for debugging

The arrival times of shock wave propagation in the air and seismic surface waves were estimated using the empirical models proposed in Wu and Hao (2005) [41]. For explosions on rock surfaces, the models are scaled based on the TNT equivalent charge weight (Q in kilograms) and the distance (R in meters) from the blast center, employing dimensionless scaling to standardize results across varying conditions. The seismic wave arrival time at a ground surface point is given by Eqn. 3.3.

$$t_s = \frac{0.91R^{1.03}Q^{-0.02}}{c_s} \tag{3.3}$$

where c_s is the P-wave velocity of the granite mass. The average P-wave velocity of an intact rock core is around 6000 m/s and the average in-situ P-wave is around 5660 m/s.

The arrival time of the shock wave propagation in the air is expressed in Eqn. 3.4.

$$t_a = \frac{0.34R^{1.4}Q^{-0.2}}{c_a} \tag{3.4}$$

where c_a is 340 m/s, the sound speed in dry air at 20 $°C$. This formula incorporates the effects of both the explosive charge size and the propagation distance on the shock wave's travel time through air.

Free Space Propagation In this mode, the sound waves from the explosion propagate through the air, encountering attenuation and dispersion effects that are dependent on the distance from the source and atmospheric conditions.

Seismic Space Propagation This mode simulates sound transmission through the ground [42]. Seismic waves, due to the denser medium of the earth, travel significantly faster than air-propagated waves and interact differently with the environment. The simulation models how seismic waves interact with the surface and couple with air-propagated sound waves, providing a comprehensive depiction of how blasts are heard and felt.

The model acknowledges the differing propagation speeds and attenuation characteristics of these pathways, contributing to the temporal and intensity profiles of the sound as detected by the drone. This dual-pathway approach to acoustic simulation enables a more nuanced understanding of the blast phenomenon's auditory footprint, integral for applications in surveillance, damage assessment, and drone navigation in complex acoustic environments.

3.3.2 Audio Packetization and Streaming

The microphone plugin employs an audio packetization and streaming approach, ensuring a seamless auditory experience within the simulation. This process involves dividing the continuous audio data into discrete packets, which are then transmitted sequentially to simulate a persistent audio stream.

3.3.2.1 Background Ambient Audio Looping:

The ambient background audio, typically of a finite duration, is looped multiple times to extend its playback for the entirety of the simulation. For example, a 10-second background audio looped three times results in a continuous 30-second audio stream.

3.3.2.2 Transmitting Scenario

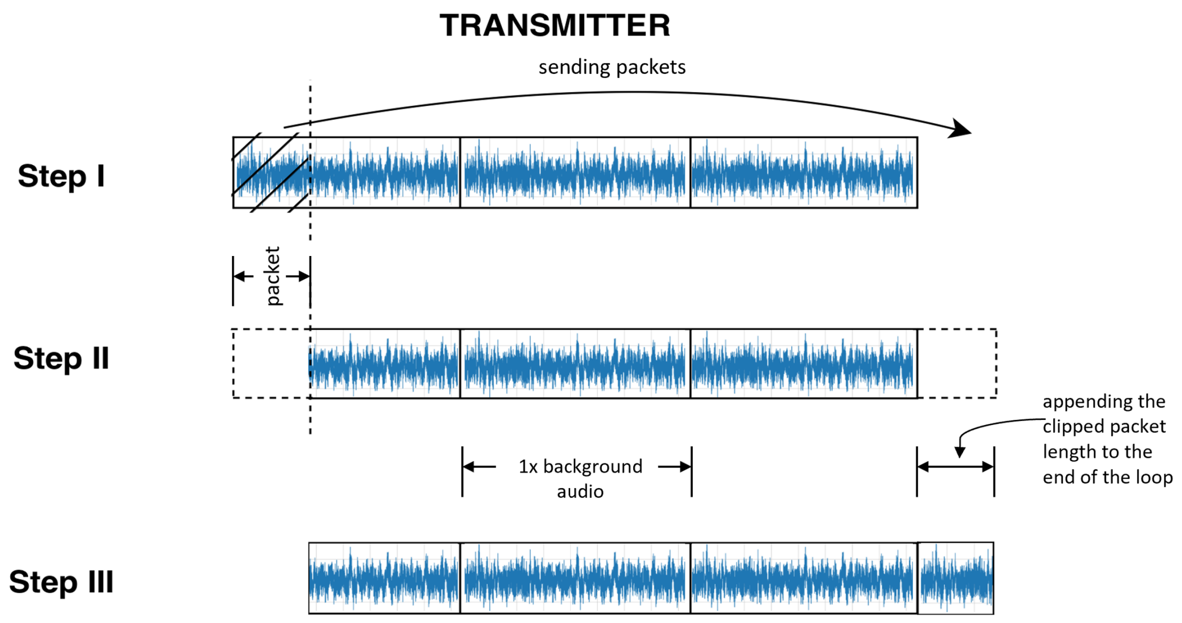

Figure 3.9: demonstrates the three steps of transmitting the background audio signal through packetization

Figure 3.9 illustrates the transmission of the background audio data in the microphone plugin, which is managed through a systematic packetization process, and can be delineated into the following three distinct steps:

Step 1: Packet Size Determination

The initial step involves determining the optimal size of the audio packet to be transmitted. This size is crucial for ensuring efficient data transmission while maintaining audio continuity. For instance, if the packet size is set to 30% of the first loop of the background audio, this fraction of audio data is prepared for transmission.

Step 2: Packet Transmission

Subsequently, the determined packet of background audio is transmitted. Concurrently, an equivalent portion of audio data, corresponding to the packet size, is clipped from the beginning of the original background audio loop. This process ensures that the transmitted audio packet is seamlessly integrated into the ongoing simulation.

Step 3: Packet Appending

The clipped audio packet is then appended to the end of the background audio loop. This action effectively recycles the transmitted audio data, maintaining the continuity of the background audio stream.

Conditional Appending

To prevent excessive elongation of the audio signal, a conditional check is implemented. The appending of packets is allowed only if the total size of the signal remains below a predefined threshold, typically three times the length of the original background audio loop. This condition ensures that the audio stream remains manageable and aligns with the temporal dynamics of the simulation.

Blast Audio Integration: The blast audio, representing the sudden explosion sound, is integrated into the background audio stream at specific time intervals, depending on the simulated blast event. The integration point, termed as end_index, marks the junction where the blast audio is appended to the background stream.

3.3.2.3 Receiving Scenario

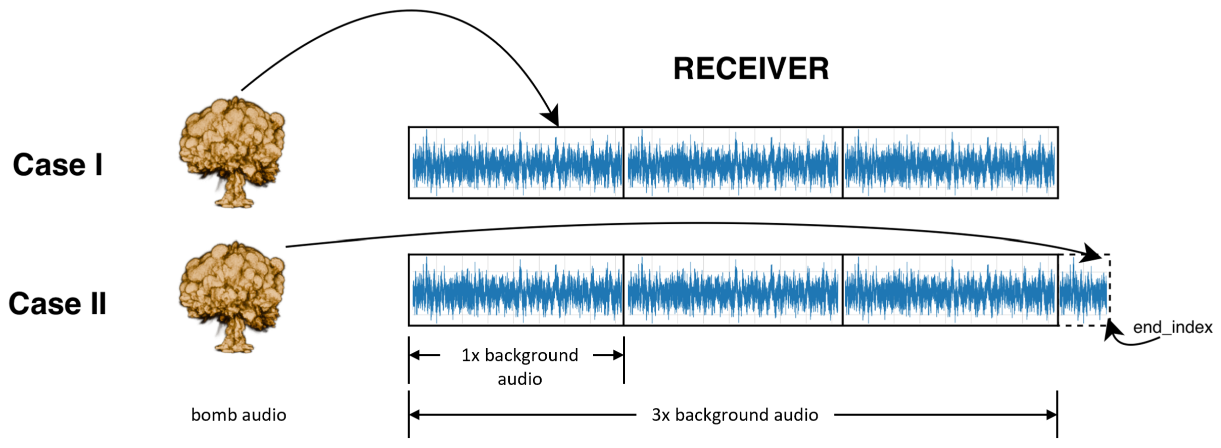

Figure 3.10: shows the two cases of receiving the blast signal on top of the background loop audio signal

Figure 3.10 illustrates the two cases at the receiving end of the output signal i.e. the background audio loop integrated with the bomb audio signal.

