

000 001 002 003 004 005 DECLARATIVE AUDIO EDITING WITH AUDIO 006 LANGUAGE MODEL 007

008 **Anonymous authors**
009

010 Paper under double-blind review
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025

026 ABSTRACT 027

028 Audio editing plays a central role in VR/AR immersion, virtual conferencing,
029 sound design, and other interactive media. However, recent generative audio editing
030 models depend on template-like instruction formats and are restricted to mono-
031 channel audio. These models fail to deal with declarative audio editing, where
032 the user declares what the desired outcome should be, while leaving the details
033 of editing operations to the system. We introduce SmartDJ, a novel framework
034 for stereo audio editing that combines the reasoning capability of audio language
035 models with the generative power of latent diffusion. Given a declarative instruction,
036 SmartDJ decomposes it into a sequence of atomic edit operations, such as adding,
037 removing, or spatially relocating events. These operations are then executed by a
038 diffusion model trained to manipulate stereo audio. To support this, we design a
039 data synthesis pipeline that produces paired examples of declarative instructions,
040 atomic edit operations, and audios before and after each edit operation. Experiments
041 demonstrate that SmartDJ achieves superior perceptual quality, spatial realism, and
042 semantic alignment compared to prior audio editing methods. Demos are provided
043 in the supplementary file. Code and data will be released upon acceptance.
044

045 1 INTRODUCTION 046

047 Imagine you recorded a forest on a rainy day and wanted to transform it into the soundscape of a
048 sunny forest. Achieving this transformation requires many edits: removing mismatched elements
049 like rainfall and adding new effects such as bright leaf rustling. Traditionally, audio editing is a
050 *procedural process*: the user specifies how to achieve the goal, step by step, by removing rain,
051 layering new samples, adjusting gain, and so on. Yet in practice, users would prefer just to issue a
052 single, declarative instruction, e.g., *“make this sound like a sunny day”*, and rely on an intelligent
053 audio editor to decide what edits are needed. We call this *declarative editing*: the user declares *what*
054 the desired outcome should be, while leaving the *how*, i.e., the sequence of operations, to the system.
055 Such a declarative paradigm would unlock a wide range of applications in VR/AR, gaming, cinematic
056 post-production, and beyond, where designing and modifying audio scenes is central to immersive
057 experiences.

058 Recent advances in deep generative models have opened exciting possibilities for text-to-audio
059 generation and language-driven editing (Evans et al., 2024a;b; Hai et al., 2024; Heydari et al., 2024;
060 Huang et al., 2023a; Liu et al., 2023a; 2024a; Jia et al., 2024; Liang et al., 2024; Wang et al., 2023;
061 Xu et al., 2024). However, existing audio editors remain limited in two key aspects. First, they rely on
062 templated instructions such as “add the sound of birds” or “remove the sound of rain”, which restricts
063 their ability to handle declarative or abstract user instructions. Second, they operate only on monaural
064 (single-channel) audio, discarding spatial features such as interaural time and level differences, which
065 are primary cues for spatial hearing. Without these cues, even semantically correct edits sound flat
066 and fail to deliver immersive experiences.

067 In contrast, declarative editing requires an editor to bridge from declarative goals to detailed operations
068 automatically. When a user issues instructions such as *“place me in a concert hall”* or *“have this*
069 *audio in a library”*, the system must reason about which sounds to remove, which to preserve, how
070 to adjust loudness, when to introduce new events, and how to shift spatial position. Current audio
071 editors based on diffusion models cannot achieve this, as they lack the reasoning capacity to interpret
072 these declarative instructions. On the other hand, language models alone are also insufficient: while
073 they can parse the user’s request, they lack grounding in the audio itself, making it impossible to
074 decide which sound elements should be suppressed, emphasized, or retained. This gap highlights
075

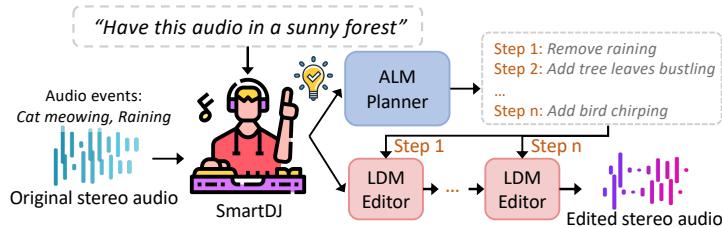


Figure 1: SmartDJ has an Audio Language Model (ALM) that acts as an edit planner to decompose the instructions into atomic steps, guiding the Latent Diffusion Model (LDM) editor to produce high-quality edited audio.

the need for a framework that jointly reasons over language and audio, combining natural-language instruction understanding with audio-aware analysis and editing.

In this work, we present SmartDJ, the first framework for declarative audio editing by introducing Audio Language Models (ALM) into the audio editing loop. This approach is motivated by the recent success of MLLMs in multimodal grounding and reasoning (Chu et al., 2023; Ghosh et al., 2025a; Kong et al., 2024a; Liu et al., 2023b; Li et al., 2023a). As illustrated in Fig. 1, our key idea is to let the ALM act as a *planner*: it perceives the original audio while interpreting the user’s declarative instruction and decomposes it into *a sequence of atomic edit operations*, such as adding or removing a sound, shifting direction, or adjusting volume, etc. These atomic edit operations are then executed sequentially by a conditional Latent Diffusion Model (LDM) as an audio *editor*, realizing the user’s declarative goal. This separation of planning and editing transforms audio editing from a procedural task into a declarative one. Moreover, because the intermediate representation is expressed in natural language, users can easily inspect, refine, or override the planned edits interactively.

Training a system to perform declarative editing introduces a fundamental challenge: it requires paired examples of *declarative instructions*, their corresponding *atomic edit sequences*, and the *before-and-after edited audio*. Such data is rarely available, as natural soundscapes are complex and difficult to edit at scale. To address this, we construct a scalable data generation pipeline that provides *controllable audio scenes*. Each scene is assembled from independently parameterized sound events with attributes such as direction and loudness. Within this pipeline, an off-the-shelf LLM acts as the *designer*, producing diverse declarative editing instructions and the corresponding atomic operations. Audio signal processing then serves as the *composer*, rendering each operation by adjusting or mixing sound events to produce the corresponding audio after every edit. Together, this designer–composer pipeline mirrors the planner–editor structure of our model. It produces a large-scale corpus of realistic editing operations, providing the supervision needed to train and evaluate our model.

Experimental results show that SmartDJ delivers superior editing quality and better alignment with declarative user instructions from both subjective metrics and human evaluations. Our LDM also outperforms existing baselines for audio editing. Ablation studies show that our ALM can effectively reason about and decompose declarative user instructions into a sequence of editing actions.

In summary, our main contributions are as follows:

- We introduce SmartDJ, the first stereo audio editor capable of interpreting declarative user instructions with an ALM and executing them as precise atomic edit operations through an LDM.
- We introduce the first scalable pipeline for generating editable stereo audio scenes, combining declarative instructions with controllable events to enable reasoning-based audio editors.
- We conduct extensive experiments and user studies with different baseline methods and demonstrate that SmartDJ achieves the highest editing quality for both objective and subjective metrics.

2 RELATED WORK

Audio generation and editing. With the current advances in deep generative models, lots of methods have achieved high-quality audio generation from text and multi-modal conditions (Chen et al., 2024; Evans et al., 2024a; Hai et al., 2024; Huang et al., 2023a; Liu et al., 2023a; 2024a; Wang et al., 2023). Recently, spatial audio generation has attracted more attention (Evans et al., 2024a; Heydari et al., 2024; Liu et al., 2025; Sun et al., 2024), providing more realistic and immersive listening experiences. Parallel to these generation efforts, text-guided audio editing also emerged as a powerful tool for modifying existing audio recordings. Audit (Wang et al., 2023) introduced an end-to-end diffusion

108 model conditioned on both the input audio and simple, structured text commands, but its reliance
 109 on fixed editing templates limits flexibility to interpret declarative user prompts. WavCraft (Liang
 110 et al., 2024) leverages GPT-api to parse user instructions, yet it expects fully specified prompts (e.g.,
 111 "extract baby crying from the audio" or "apply a low-pass filter to the wave crashing sound and
 112 add it back"). Some recent work (Jia et al., 2024; Manor & Michaeli, 2024; Xu et al., 2024) adapts
 113 image-editing techniques (e.g., DDPM inversion, Null-text inversions, attention map manipulation) to
 114 monaural audio. They require precise token-level swapping or deletion to complete the edit, struggle
 115 with declarative instructions, and offer no support for stereo audio editing. To summarize, none of
 116 the existing work can interpret declarative user input to complete the audio editing. Besides, existing
 117 frameworks remain confined to monaural outputs and are ill-suited for immersive spatial scenarios.

118 **Multimodal Large Language Model.** Large language models (LLMs) are remarkable in natural
 119 language processing tasks. Provided with multimodal inputs, such as images and audio, multimodal
 120 LLMs (MLLMs) (Alayrac et al., 2022; Liu et al., 2023b; Li et al., 2023a; Team, 2025; Kong
 121 et al., 2024b; Ghosh et al., 2025b) demonstrate exceptional performance across a wide range of
 122 downstream visual-language and audio-language tasks. In the vision domain, LLaVA (Liu et al.,
 123 2023b) enables LLMs to achieve general-purpose visual and language understanding by fine-tuning on
 124 a multimodal instruction-following dataset. In the audio domain, LTU (Gong et al., 2023) and Audio
 125 Flamingo (Kong et al., 2024b; Ghosh et al., 2025b) enhance LLMs with the ability to process non-
 126 speech sounds and non-verbal speech. With strong capabilities of MLLMs, researchers introduced
 127 them into the field of visual content generation (Wu et al., 2024c; Xia et al., 2023; Wu et al., 2024b;
 128 Koh et al., 2023; Fu et al., 2024; Huang et al., 2023b), world modeling (Wu et al., 2024a; Ge et al.,
 129 2024; Mai et al., 2024), and embodied AI (Driess et al., 2023; Li et al., 2023b; Mu et al., 2023; Szot
 130 et al., 2024). In image generation and editing, various works (Koh et al., 2023; Fu et al., 2024; Huang
 131 et al., 2023b) use VLM to guide diffusion models. However, in the audio domain, existing methods
 132 have not exploited the reasoning capabilities of audio language models.

133 **Diffusion-based Image Editing.** In contrast to text-to-image generation, image editing focuses on
 134 altering specific elements or attributes within an image while preserving the contents of the remaining
 135 image. Diffusion models have been widely used in image editing tasks (Couairon et al., 2022; Hertz
 136 et al., 2022; Hui et al., 2024) by altering the inversion process, which produces a latent representation
 137 that can reconstruct the image through the generative process. SDEdit (Meng et al., 2022) first adds
 138 noise to the source image, and then subsequently denoises the image through the SDE to produce the
 139 target image. P2P (Hertz et al., 2022) adjusts the cross-attention features according to the difference
 140 between the source and target captions to generate the target images. Based on this, IP2P (Brooks
 141 et al., 2023) finetuned a diffusion model on edit image triplets to enable image editing with simple
 142 natural language instructions. Following works (Geng et al., 2023; Hui et al., 2024) further scale
 143 up the dataset to support more capable and generalized models. Furthermore, later works (Fu et al.,
 144 2024; Huang et al., 2023b) use vision-language models to guide diffusion models for image editing.