Case 1: If the blast occurs within the first loop of the background audio, the audio stream remains unchanged, maintaining the original sequence.

Case 2: If the blast occurs after the completion of the final loop of the background audio, the audio stream is extended to accommodate the blast sound. The new audio stream length becomes equivalent to 'end_index + duration of the background audio loop'.

The microphone plugin's packetization and streaming mechanism mirrors real-world audio streaming services, utilizing buffers to manage latency and ensure uninterrupted playback. This approach is crucial in the blast simulation for preserving the audio stream's temporal integrity, thereby enhancing realism and immersion. By dynamically managing the audio signal size and adapting to evolving requirements, the plugin significantly contributes to the fidelity and engagement of the simulated auditory environment.

3.3.3 Algorithm for the Microphone Plugin

As shown in Algorithm 2, the GazeboBlast3DMicrophonePlugin simulates the acoustic effects of blasts within a Gazebo simulation by processing and broadcasting audio signals corresponding to simulated explosions. Initially, the plugin loads and sets parameters from the SDF file, establishes necessary communication channels, and prepares audio files for background and blast noises. During each simulation tick, it checks if the conditions for publishing new audio data are met based on a predefined interval. If so, it processes the stored audio data, applies necessary filters (like averaging filters for seismic audio), and simulates the audio effect of a blast based on its distance and characteristics. This processed audio is then sent over the network, mimicking the real-time acoustic impact of blasts in a simulated environment.

Algorithm 2 Load and operate the GazeboBlast3DMicrophonePlugin

```
procedure LOAD(model, sdf)
    if model is NULL then
        print "Invalid model pointer."
        return
    end if
    Initialize model, world, and link references
    Read parameters from sdf
    Set namespace, link names, and register topics
    Load audio files and configure audio parameters
    Apply audio filters to simulate environmental effects
    Calculate audio physics properties based on environment
    Setup node and subscriptions for audio topics
    Register update event connection
end procedure

procedure ONUPDATE
    Capture current simulation time
    if publish interval elapsed then
        Prepare and send audio packet
        Replenish buffer with audio data if necessary
    end if
end procedure

procedure PUBLISHAUDIOMESSAGE(sampleData)
    Format audio data for transmission
    Set metadata (timestamp, frame id, etc.)
    Send audio data over network
end procedure

procedure BLAST3DCALLBACK(blast3d_msg)
    Extract blast information from message
    Compute audio attenuation and propagation delay
    Insert blast audio into buffer based on computed timing
    Adjust audio data based on environmental physics
end procedure

function AVERAGEFILTERWITHCUTOFF(signal, cutoffFreq, samplingFreq)
    Calculate window size based on cutoff frequency
    Apply average filter across the signal window
    Return the filtered signal
end function
```

3.4 Model Plugin

The model plugin is a key component of the blast simulation project, responsible for simulating the pressure and wind effects generated by a blast within the Gazebo environment. As a Gazebo model plugin, it interacts with the simulation environment to provide a realistic representation of the environmental impact of a blast scenario. The model plugin is responsible for simulating the dynamic effects of the blast, such as changes in pressure and wind velocity, and updating the simulation environment accordingly. Additionally, the plugin offers flexibility in simulating different types of blasts by utilizing both pre-defined and custom blast data. It is equipped to read detailed and specific blast profiles from CSV files, enabling the simulation of a wide range of blast scenarios and enhancing the versatility of the blast simulation project. In conclusion, the model plugin is crucial for simulating blast events in Gazebo, offering versatility with pre-defined and custom data and ensuring realistic environmental effects through integration with the Gazebo transport system.

3.4.1 Signal Modeling of Blast Waves

A description of a nominal blast pressure wave model is as follows:

> *The simplest form of a blast wave model has been described and termed the Friedlander waveform. It occurs when a high explosive detonates in a free field, that is, with no surfaces nearby with which it can interact. Blast waves have properties predicted by the physics of waves. For example, they can diffract through a narrow opening, and refract as they pass through materials. Like light or sound waves, when a blast wave reaches a boundary between two materials, part of it is transmitted, part of it is absorbed, and part of it is reflected. The impedances of the two materials determine how much of each occurs. The equation for a Friedlander waveform (Eqn. 3.5) describes the pressure of the blast wave as a function*

of time:

$$P(t) = P_s e^{-\frac{t}{t^*}} \left(1 - \frac{t}{t^*}\right) \tag{3.5}$$

where P_s is the peak pressure and t^* is the time at which the pressure first crosses the horizontal axis (before the negative phase). – Blast Wave, Wikipedia [43]

3.4.1.1 Signal propagation models and temporal consistency of visual, audio and shock wave sensor telemetry

Our payload sensor model includes an event camera, microphones, and anemometer data. The arrival times of shock waves that propagate using the seismic/air model are computed using the same equations mentioned earlier in section 3.3.1, Eqn.3.3 and Eqn.3.4, which originates from academic studies of blast events from the literature[41].

3.4.2 Algorithm for the Model Plugin

As shown in Algorithm 3, the GazeboBlast3DModelPlugin focuses on applying physical effects from simulated blast events directly to model elements within the Gazebo environment. When loaded, the plugin sets up connections and reads necessary configurations from the SDF file. It registers callbacks to handle incoming messages that contain blast event data and updates model states based on these events. In each update cycle, the plugin evaluates if any registered blast should be applied by checking its scheduled time against the current simulation time. If the criteria are met, it computes the force imparted by the blast based on its TNT equivalent and the distance to the affected link, applying this force directly to the model in the simulation, thereby altering its physical state as one would expect in a real-world scenario.

3.5 World Plugin

The World Plugin serves as a central coordinator within the blast simulation framework, operating within the Gazebo simulation environment. Its primary function is

Algorithm 3 Operation of the GazeboBlast3DModelPlugin

```
 1: procedure Load(model, sdf)
 2:     if model is NULL then
 3:         print "Invalid model pointer."
 4:         return
 5:     end if
 6:     Initialize model and world references
 7:     Display model name for debugging
 8:     Read and set parameters from sdf for namespaces and topics
 9:     Attempt to get link based on the specified link name
10:     Connect to the world update event
11: end procedure

12: procedure OnUpdate
13:     if not already done then
14:         Create and configure publishers and subscribers
15:         Mark pubs and subs as created
16:     end if
17:     Get current simulation time
18:     for each blast message in list do
19:         if blast is due then
20:             Calculate blast effects on the model
21:             Apply force to the model link based on blast parameters
22:             Optionally remove the blast message from the list
23:         end if
24:     end for
25:     Remove handled blast messages from list
26: end procedure

27: procedure Blast3DCallback(blast3d_msg)
28:     print "Blast message received."
29:     Copy message details to local structure
30:     Add to list of upcoming blasts
31: end procedure

32: procedure CreatePubsAndSubs
33:     Advertise registration and blast topics
34:     Subscribe to blast messages
35:     Send registration message to server
36: end procedure
```

to manage and disseminate information about simulated blast events to various components of the simulation, ensuring a synchronized and realistic representation of explosions. Upon initialization, the plugin establishes its communication channels by subscribing to a topic designated for receiving blast event registration requests. This allows other plugins or simulation entities to register themselves as recipients of blast event notifications, thereby enabling a coordinated response to simulated explosions.