3 METHOD

145 In this section, we first define the task and notations. We then introduce our proposed framework
 146 SmartDJ that combines an Audio Language Model for interpreting declarative editing instructions
 147 with a diffusion model for executing sequential audio edits. Finally, we describe a scalable data
 148 generation pipeline powered by LLMs to support supervised training and evaluation.

3.1 PROBLEM DEFINITION AND NOTATIONS

149 Let a_0 denote the original audio waveform, which contains multiple audio events (e.g., *cat meowing*,
 150 *rainfall*), as shown in Fig. 1. We define a *declarative editing instruction* \mathcal{P} as a natural language
 151 description of a desired transformation of the overall audio scene. Such instructions are typically
 152 declarative: they specify what the desired outcome should be (e.g., "*make it sound like a quiet*
 153 *morning in a sunny forest*" or "*transform this into an indoor library setting*"), but do not explicitly
 154 prescribe the individual operations. Since \mathcal{P} is a declarative instruction, it must be decomposed into a
 155 detailed sequence of atomic editing steps $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$, where each step s_i either modifies an
 156 existing audio event in a_0 or introduces a new event needed to fulfill \mathcal{P} . We denote a_i ($i = 1, 2, \dots, n$)
 157 as the intermediate audio after applying step s_i , where a_0 is the original input and a_n is the final
 158 edited audio.

162 Specifically, each step s_i either modifies an existing audio event or introduces a new event required
 163 to satisfy \mathcal{P} . The atomic editing operations considered in this work are:

164

- 165 • **Add** : Mix a new sound event into the scene (e.g., inject bird chirps).
- 166 • **Remove** : Delete an existing sound event (e.g., remove car engine noise).
- 167 • **Extract** : Isolate a particular sound event from the original audio while removing others.
- 168 • **Volume adjust** : Adjust the volume of a specific event.
- 169 • **Change direction** : Modify the spatial location of an event.
- 170 • **Time shift** : Adjust the sound timing by x seconds.
- 171 • **Add reverberation** : Add reverberation to a single source.
- 172 • **Timbre adjust** : Modify the timbre of a single sound source.

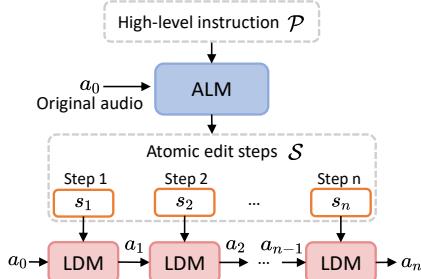
173 The goal of the editing is to produce a target edited audio clip a_n by applying the sequence of
 174 edits \mathcal{S} to a_0 . Importantly, this editing formulation must preclude shortcut solutions that ignore the
 175 original input (e.g., re-generating an entirely new clip from scratch). Instead, the edited audio a_n
 176 must preserve all unedited content from a_0 while achieving the requested audio scene transformation.
 177 Please refer to Appendix A.3 for more details.

178 3.2 SMARTDJ FRAMEWORK

181 SmartDJ consists of an Audio Language Model (ALM) and a Latent Diffusion Model (LDM). We
 182 leverage the ALM as a planner to interpret declarative instructions and generate a sequence of
 183 atomic editing steps. The LDM editor then executes these
 184 steps sequentially to transform the original audio. As illus-
 185 trated in Fig. 2, the ALM takes the original audio a_0 and
 186 the declarative editing instruction \mathcal{P} as input, and outputs
 187 a sequence of atomic editing steps $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$.
 188 These editing steps are then executed sequentially by
 189 the LDM, producing intermediate results a_1, a_2, \dots, a_n ,
 190 where a_n is the final edited audio. The overall process is
 191 formulated as:

$$\{s_1, s_2, \dots, s_n\} = \text{ALM}(a_0; \mathcal{P}) \quad (1)$$

$$a_i = \text{LDM}(a_{i-1}; s_i), i = 1, 2, \dots, n \quad (2)$$



192 Figure 2: SmartDJ framework overview

193 3.3 AUDIO LANGUAGE MODEL FOR ATOMIC EDITING STEPS GENERATION

196 The Audio Language Model (ALM) takes as input the original audio clip and the declarative editing
 197 instruction and generates a sequence of atomic editing steps. As shown at the top of Fig. 3, we
 198 first encode a_0 using a pretrained audio encoder (i.e., CLAP (Wu et al., 2023)) to obtain an audio
 199 embedding z_a , which is injected into the ALM via adapter layers. In parallel, the instruction \mathcal{P} is
 200 tokenized and encoded as a sequence of embeddings (p_1, p_2, \dots, p_k) , which serve as the textual
 201 context for the ALM.

202 Our ALM is trained in an auto-regressive manner to generate the token sequence corresponding to
 203 the atomic editing steps \mathcal{S} , by minimizing the following objective:

$$\mathcal{L}_{\text{ALM}} = - \sum_{t=1}^l \log P_{\theta}(r'_t = r_t \mid z_a, r_{1:t-1}, p_{1:k}), \quad (3)$$

208 where r and r' are the ground truth and predicted text tokens for the atomic editing steps \mathcal{S} , l is the
 209 length of the tokens, and θ denotes the model parameters. To enable efficient fine-tuning, we freeze
 210 the parameters of the CLAP audio encoder, apply Low-Rank Adaptation (LoRA) (Hu et al., 2022) to
 211 a small subset of the LLM layers (Ghosh et al., 2025b), and fully fine-tune the adapter layers.

212 **Separate training.** We train the ALM and LDM as independent modules rather than end-to-end.
 213 This enables human-in-the-loop editing, where users can easily intervene at the level of generated
 214 natural-language-based atomic steps before LDM editor inference. Besides, it makes training and
 215 computation more efficient and promotes modularity. This design allows for different ALMs or
 LDMs to be swapped in or out with minimal re-training effort, making it both practical and extensible
 for diverse editing scenarios.