The World plugin is equipped with the capability to randomly generate blast events using a stochastic model based on a uniform distribution. The use of a uniform distribution is particularly effective in environments where no prior data suggests a pattern or trend in blast occurrences, ensuring that simulations remain unbiased and encompass a broad spectrum of potential scenarios. By drawing parameters uniformly at random, the model prevents any unintentional bias that might influence the training or analysis outcomes of the simulation, making it a robust tool for testing and developing blast response strategies. This model systematically determines the occurrence, location, TNT equivalent, and timing of blasts, drawing each parameter uniformly at random from predefined ranges specified for the operational environment. The resulting blast information is then encapsulated in messages and disseminated to registered entities within the simulation, enabling them to respond appropriately.

Furthermore, the plugin maintains a registry of entities that have subscribed to receive blast event notifications. This ensures that all registered components are promptly and accurately informed about occurring blast events. This registration system promotes a modular and adaptable simulation architecture, allowing various components to independently process and react to blast events.

3.5.1 Algorithm for the World Plugin

As shown in Algorithm 4, the GazeboBlast3DWorldPlugin administers the overall environment in which blasts occur within the Gazebo simulation. This plugin initializes by setting up the simulation environment, subscribing to specific topics to listen

Algorithm 4 Operation of the GazeboBlast3DWorldPlugin

```
 1: procedure LOAD(world, sdf)
 2:     Initialize world reference
 3:     Create a node handle and initialize
 4:     Read parameters from sdf for topic names and rates
 5:     Subscribe to the blast registration topic
 6:     Connect to the update event for simulation iteration
 7: end procedure

 8: procedure REGISTERLINKCALLBACK(msg)
 9:     Extract registration details from msg
10:     if model and link are valid then
11:         Register link for receiving blast effects
12:         Log registration
13:     else
14:         Log failure to find model or link
15:     end if
16: end procedure

17: procedure ONUPDATE
18:     Capture current simulation time
19:     if interval elapsed and random condition for blast met then
20:         Generate random parameters for blast
21:         Log blast occurrence details
22:         Distribute blast details to registered links
23:     end if
24: end procedure
```

for blast registrations, and connecting to the simulation update event. Its primary function during the simulation is to randomly initiate blasts based on predefined probabilities and intervals. When a blast is decided to occur, it randomly generates blast characteristics such as location, time of occurrence, and intensity. These details are then communicated to all registered entities within the simulation that might be affected by the blast, ensuring that the simulated world dynamically responds to these random events, mimicking unpredictability and chaos of real-life explosions within a controlled virtual environment.

CHAPTER 4: RESULTS

This chapter presents the results obtained from the deployment of the four specialized plugins within the Gazebo simulation framework. Each plugin was critically evaluated to assess its effectiveness in enhancing the realism and computational efficiency of simulating blast phenomena.

4.1 Camera Plugin

The Camera Plugin was crucial for capturing the visual dynamics of the blast events. It offers two operational modes to accommodate different observational needs:
RGB Mode: This mode provides a realistic representation of the blast as it would appear to the human eye.
Events Mode: This mode displays the events of the blast, capturing dynamic changes as the UAV maneuvers and the camera detects the blast. The visibility of background elements can be adjusted based on the threshold settings. The plugin successfully indexes and manages the rotation of visual data to align with the UAV's orientation, ensuring accurate visual representation from various perspectives.

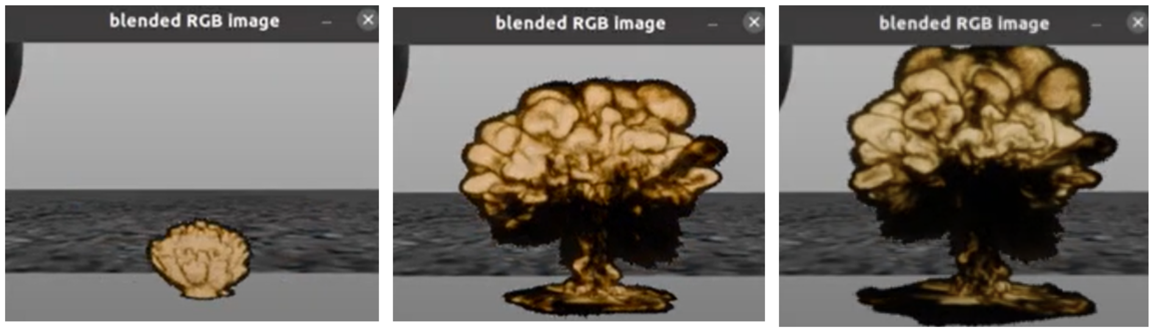

Figure 4.1: shows three frames of the RGB mode of the Camera Plugin

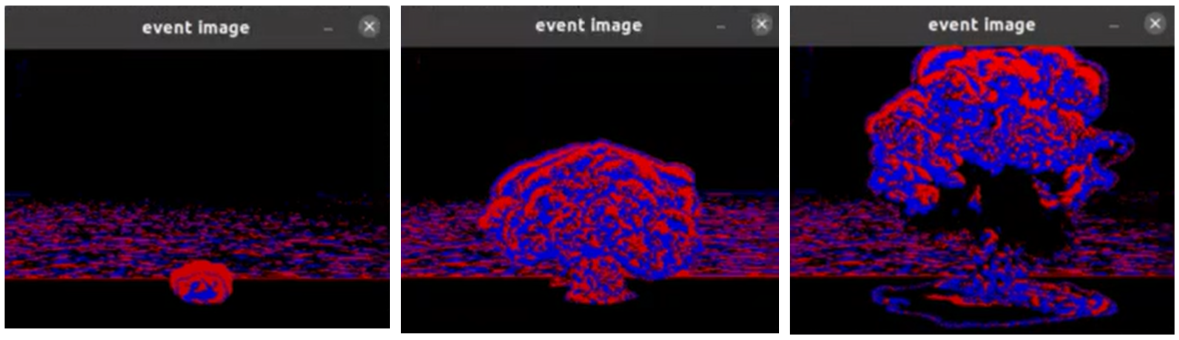

Figure 4.2: shows three frames of the Event mode of the Camera Plugin

Figures 4.1 and 4.2 are screenshots illustrating the RGB and Events modes of the Camera Plugin, respectively. These images display the visual dynamics of the blast events, capturing the blast's appearance to the human eye in RGB mode and highlighting dynamic changes in the environment through the Events mode.

4.2 Microphone Plugin

The Microphone Plugin's performance was evaluated based on its ability to accurately simulate the acoustic effects of blasts. The plugin successfully replicated the dual nature of sound propagation - through the ground as seismic waves and through the air as sound waves. Audio fidelity tests confirmed that the simulated blast sounds were consistent with empirical data, with minimal discrepancies. The plugin efficiently managed ambient noise levels, adjusting audio outputs based on the simulated altitude changes, which added an additional layer of realism to the acoustic environment.