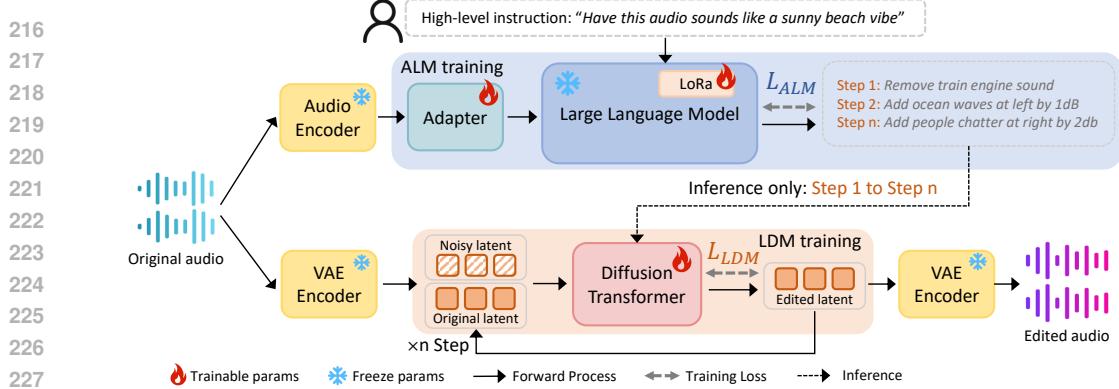


Figure 3: **SmartDJ framework**. Our method incorporates an ALM as an edit planner that understands both the original audio and the declarative instructions to produce atomic edit steps. These atomic steps are then fed into an LDM editor to edit the audio sequentially. The ALM and LDM modules are trained separately.

3.4 SEQUENTIAL STEREO AUDIO EDITING WITH LATENT DIFFUSION MODEL

The Latent Diffusion Model (LDM) in our framework performs audio editing conditioned on the atomic editing steps \mathcal{S} . To support this, we adopt a latent diffusion architecture (Hai et al., 2024; Rombach et al., 2022) and extend it to enable editing of stereo audio with spatial effects.

Stereo Audio VAE. Given a stereo audio signal $a \in \mathbb{R}^{2 \times L}$, where L is the number of time-domain samples at two audio channels (left and right). The audio Variational Autoencoder (VAE) encodes a into a latent representation $\hat{a} \in \mathbb{R}^{C \times L'}$, where C and L' denote the number of latent channels and the temporal length of the latent sequence. Similar to DAC (Kumar et al., 2023) and Stable-Audio-Open (Evans et al., 2024b), our audio VAE is based on a 1D-CNN autoencoder with a continuous VAE bottleneck and snake activation functions. The resulting latent \hat{a} has a dimension of $C = 128$ and length of $L' = L/480$, resulting a compression ratio of $7.5 \times$.

Latent Diffusion Model: Our diffusion model conditions on both the text description s_i at the i -th editing step and the latent representation of the audio from the previous step, \hat{a}_{i-1} , to generate the updated latent \hat{a}_i . We use the FLAN-T5 (Chung et al., 2024) text encoder $E_{\text{text}}()$ to convert s_i to text embeddings. At each editing step, we initialize a randomly noised latent $\hat{a}'_i \in \mathbb{R}^{C \times L'}$, which is concatenated with \hat{a}_{i-1} to form the input $[\hat{a}_{i-1}; \hat{a}'_i] \in \mathbb{R}^{2C \times L'}$ to the diffusion model. The model is conditioned on $E_{\text{text}}(s_i)$ via cross-attention layers, and the diffusion timestep t is incorporated through a modified AdaLN module (Hai et al., 2024) to reduce model parameters. We implement a Diffusion Transformer (DiT) that learns to denoise the latent \hat{a}'_i across multiple timesteps by predicting the added noise. Let ϵ denote the true added Gaussian noise, and let $\epsilon_\theta(\cdot)$ be the predicted noise output by the model. The training objective is to minimize the following denoising loss:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I), t, s_i, \hat{a}'_i} \|\epsilon - \epsilon_\theta(t, E_{\text{text}}(s_i), [\hat{a}_{i-1}; \hat{a}'_i])\|_2. \quad (4)$$

During inference, we use DDIM sampling (Song et al., 2021) with classifier-free guidance (CFG), which has proven effective for text-guided generation and editing (Ho & Salimans, 2022). CFG steers the denoising process by interpolating between conditional and unconditional model predictions:

$$\tilde{\epsilon}_\theta = \omega \cdot \epsilon_\theta(t, E_{\text{text}}(s_i), [\hat{a}_{i-1}; \hat{a}'_i]) + (1 - \omega) \cdot \epsilon_\theta(t, \emptyset, [\hat{a}_{i-1}; \hat{a}'_i]), \quad (5)$$

where ω is the guidance scale and \emptyset denotes the text embedding of an empty string.