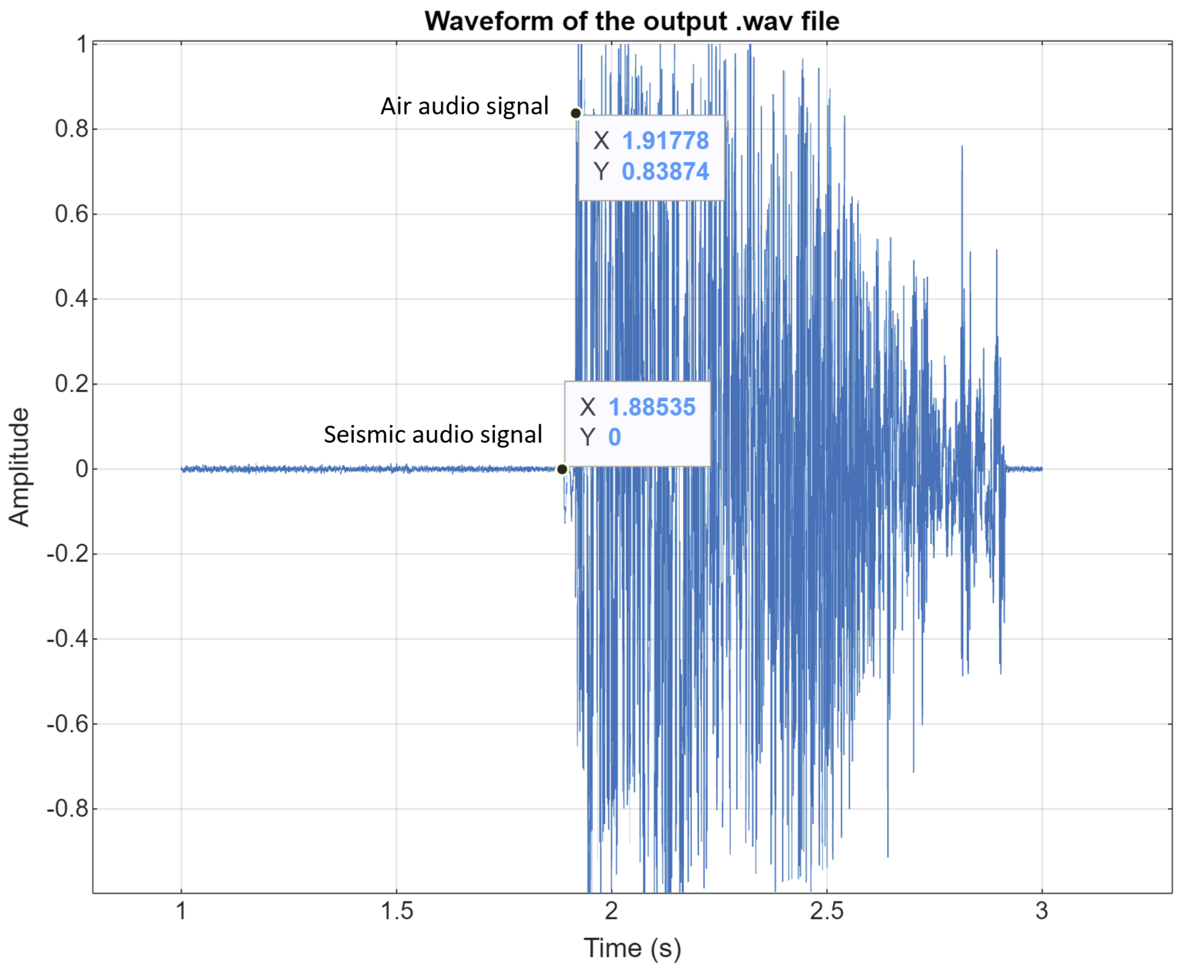

Figure 4.3: shows a plot of the output audio signal

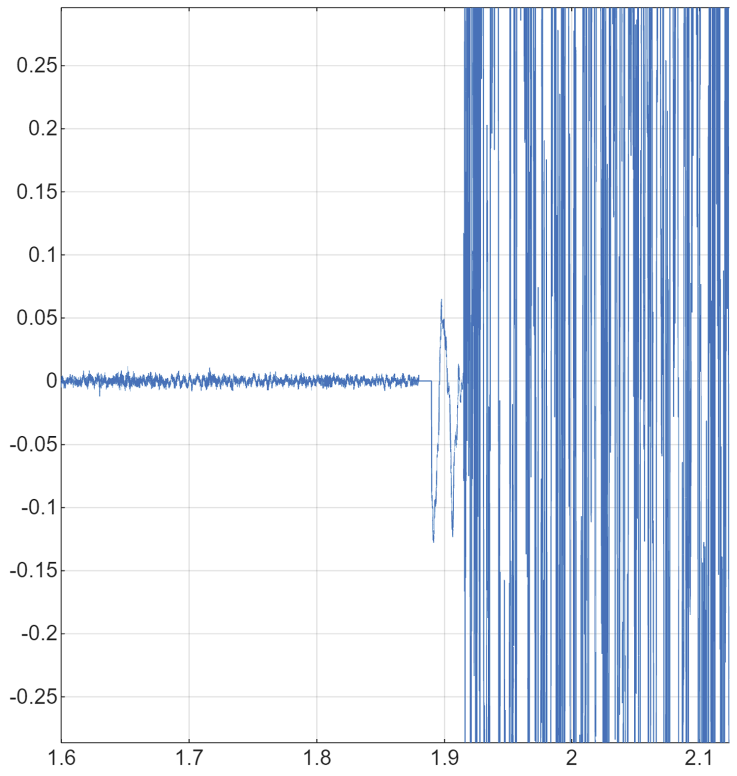

Figure 4.4: shows a zoomed in plot of the low frequency seismic output signal, followed immediately by the air propagated signal

Figures 4.3 and 4.4 present the overall output plot of the audio signal and a zoomed-in view focusing on the low-frequency seismic audio, respectively. These figures illustrate the slight but noticeable delay between the air propagated signal and the seismic signal, effectively capturing the sequence in which the events are perceived.

4.2.1 Analysis of Propagation Times

The propagation times for seismic and air waves, as observed in the simulation, were meticulously compared to the theoretical values derived from the empirical models introduced in Chapter 3. Specifically, the arrival times were calculated using the equation for seismic waves Eqn. 3.3 and the equation for air waves Eqn. 3.4 .

4.2.1.1 Observed Propagation Times

The observed propagation times from the simulation were as follows:

- Seismic propagation: 1.88535 seconds.

- Air propagation: 1.91778 seconds.

4.2.1.2 Theoretical Validation

These observed times were closely examined against the theoretical predictions. The parameters used for these calculations included:

- Equivalent TNT charge weight (Q): [5,15] kg.
- Distance from the blast center (R): [10, 20] meters.
- Seismic wave velocity (c_s): 6000 m/s (as per granite mass properties).
- Air wave velocity (c_a): 340 m/s (speed of sound in dry air).

The theoretical arrival times were calculated using dimensionally consistent empirical models, where distance (R) and charge weight (Q) are transformed into dimensionless parameters by normalizing with the relevant scaling factors, as described in Wu and Hao (2005). This normalization ensures that the models are applicable across different scales of blasts and observational distances.

4.2.1.3 Comparison and Discussion

The calculated theoretical values, adjusted for dimensionless scaling, were found to be closely aligned with the observed results. This alignment validates the model's accuracy and the application of dimensionless scaling in predicting propagation times for seismic and air waves. The slight discrepancies noted between observed and theoretical values are attributed to computational precision limitations and the simplified assumptions used in the wave propagation models.

The 'Difference' results in table 4.1 substantiate the efficacy of the implemented simulation model in reproducing the temporal dynamics of blast-induced wave propagation, offering a robust tool for accurate analysis. Further analysis for the simulation as an integrated system are included in section 4.5.