3.5 DECLARATIVE INSTRUCTION AUDIO EDITING DATASET CURATION

Since no public dataset features audio editing conditioned on declarative instructions, we develop a scalable data generation pipeline, as illustrated in Fig. 4a. For each data point, we first randomly sample K single-event audio clips from public datasets, each labeled with tags such as {"car engine", "bell ring", "goat bleat", ...}. We feed these labels into GPT-4o and prompt it to act as a sound designer: it designs a declarative editing instruction \mathcal{P} that transforms the original mix into a new audio scene (e.g., "Make this sound like a countryside morning" or "Make it sound like a busy train station on a sunny afternoon"). It then decomposes \mathcal{P} into a sequence of atomic

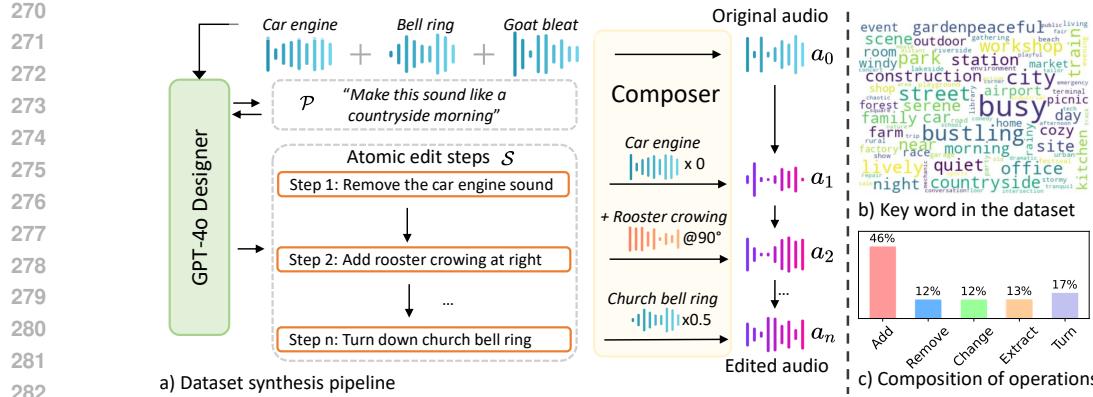


Figure 4: **Scalable data synthesis.** a) pipeline: we sample audio clips from databases with text labels and compose them into original audio a_0 ; These text labels are then fed into GPT-4o, which is prompted to design a declarative instruction \mathcal{P} and generate corresponding atomic steps \mathcal{S} . We compose the target audio a_1, a_2, \dots, a_n following the atomic steps sequentially with rule-based composer. b) Key words in the declarative instruction. c) The proportion of each single-step edit operation in the dataset.

edits $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$, including both add operations and modifications to existing events (e.g., remove, turn up/down, change sound direction).

Once the sound designer has provided the sequential edit operations, an audio signal processing based composer takes over. To generate edit audio pairs, the composer first synthesizes the initial audio a_0 by superposing the K audio clips. Spatial effects are rendered with direction-dependent phase and amplitude on two channels. For each atomic edit s_i , we proceed as follows:

- If s_i modifies an existing event, we update its volume level or sound direction.
- If s_i is an *add* operation, we retrieve a new clip from the database with a matching label.

Since each sampled audio event is *independently editable*, an edit step s_i that modifies an existing event can be converted into event-level parameter adjustments, without altering any other sources in the mixture a_{i-1} . For example, to simulate “*remove the car engine sound*” (Fig. 4), we set the car engine’s volume to zero to generate a_1 . To simulate “*add rooster crowing at right*”, we retrieve a rooster clip, apply the specified spatial effect, and superpose it on a_1 to obtain a_2 . To simulate “*turn down the dog bark*”, we reduce the volume of the corresponding clip. This allows us to generate a complete editing trajectory a_1, a_2, \dots, a_n by progressively updating event-level parameters and re-composing the audio scene. Our resulting dataset contains declarative editing instructions, atomic edit sequences, and the audio for the full editing trajectory. More details available in Appx. A.4.

4 EXPERIMENT

4.1 SETUP

Dataset. We use a combination of datasets including AudioCaps (Kim et al., 2019), VGGSound (Chen et al., 2020), FSD50k (Fonseca et al., 2021), ESC50 (Piczak, 2015), and WavCaps (Mei et al., 2024). We adopt dataset cleaning pipelines following prior work (Hai et al., 2024; Sun et al., 2024; Wang et al., 2023) by filtering events with noisy data labels or low clap scores. Each audio is trimmed or padded into 10 seconds with a sampling rate of 24 K. We sample 2-5 audio events and use GPT-4o to create 50k training pairs and 1k evaluation pairs of declarative audio editing data (from test set of VGGSound and AudioCaps) to train ALM planner and evaluate the whole editing pipeline. We present the keyword in the declarative instructions in Fig. 4b. We also expand the size of single-step editing data pairs (s_t, a_{t-1}, a_t) to 1M, where each step is a single atomic operation to train our LDM audio editor. We additionally generate 3k single step audio editing from AudioCaps to evaluate the results. Fig. 4c shows the proportion of each operation. Please find more details in the Appx. A.1.

Spatial Sound Effects. We use an interaural distance of 17 cm and adopt a cardioid directivity gain pattern to model the sound level differences between left/right ear when sound events are coming from different directions. To introduce room realism, we further use PyRoomAcoustics (Scheibler et al., 2018) to simulate room impulse response with reverberation time (RT60) (Sun et al., 2024) ranging



Figure 5: Examples of ALM’s output detailed steps. Our ALM module identifies events in the original audio clips and reasons on the given declarative instruction to produce aligned editing steps.

from 0.3 s to 1.2 s. Room size, microphone orientation, and source/mic placement are randomized during synthesis. More details room acoustic simulation and binaural rendering in Appx. A.2

Metrics. To evaluate edit quality and diversity, we use common metrics in audio generation and editing (Liu et al., 2023a; Wang et al., 2023), including Fréchet Distance (FD), Kullback-Leibler divergence (KL), Fréchet Audio Distance (FAD), Inception Score (IS), and Log-Spectral Distance (LSD). We use CLAP score to measure the semantic similarities between the edited audio and the text prompt. For spatial audio, we calculate GCC MSE (GCC) based on Generalized Cross-Correlation with Phase Transform (GCC-PHAT), and use StereoCRW (Chen et al., 2022) to produce stereo audio features to evaluate CRW MSE (CRW) and Fréchet Stereo Audio Distance (FSAD).

Baselines. To evaluate the declarative instruction based audio editing task, we first train an end-to-end version of Audit (Wang et al., 2023) that directly predicts the final-step edited audio a_n conditioned on the original audio a_o and declarative instruction \mathcal{P} in one step without ALM. We extend the mono-channel Audit to our binaural setting, where we stack the left and right channels of the mel-spectrograms as the model inputs. We also evaluate various zero-shot and training-required editing methods based on the ALM’s outputs to perform multi-step sequential editing. The zero-shot methods include SDEdit (Meng et al., 2022), DDIM Inversion (Mokady et al., 2022), ZETA (Manor & Michaeli, 2024), and AudioEditor (Jia et al., 2024). For a fair comparison, we replace these methods’ generation backbone with the current SOTA methods. In AudioEditor, we replace Affusion with BEWO (Sun et al., 2024) to support binaural editing. In SDEdit, DDIM Inversion, and ZETA, we use Stable-Audio-Open (Evans et al., 2024b) as the backbone. For sequential editing with a trainable editor, we also use an Audit baseline trained on single-step audio editing. In addition, we evaluate the same set of baseline methods on single-step audio editing tasks. More details available in Appx. B.1.

Implementation details. SmartDJ ALM is initialized from Audio Flamingo 2 (Ghosh et al., 2025a) with 3B parameters. During training, we freeze the AF-CLAP module and fine-tune the adapter layers and LLM with a LoRA module for 20 epochs with a batch size of 24. For LDM, it uses velocity prediction with Zero-SNR, and CFG rescaling technique (Lin et al., 2024) to adjust the magnitude of the predicted velocity and avoid over-exposure. It is trained on single-step editing data with a batch size of 256 of 500k iterations. 10% text is replaced with empty strings to enable unconditional modeling. The learning rates for the ALM and LDM training are 1e-5 and 5e-5 with AdamW. All experiments are conducted with four NVIDIA L40S GPUs. More details available in Appx. B.2.

4.2 RESULTS

Declarative instruction audio editing task. We first show inference examples from our ALM module in Fig. 5. The ALM-generated atomic editing steps accurately align both the original audio contents and the declarative editing instructions. For example, it correctly removes audio event *engine roar* when transferring audio scene into a *busy family home* vibe. It also removes *people whistle* and adds *pages turning* to enhance the immersion of being in an *old library*. More examples in Appx. C.1.

We present the results of the declarative instruction editing task in Tab. 1. The first row is the end-to-end Audit baseline, which is directly trained on declarative instructions and the final target audio, showing the worst performance overall. Since no prior method can interpret the declarative instructions, we use the same set of ALM-generated atomic steps to guide all audio editing models in the multi-step evaluation, including the baselines and our method. We compare the edited audios from each method with 1k reference audios. Our method achieves the lowest metric in FD, FAD, LSD, and comparably low in KL, indicating the smallest discrepancy from the reference audios. It

Framework	Method	Training	Speed	FD ↓	FAD ↓	KL↓	LSD↓	IS ↑	CLAP↑
w/o ALM	Audit	✓	2.07s	39.6	10.07	3.13	1.96	3.29	0.125
	SDEdit	✗	301s (74.6s)	27.3	3.73	3.26	2.25	6.66	0.188
	DDIM	✗	331s (82.1s)	34.3	9.49	4.07	2.23	3.97	0.076
	ZETA	✗	356s (88.2s)	28.8	3.75	2.93	2.24	7.72	0.224
	AE	✗	406s (101s)	27.6	5.02	3.22	2.11	8.91	0.211
	Audit	✓	11.6s (2.07s)	29.4	5.71	2.81	1.51	3.95	0.197
	SmartDJ (Ours)	✓	13.1s (2.40s)	14.7	1.53	2.85	1.42	8.36	0.238

Table 1: Quantitative results of the whole pipeline from declarative instructions to audio editing. Speed in () is the time for a single-step edit. AE denotes AudioEditor; DDIM denotes DDIM Inversion.

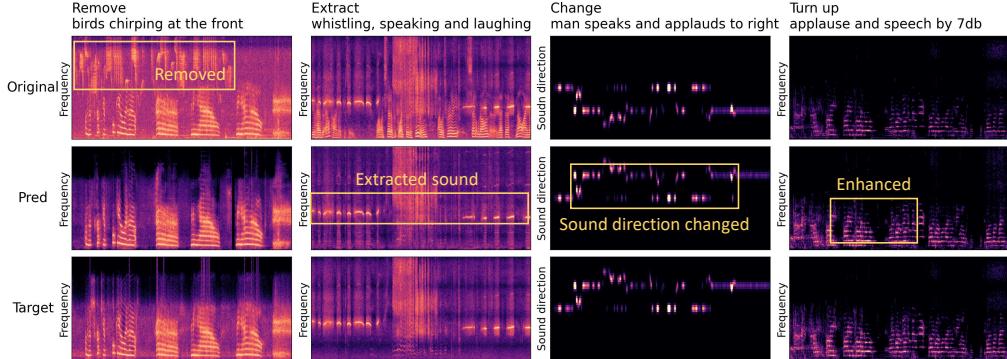


Figure 6: Examples on Remove, Extract, Change, Turn up operations. SmartDJ edited audio are closely aligned with the ground truth. Yellow boxes highlight edited sound components.

also delivers a high IS metric, showing strong audio quality and diversity. In addition, the highest CLAP score demonstrates the best semantic alignment between the edited audio and instruction.

Inference time. We also provide inference speed analysis in Tab. 1. The inference speed reported outside the brackets denote the total time to complete the entire declarative instruction edit, while the values inside indicate the per-round inference time of LDM. On average, SmartDJ’s ALM requires 4.8s to generate one set of atomic editing instructions. With multi-round reasoning, SmartDJ completes a full edit in 13.1s, which is significantly faster than training-free baselines. Our approach is slower compared with end-to-end Audit (2.07s) or Audit with ALM (11.6s), but this trade-off yields substantially better editing quality and alignment with target instructions.