Table 4.1: Comparison of observed and calculated propagation times, the specific case of Q=15 and R=20 meters

Propagation Type	Observed Time (s)	Calculated Time (s)
Seismic (t_s)	1.88535	0.00314
Air (t_a)	1.91778	0.03857
Difference (t_a - t_s)	0.03243	0.03543

4.3 Model Plugin

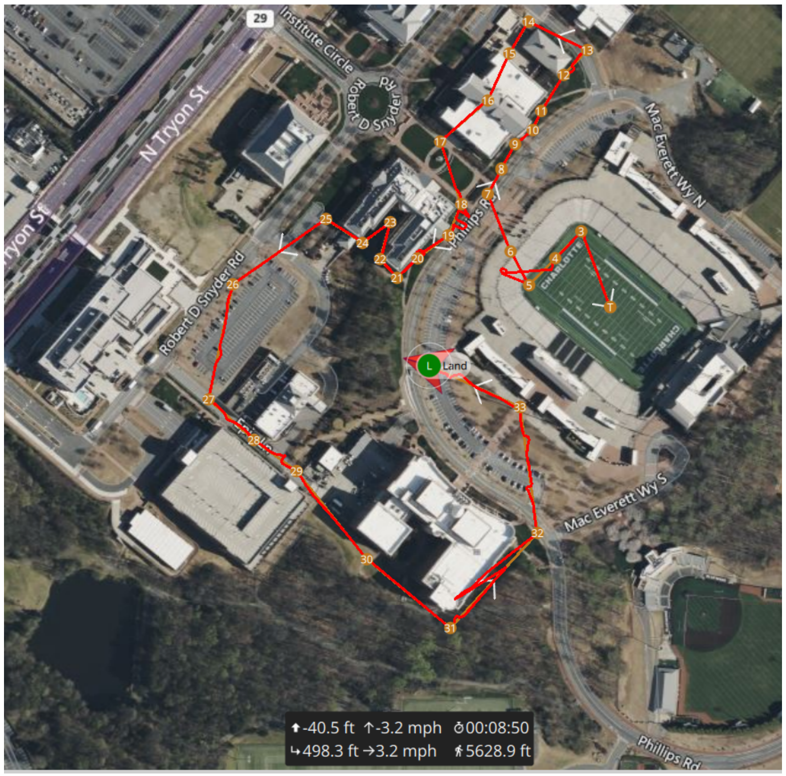

Figure 4.5: shows the trajectory followed by the UAV, as it is being affected by the model plugin

The Model Plugin's performance is crucial for simulating the environmental impacts of blasts, particularly in terms of pressure wave propagation. To assess its effectiveness, the plugin was supplied with a detailed shockwave pressure dataset in .csv format. This dataset included various blast parameters such as time delay, distance, and directional angles, which were used to generate simulations of blast effects.

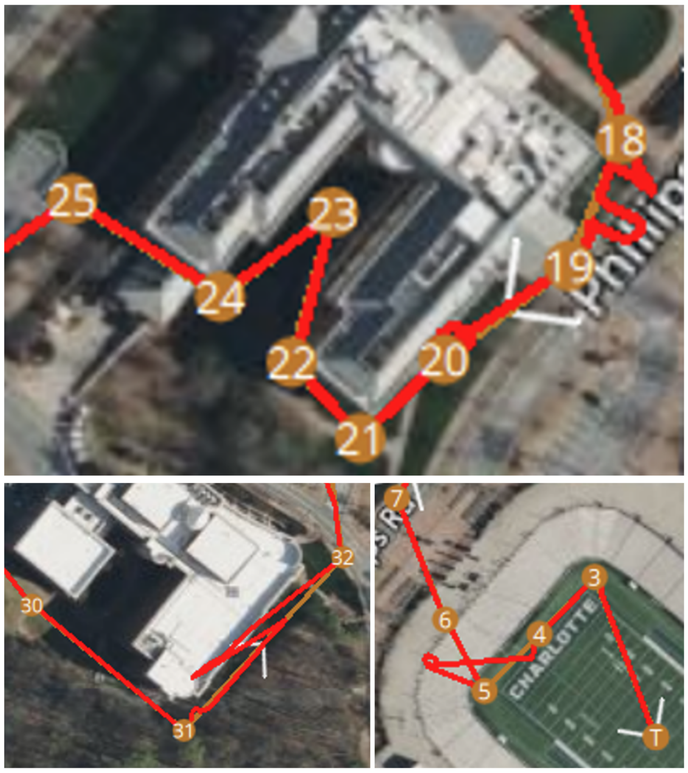

Figure 4.6: shows the zoomed-in path followed by the UAV, particularly between waypoints 18-19, 4-5-6, and 31-32 (clockwise)

The Model Plugin processed the inputs from the dataset to display the simulated blast effects within the Gazebo environment, showing how the UAV deviates from its intended trajectory due to the blast impact as seen in figure 4.5. The deviation from original trajectory is very clear between waypoints 4-5-6, 18-19, and 31-32, as is observed in figure 4.6. This trajectory is based on the latitude and longitude coordinates of the University of North Carolina Charlotte campus in the QGroundControl tool. The deviation is a direct result of the forces applied by the simulated blast, as calculated by the plugin. Additionally, the blast's arrival time is indicated on the Gazebo terminal with a future time marker, demonstrating the latency and real-time capabilities of the simulation. This visual and data-driven feedback from the plugin highlights its precision and utility in creating realistic blast scenarios that can dynamically affect the environment and objects within the simulation.

4.4 World Plugin

The integration capabilities of the World Plugin were thoroughly tested, showing excellent management of data flow and synchronization across the simulation framework. The plugin effectively coordinated the operations of the other three plugins, ensuring that all sensory data was aligned with the simulated time frame. This synchronization was crucial for maintaining the continuity and coherence of the simulation, allowing for complex scenarios involving multiple blasts to be accurately depicted.

4.5 Experimentation

In the experimental validation of the blast simulation framework, two critical setups were analyzed through live plotting within the Gazebo simulation environment:

4.5.1 Blast Visualization and Data Plotting

During the experiments, a vehicle was navigated along a predefined path while the blast plugin was active. This setup allowed for the real-time visualization and live plotting of force, torque, audio, and color and/or event imagery associated with the blast events. The objective was to demonstrate the vehicle's real-time response to the simulated blast dynamics and to illustrate the plugin's capability to accurately simulate the physical and acoustic effects on the vehicle. The result of one such experiment can be seen in Fig. 4.7.

4.5.2 Path Tracking and Error Analysis

Another setup focused on evaluating the vehicle's ability to maintain or regain its course following a blast-induced disturbance. As the vehicle traversed the designated path, live plots captured force, torque, and particularly path error metrics both before and after the blast. This included documenting the recovery time necessary for the vehicle to reacquire nominal path-tracking status post-disturbance. The live plotting provided a clear, quantitative insight into the simulation's effectiveness in replicating the real-world disruptions caused by explosions. The result of one such experiment can be seen in Fig. 4.8.