Single-step audio editing. We present the results of single-step audio editing operations. Tab. 2a shows results on `add`. Our method consistently outperforms baseline methods in the edited audio, including better similarities to the ideal target audio (lowest FD, FAD and KL), and higher quality and diversity shown by the highest IS. Furthermore, stronger spatial metrics also indicate that the stereo audio characteristics are preserved better by SmartDJ.

Tab. 2a also shows the performance on `remove` and `extract` tasks. The results clearly indicate that our method delivers the best performance in aligning with the ground truth edited audio across both operations. We show the results on `Volume`, `Time`, `Reverberation`, `Timbre` and `change sound direction` tasks in Tab. 2b and 2c. Our method again shows stronger performance over Audit and training-free baselines, which demonstrates SmartDJ has better fine-grained manipulation in audio event properties. This is because Audit’s VAE operates on mel-spectrograms and discards phase information, which is critical for spatial cues. SmartDJ employs a diffusion transformer, which provides stronger long-range temporal modeling and richer cross-attention conditioning. These architectural upgrades allow edits to be both semantically precise and spatially coherent. Some examples of spectrogram visualization are shown in Fig. 6. More qualitative comparisons can be found in Appx. C.3.

Human evaluations. We conducted a blind, pairwise A/B preference survey with 19 participants. Each participant evaluated 20 randomly sampled pairs (10 declarative editing, 10 single-step editing). Each data pair consists of the original audio, the editing prompt, SmartDJ edited result, and edited results from a random competing baseline. In each pairwise comparison, the participants were asked

432	433	Method	Add							Remove/Extract						
			FD↓	FAD↓	KL↓	LSD↓	IS↑	GCC↓	CRW↓	FSAD↓	FD↓	FAD↓	KL↓	LSD↓	GCC↓	CRW↓
434	SDEdit	33.0	3.92	2.59	2.01	4.71	143.3	131.4	0.42	46.6	4.64	2.38	1.89	<u>159.5</u>	157.9	0.45
435	DDIM	36.9	6.28	2.64	2.02	4.57	<u>131.2</u>	<u>116.1</u>	<u>0.11</u>	53.9	5.69	2.75	1.85	159.6	<u>138.5</u>	<u>0.29</u>
436	ZETA	37.6	<u>3.64</u>	2.46	1.71	5.04	143.3	127.6	0.52	<u>40.6</u>	3.78	<u>1.92</u>	1.99	161.7	155.9	0.43
437	AE	<u>30.5</u>	3.66	2.13	1.84	<u>5.51</u>	133.5	145.8	0.55	47.1	<u>3.65</u>	2.43	2.01	164.0	165.6	0.58
438	Audit	36.5	4.49	<u>1.95</u>	<u>1.42</u>	4.37	145.7	136.3	0.31	54.5	7.56	2.12	1.65	189.2	204.5	1.07
439	SmartDJ	22.7	1.82	1.39	1.35	5.96	76.5	41.3	0.03	26.0	2.57	1.03	<u>1.71</u>	15.9	5.5	0.02

(a) Editing operation add and remove/extract

440	Operation	Method	Add							Remove/Extract							
			FD↓	FAD↓	KL↓	LSD↓	GCC↓	CRW↓	FSAD↓	FD↓	FAD↓	KL↓	LSD↓	GCC↓	CRW↓	FSAD↓	
441	Volume	Audit	47.1	5.6	1.51	1.04	136.8	139.0	0.89	SDEdit	41.47	3.83	1.78	1.42	165.6	169.56	0.49
442		SmartDJ	11.8	1.0	0.27	1.01	23.3	2.86	0.01	DDIM	58.0	7.32	2.85	1.30	193.71	168.92	0.17
443	Time	Audit	31.91	5.11	1.35	1.31	127.9	125.9	0.87	ZETA	39.6	3.35	1.38	1.54	158.78	173.81	0.42
444		SmartDJ	10.01	1.25	0.40	1.12	1.82	25.96	0.10	AE	46.5	4.95	1.85	1.51	95.04	133.12	0.32
445	Reverb	Audit	31.71	3.91	1.40	1.09	135.2	135.9	0.84	Audit	42.5	4.9	1.34	1.02	170.3	163.5	0.99
446		SmartDJ	8.68	0.8	0.35	1.04	3.87	32.2	0.11	SmartDJ	13.2	0.90	0.35	1.10	61.4	38.8	0.02
447	Timbre	Audit	31.42	3.68	1.45	1.06	132.1	127.5	0.81								
448		SmartDJ	10.20	0.98	0.47	1.01	6.67	23.16	0.12								

(b) Editing operation Volume, Time, Reverb and Timbre (c) Editing operation change sound direction

Table 2: Quantitative results on all individual audio editing operations.

Complex edit quality			Complex edit alignment		
SmartDJ	90.41%	Audit	SmartDJ	93.15%	Audit
SmartDJ	95.52%	AE	SmartDJ	91.04%	AE
SmartDJ	80.00%	ZETA	SmartDJ	87.00%	ZETA
Single step quality			Single step alignment		
SmartDJ	95.35%	Audit	SmartDJ	96.76%	Audit
SmartDJ	86.27%	AE	SmartDJ	84.92%	AE
SmartDJ	77.36%	ZETA	SmartDJ	88.89%	ZETA

Figure 7: User study results. In both audio quality and text/audio alignment, SmartDJ is consistently preferred over baselines in the declarative instruction editing task and single-step tasks.