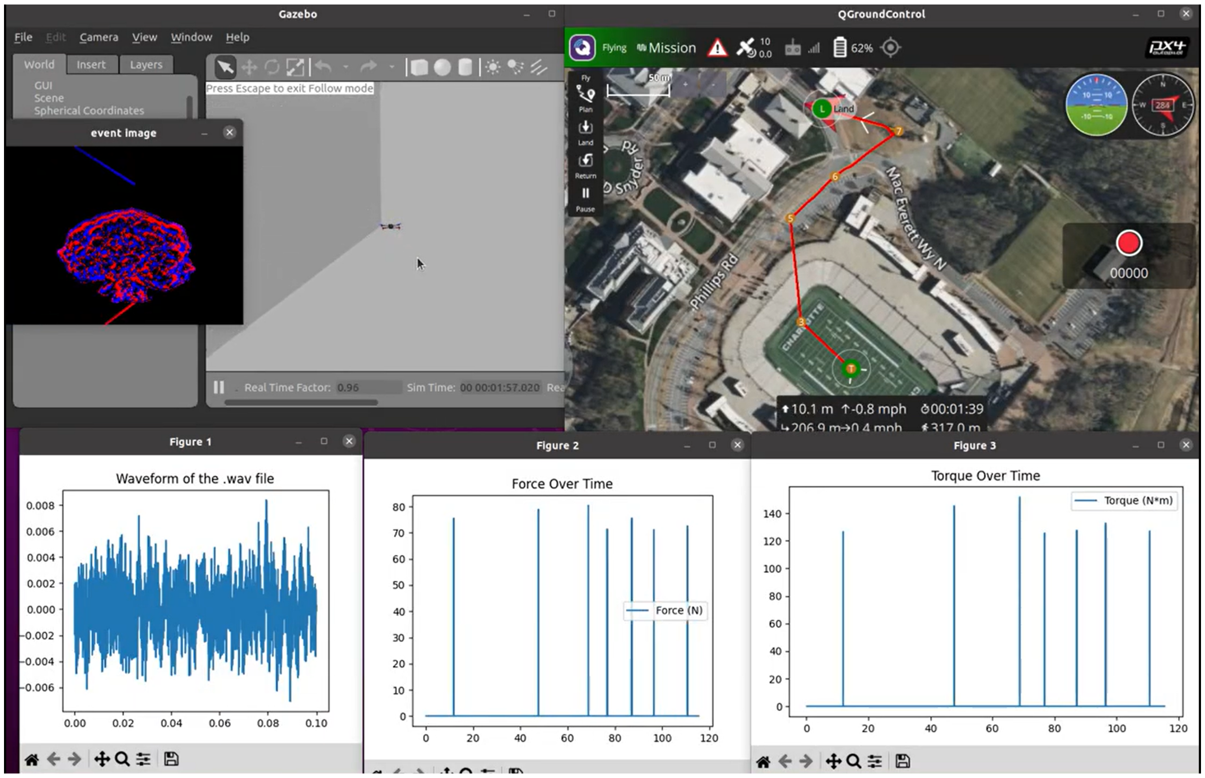

Figure 4.7: shows a screen capture of the results generated while performing the Blast Visualization and Data Plotting experiment

The figure 4.7 displays a multi-faceted view of a vehicle navigating through a virtual environment while subjected to blast simulations. The top left section shows the Gazebo simulation environment where the vehicle's trajectory is visualized. Next to it, the QGroundControl software maps the vehicle's path over actual geographic data, adding a layer of realism and enabling precise path tracking and mission planning. Below these, there are synchronized live plots depicting the waveform of the blast's audio signal and graphs showing the force and torque experienced by the vehicle over time. This setup demonstrates the integration of visual, auditory, and physical simulation data, which helps in understanding the vehicle's response to simulated blast forces and sound waves.

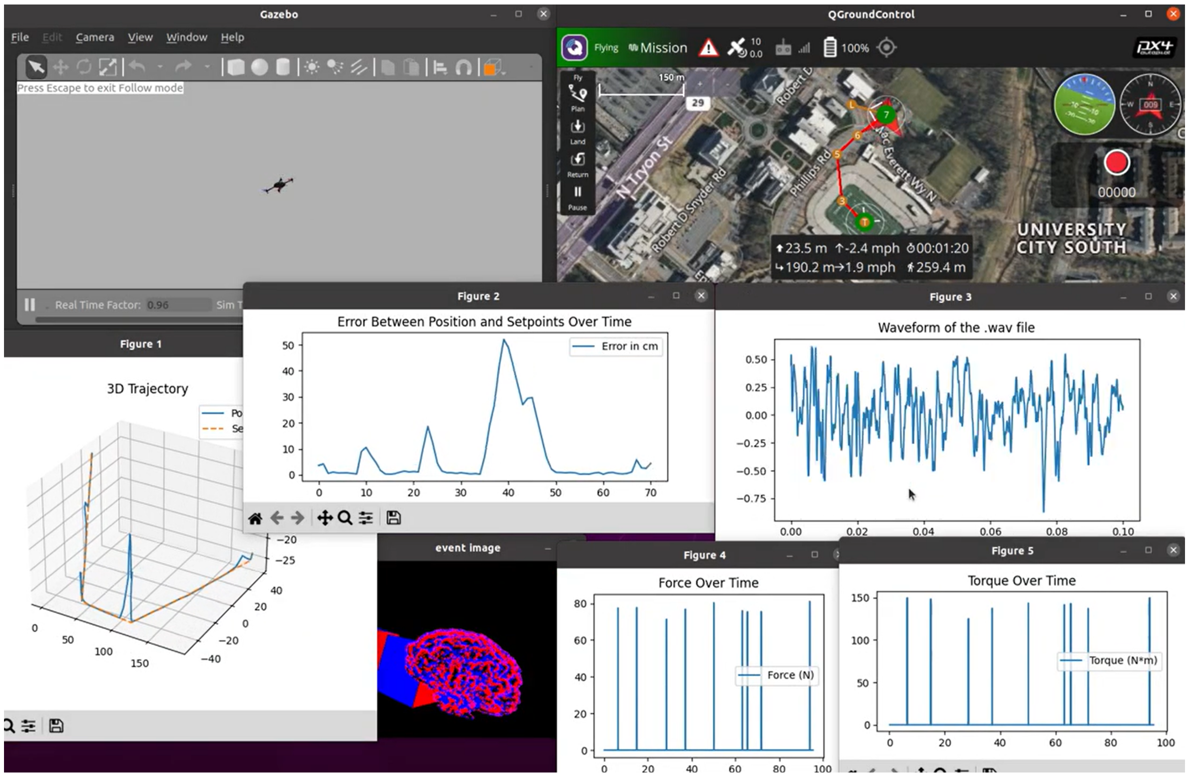

Figure 4.8: shows a screen capture of the results generated while performing the Path Tracking and Error Analysis experiment

In the figure 4.8, a comprehensive analysis of path tracking and error metrics is presented. It includes a 3D plot showing the vehicle's trajectory through the simulation environment, highlighting deviations and adjustments post-blast. Accompanying this are detailed plots measuring the error between the vehicle's actual position and the setpoints over time, which vividly illustrate the impact of the blast on the vehicle's path. Additional graphs again detail the force and torque over time, underlining the physical strains imposed by the blast events. This configuration provides clear, quantifiable insights into the vehicle's capability to adjust its path in response to external disturbances, which is critical for assessing the effectiveness of the simulation in mimicking real-world explosive interactions.

These figures collectively capture the dynamic interplay between visual, acoustic, and physical data in the simulation, offering a robust tool for analyzing the complex effects of blasts on vehicle dynamics within a controlled virtual environment.

These live experimental setups were crucial in assessing the robustness of the blast simulation framework. They provided immediate visual and quantitative feedback on how well the system predicts and replicates the complex interactions between a moving vehicle and blast-induced forces within a virtual environment.

To enhance the performance of these experimental setups, integrating a ring buffer data structure could significantly improve data management:

Continuous and Efficient Data Handling: The ring buffer provides a fixed-size, cyclic storage mechanism that allows continuous data capture without pauses for memory reallocation, crucial for managing high-frequency data streams like audio and force measurements.

Real-Time Processing: With a ring buffer, the most recent data is always available for quick access and processing, enabling real-time analysis and live plotting. This ensures immediate feedback and minimal latency in dynamic simulations, vital for accurate monitoring and response to blast effects.

CHAPTER 5: PROSPECTIVE DEVELOPMENTS

In this chapter, it is crucial to address the limitations of the microphone and event camera used in the simulations. In future iterations of the simulation framework, for the microphone, issues such as decibel saturation-the point at which a microphone's response to sound becomes non-linear and compressed-must be carefully managed. Advanced algorithms for dynamic range compression and better calibration methods could mitigate these effects, ensuring more accurate acoustic data capture, particularly in high-intensity environments like explosions.

Similarly, the resolution, distortion, and intrinsic parameters of the event camera need rigorous examination. The current setup may not fully account for optical distortions and resolution limitations, which can affect the accuracy of visual data, especially in dynamic and complex visual fields. Future work can include implementing distortion correction techniques and evaluating higher-resolution cameras to enhance the fidelity of visual simulations. Additionally, refining the camera's intrinsic parameters, such as focal length and lens characteristics, will improve the precision of event detection and image processing, critical for accurate reproduction of blast phenomena.

Future enhancements of this work will focus on improving the scalability of the simulation framework to handle larger and more complex environments. Optimizations may include refining the data handling and processing capabilities of the World Plugin to reduce latency further and extend the framework's application to other types of explosive phenomena in industrial and mining sectors.

5.1 Detectors

The initial development phase of the explosion detector has yielded promising results, yet the system currently functions independently of the integrated simulation environment and associated plugins. This separation represents a significant opportunity for future integration, which is expected to enhance both the utility and realism of the overall simulation framework.

The explosion detection system is comprised of two distinct components, each designed to operate on a different sensory modality but united by their common goal to achieve ultra-low-latency detection of blast events using C++.

5.1.1 Visual Events Detector

The visual detector operates based on principles similar to those used in the camera plugin. It specifically monitors rapid movements within the visual frame, with a particular focus on vertical movements along the y-axis. This detection criterion is based on the observation that explosions typically result in significant upward displacement of smoke and debris, which manifests as a sharp increase in y-axis values. Such changes, occurring virtually at the speed of light, can reliably indicate an explosion when they exceed a predetermined threshold.

5.1.2 Audio Signal Detector

The acoustic detector employs a matched filtering technique to discern the presence of a "bomb audio" signature within a noisy audio signal. The process involves several key steps: initially, both the reference and input audio signals undergo a Fast Fourier Transform (FFT) to convert them into the frequency domain. The FFT of the input signal is then multiplied by the complex conjugate of the FFT of the reference signal. The product of these transforms is subjected to an inverse FFT to generate the filtered output signal. The effectiveness of this detection is determined by analyzing the peaks in the filtered output, applying an amplitude threshold to

Algorithm 5 Visual Event Detection for Blast Identification

```
Require: Folder containing sequential grayscale images
Ensure: Detection of blast based on significant vertical event changes
 1: Initialize:
 2: previous_image ← None
 3: previous_event_count ← 0
 4: blast_detected ← False
 5: Process Images:
 6: for each image $i$ in the folder do
 7:     current_image ← load_image($i$)
 8:     if previous_image is not None then
 9:         events ← generate_events(previous_image, current_image)
10:         current_event_count ← count_vertical_events(events)
11:         if current_event_count − previous_event_count > vertical_event_increase_threshold then
12:             Print "Blast detected between images ($i - 1$) and $i$!"
13:             blast_detected ← True
14:             break
15:         end if
16:         previous_event_count ← current_event_count
17:     end if
18:     previous_image ← current_image
19: end for
20: Finalize:
21: if not blast_detected then
22:     Print "No blast detected in the sequence."
23: end if
```

differentiate genuine detections from noise, and calculating the signal-to-noise ratio (SNR) for each peak to assess detection quality. Remarkably, this acoustic detection process can identify explosions in just a few milliseconds, underscoring its potential for real-time applications. The computational efficiency and efficacy of this process are crucial, with performance metrics including the time required for filtering and the accuracy of "bomb audio" identification based on peak and SNR analysis.

Looking ahead, the prospective integration of these detectors into the larger simulation framework is a primary focus. Such integration promises to streamline processes and improve the real-time responsiveness and accuracy of the simulation. This effort will involve embedding the detectors within the existing plugin architecture, allowing them to leverage real-time data directly from the simulation environment. This enhancement is expected to improve the performance of the simulation significantly, making it a more effective tool for training and analysis in safety-critical applications.

5.2 Machine Learning Optimization

Implementing machine learning algorithms to optimize simulation parameters dynamically could improve the realism and efficiency of the simulations. For instance, machine learning could be used to automatically adjust the parameters for visual and acoustic effects based on real-world data, enhancing the predictive accuracy and visual fidelity of the simulations. Relevant examples of such applications include the innovative rendering techniques by Kallweit et al. [44], fluid simulation enhancements by Tompson et al. [45], and advanced learning frameworks for unmanned aerial systems by Jagannath et al. [46].

5.3 Interdisciplinary Applications

This framework serves as a pioneering case study demonstrating its versatility and adaptability to other fluid dynamics simulations beyond blast phenomena. Specifi-

Algorithm 6 Audio Signal Processing for Bomb Detection using Matched Filter

```
Require: Input audio file with potential bomb sound, Reference bomb sound file
Ensure: Detection of bomb sound
 1: Read WAV Files:
 2: Reference ← read_wav("reference_bomb.wav")
 3: Input ← read_wav("input_audio.wav")
 4: Perform FFT:
 5: Reference_FFT ← perform_fft(Reference)
 6: Input_FFT ← perform_fft(Input)
 7: Matched Filtering:
 8: for $i = 0$ to length(Reference_FFT) do
 9:     Result_FFT[$i$] ← Input_FFT[$i$] * conj(Reference_FFT[$i$])
10: end for
11: Inverse FFT:
12: Output_Signal ← inverse_fft(Result_FFT)
13: Save Output Signal:
14: save_wav(Output_Signal, "output_signal.wav")
15: Detect Peaks:
16: Peaks ← detect_peaks(Output_Signal, Threshold)
17: Check Signal-to-Noise Ratio (SNR):
18: for each peak in Peaks do
19:     SNR ← calculate_snr(Output_Signal, peak)
20:     if SNR > SNR_Threshold then
21:         Bomb Detected ← True
22:         break
23:     end if
24: end for
25: Print Detection Result:
26: if Bomb Detected then
27:     Print "Bomb audio detected!"
28: else
29:     Print "Bomb audio NOT detected."
30: end if
```

cally, the methodologies and computational techniques developed for blast simulations could be directly applicable to the study of fluid flows in scenarios such as water currents, oil spills, or atmospheric disturbances. By leveraging the existing architecture, researchers can model the behavior of various fluids under different environmental conditions, providing valuable insights in fields ranging from environmental science to chemical engineering. This approach not only broadens the applicability of the current project but also sets a foundation for future research that could lead to breakthroughs in understanding complex fluid dynamics in natural and industrial contexts.

CHAPTER 6: CONCLUSIONS

This thesis has successfully developed and implemented a comprehensive simulation framework within Gazebo, designed to enhance the realism and accuracy of blast phenomenon simulations. By integrating visual and acoustic data, the project has demonstrated significant improvements in the fidelity and effectiveness of simulations used for safety and defense training. The implementation of four specialized plugins - the Camera, Microphone, Model, and World plugins - has allowed for a nuanced simulation environment where each aspect of a blast, from visual appearance to sound propagation and the pressure modeling, is dynamically represented.

The integration of the Robot Operating System (ROS) with enhancements from Gazebo and the utilization of QGroundControl and PX4 Software In The Loop (SITL) firmware have streamlined the simulation process, reducing manual intervention and increasing efficiency. The transition to the OpenAI GymFC framework has further enabled a more controlled and configurable simulation environment, proving essential for detailed analysis and robust simulation capabilities [8].

Furthermore, the project has extended the utility of these simulations by focusing on realistic rendering of explosions, making these tools not only more accurate but also more responsive to the needs of real-world applications. The development of an ambient acoustic model that adjusts background noise based on altitude is a testament to the project's commitment to creating an immersive and authentic simulation environment.

Future research will delve deeper into refining these simulations, exploring advanced computational techniques to further reduce latency and enhance the detail and accuracy of both visual and acoustic simulations. There is also potential to expand the use of these simulations in other realms such as urban planning and disaster management, where understanding the impacts of explosions can lead to better preparedness and response strategies.

In conclusion, this thesis has laid a solid foundation for the realistic simulation of blast phenomena, providing a valuable tool for training, planning, and research. It has set the stage for future innovations that will continue to push the boundaries of what is possible in simulation technology.

REFERENCES

[1] "Gazebo — gazebosim.org." https://gazebosim.org/home. [Accessed 06-05-2024].

[2] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, 2004.

[3] "ROS: Home — ros.org." https://www.ros.org/. [Accessed 06-05-2024].

[4] "QGroundControl." https://qgroundcontrol.com/. [Accessed 06-05-2024].

[5] "Open Source Autopilot for Drones - PX4 Autopilot — px4.io." https://px4.io/. [Accessed 06-05-2024].

[6] "Open AI." https://openai.com/. [Accessed 06-05-2024].

[7] "GitHub - wil3/gymfc: A universal flight control tuning framework — github.com." https://github.com/wil3/gymfc. [Accessed 06-05-2024].

[8] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement Learning for UAV Attitude Control," *ACM Transactions on Cyber-Physical System*, vol. 3, feb 2019.

[9] H. Fu, W. Xu, H. Xue, H. Yang, R. Ye, Y. Huang, Z. Xue, Y. Wang, and C. Lu, "RFUniverse: A Physics-based Action-centric Interactive Environment for everyday household tasks," *CoRR*, vol. abs/2202.00199, 2022.

[10] T. Henderson, P. McMurtry, P. Smith, G. Voth, C. Wight, and D. Pershing, "Simulating accidental fires and explosions," *Computing in Science Engineering*, vol. 2, no. 2, pp. 64–76, 2000.

[11] R. Bridson and M. Müller-Fischer, "Fluid simulation: Siggraph 2007 course notes," in *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, (New York, NY, USA), p. 1â81, Association for Computing Machinery, 2007.

[12] N. Rasmussen, D. Q. Nguyen, W. Geiger, and R. Fedkiw, "Smoke simulation for large scale phenomena," in *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, (New York, NY, USA), pp. 703–707, Association for Computing Machinery, 2003.

[13] K. Um, X. Hu, and N. Thuerey, "Liquid splash modeling with neural networks," *Computer Graphics Forum*, vol. 37, no. 8, pp. 171–182, 2018.

[14] P. Hernandez, J. Ramírez, and M. Soto, "Deep-learning-based earthquake detection for fiber-optic distributed acoustic sensing," *Journal of Lightwave Technology*, vol. PP, pp. 1–1, Dec. 2021.

[15] Z. Li and Z. Zhan, "Pushing the limit of earthquake detection with distributed acoustic sensing and template matching: a case study at the Brady geothermal field," *Geophysical Journal International*, vol. 215, pp. 1583–1593, Sept. 2018.

[16] Q. Brissaud, S. Krishnamoorthy, J. M. Jackson, D. C. Bowman, A. Komjathy, J. A. Cutts, Z. Zhan, M. T. Pauken, J. S. Izraelevitz, and G. J. Walsh, "The first detection of an earthquake from a balloon using its acoustic signature," *Geophysical Research Letters*, vol. 48, June 2021.

[17] A. Golas, A. Khan, P. Kalra, and S. Kumar, "Explosion simulation using compressible fluids," in *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, pp. 63–70, 2008.

[18] K. Sims, "Particle animation and rendering using data parallel computation," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, (New York, NY, USA), pp. 405–413, Association for Computing Machinery, 1990.

[19] C. Martins, J. Buchanan, and J. Amanatides, "Visually believable explosions in real time," in *Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No.01TH8596)*, pp. 237–259, 2001.

[20] X. He, L. Yang, S. Li, and A. Hao, "Physics based real-time explosion simulation," in *2012 Fourth International Conference on Digital Home*, pp. 309–314, 2012.

[21] G. D. Yngve, J. F. O'Brien, and J. K. Hodgins, "Animating explosions," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pp. 29–36, ACM Press, Jan 2000.

[22] B. E. Feldman, J. F. O'Brien, and O. Arikan, "Animating suspended particle explosions," pp. 708–715, July 2003.

[23] B. Klingner, B. Feldman, N. Chentanez, and J. O'Brien, "Fluid animation with dynamic meshes," *ACM Trans. Graph.*, vol. 25, pp. 820–825, July 2006.

[24] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," *ACM Transactions on Graphics*, vol. 24, pp. 910–914, July 2005.

[25] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH01, ACM, Aug. 2001.

[26] J. Sewall, N. Galoppo, G. Tsankov, and M. Lin, "Visual simulation of shockwaves," *Graphical Models*, vol. 71, no. 4, pp. 126–138, 2009. Special Issue of ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2008.

[27] J. S. Díaz and S. E. Rigby, "Blast wave kinematics: theory, experiments, and applications," *Shock Waves*, vol. 32, pp. 405–415, July 2022.

[28] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, *Extending Open Dynamics Engine for Robotics Simulation*, pp. 38–50. Springer Berlin Heidelberg, 2010.

[29] E. Coumans, "Bullet physics simulation," in *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, ACM, July 2015.

[30] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. Karen Liu, "Dart: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, p. 500, Feb. 2018.

[31] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: multibody dynamics for biomedical research," *Procedia IUTAM*, vol. 2, pp. 241–261, 2011.

[32] D. Ferigo, S. Traversaro, and D. Pucci, "Gym-ignition: Reproducible robotic simulations for reinforcement learning," *CoRR*, vol. abs/1911.01715, 2019.

[33] J. Kaiser, J. C. V. Tieck, C. Hubschneider, P. Wolf, M. Weber, M. Hoff, A. Friedrich, K. Wojtasik, A. Roennau, R. Kohlhaas, R. Dillmann, and J. M. Zöllner, "Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks," in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pp. 127–134, Dec 2016.

[34] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an open event camera simulator," *Conf. on Robotics Learning (CoRL)*, Oct. 2018.

[35] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "tempogan: a temporally coherent, volumetric gan for super-resolution fluid flow," *ACM Transactions on Graphics*, vol. 37, pp. 1–15, July 2018.

[36] S. Pearson, "Blender Tutorial - Creating a Simple Explosion Simulation — YouTube." https://www.youtube.com/watch?v=MxfctMsd0Ck, Feb 2023. [Online; accessed 5-May-2024].

[37] D. P. Hardin, T. J. Michaels, and E. B. Saff, "A Comparison of Popular Point Configurations on S^2 - arxiv.org." https://arxiv.org/abs/1607.04590, 2016. [Accessed 07-05-2024].

[38] M. Roberts, "How to evenly distribute points on a sphere more effectively than the canonical fibonacci lattice," Feb 2024.

[39] Wikipedia contributors, "Stokes's law of sound attenuation — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Stokes2024. [Online; accessed 5-May-2024].

[40] Wikipedia contributors, "Acoustic attenuation — Wikipedia, the free encyclopedia," 2023. [Online; accessed 5-May-2024].

[41] C. Wu and H. Hao, "Modeling of simultaneous ground shock and airblast pressure on nearby structures from surface explosions," *International journal of impact engineering*, vol. 31, no. 6, pp. 699–717, 2005.

[42] E. C. Pielou, "Wave Energy: Sound Waves and Seismic Waves," in *The Energy of Nature*, University of Chicago Press, May 2001.

[43] Wikipedia contributors, "Blast wave — Wikipedia, the free encyclopedia," 2024. [Online; accessed 5-May-2024].

[44] S. Kallweit, T. Müller, B. Mcwilliams, M. Gross, and J. Novák, "Deep scattering: rendering atmospheric clouds with radiance-predicting neural networks," *ACM Trans. Graph.*, vol. 36, nov 2017.

[45] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, "Accelerating Eulerian Fluid Simulation With Convolutional Networks," *CoRR*, vol. abs/1607.03597, 2016.

[46] J. Jagannath, A. Jagannath, S. Furman, and T. Gwin, "Deep learning and reinforcement learning for autonomous unmanned aerial systems: Roadmap for theory to deployment," *CoRR*, vol. abs/2009.03349, 2020.