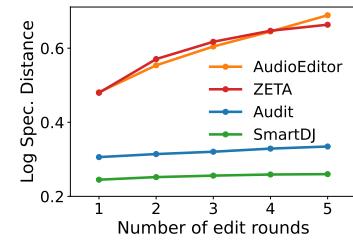


Figure 8: Similarity with original audio after multiple editing rounds.

to select one of the edited audio with higher quality, better alignment to the text or spatial instructions, and better alignment to the original audio. We separate the evaluation for declarative instruction audio editing and single-step editing in Fig. 7. In both tasks, SmartDJ is much preferred over all competing methods **with a higher win-rate**. Our method delivers the best audio quality and the best alignment with both declarative instructions and single-step instructions. **We also use a two-sided Z-test and find SmartDJ demonstrates statistical significance ($p < 0.001$).** We find that our 95% confidence intervals further confirm robustness, with all lower bounds (ranging from 66.1% to 94.2%) exceeding the random chance threshold of 50%. More details of the user study are listed in Appx. C.4.

4.3 ABLATION STUDIES

Performance of ALM generate editing steps. We use BERTScore (Zhang et al., 2019) to evaluate the ALM generated results. It measures semantic similarity between two sentences with a pretrained BERT model. For our task, each detailed editing plan consists of multiple atomic steps. To ensure fair evaluation, we group atomic operations by type (add, remove and so on), and compare only between matching operation types in the GT and generated plan. For example, GT “add” steps are compared only against generated “add” steps, and similarly other operations. We then compute BERTScore for each operation type and report the average across all operation groups (results in 91.8% Precision, 92.0% Recall and 91.5% of F1). We also provide some failure case analysis in Appx. C.2

Audio quality over multi-round editing. Since the declarative instruction audio editing task involves a sequence of editing, unchanged content must remain intact after multiple steps. To evaluate this, we design a “round-trip” edit experiment: we perform the operations “add the sound of \mathcal{A} ” and “remove the sound of \mathcal{A} ” on an audio clip for five rounds, where \mathcal{A} is a pseudo audio label. For an ideal audio editor, this sequence of round-trip operation should exactly reconstruct the original audio. We measure the LSD between each round’s edited output with the original audio clip (Fig. 8). SmartDJ consistently achieves the lowest LSD, indicating the smallest drift from the original content. This demonstrates that our method preserves the unedited audio content under repeated editing actions.

Study Objectives	Variation	FD \downarrow	FAD \downarrow	KL \downarrow	LSD \downarrow	IS \uparrow	CLAP \uparrow
ALM module	w/o ALM	23.6	3.14	2.91	1.84	4.63	0.137
	LTU+SmartDJ editor	16.5	1.58	3.00	1.79	8.10	0.204
	Caption+GPT-4o+SmartDJ editor	16.8	2.70	2.96	1.45	6.35	0.184
	w/ SmartDJ ALM	14.7	1.53	2.85	1.42	8.36	0.238
Editing order	Add \rightarrow Modify \rightarrow Remove	14.9	1.59	2.88	1.48	8.09	0.234
	Random order	14.7	1.56	2.88	1.47	8.16	0.237
	Remove \rightarrow Modify \rightarrow Add	14.7	1.53	2.85	1.42	8.36	0.238
Extract operation	AudioSep (Liu et al., 2024b)	27.1	2.86	0.88	1.63	N/A	N/A
	SmartDJ	25.7	2.55	0.78	1.71	N/A	N/A

Table 3: Model ablations.

Effectiveness of ALM. We conduct an ablation by removing the ALM module and training a variant of the LDM end-to-end. As shown in Tab. 3, SmartDJ performs significantly worse without ALM, showing the importance of ALM’s intermediate reasoning capabilities. ALM enables the model to produce semantically coherent edits aligned with the declarative instruction and the original audio.

Choice of ALM. To assess the sensitivity of SmartDJ to the choice of ALM backbone, we replaced our AF2-based ALM with LTU (Gong et al., 2023) while keeping the editor unchanged. As shown in Tab. 3, using LTU results in slightly worse performance. We attribute this gap primarily to the fact that our ALM is pretrained on a much larger and more diverse corpus, giving it stronger performance on decomposing instructions. Nevertheless, the LTU+SmartDJ editor variant still produces reasonable results. This indicates that our framework is generalizable to different model ALM architectures.

Can Caption Model + LLM replace ALM? We evaluate a captioning-based alternative pipeline, using a DCASE captioning model (Xu, 2024) to describe the audio and GPT-4o to produce step-wise instructions executed by SmartDJ editor. As shown in Tab. 3, this captioning-based pipeline performs consistently worse than SmartDJ. We attribute the performance drop to two factors: (1) Captioning models summarize only dominant events and often miss fine-grained or overlapping sounds. (2) Missing or misidentified events in the caption propagate through the LLM’s reasoning, compounding into hallucinated or non-executable editing instructions with cumulative errors. But future works can explore using a strong audio captioning model to directly get the audio events and prompt LLM to achieve the declarative editing.

Editing order. We adopt a simple ordering strategy of remove \rightarrow modify (volume/direction) \rightarrow add, which intuitively avoids removing newly inserted content. To test the impact of ordering, we also experiment with two alternatives: randomized and reversed order. As shown in Tab. 3, both alternatives result in only marginal degradation compared to our default ordering. This indicates that the editing order has minimal influence on the final results, suggesting that ALM-generated instructions rarely contain conflicting operations.

Compare with sound separation model. We compare SmartDJ with a recent sound separation model AudioSep (Liu et al., 2024b) on the Extract operation. Ours achieves comparable or slightly better performance (Tab. 3). This suggests that while our model is designed for general-purpose audio editing, it is competitive on target sound separation tasks.

5 DISCUSSION

Conclusion. We presented SmartDJ, the first framework for declarative instruction guided stereo audio editing that utilizes the reasoning capability of audio language models and strong editing capabilities of the latent diffusion model. Our approach produces atomic editing steps and executes them sequentially to achieve perceptually realistic stereo audio transformations. Extensive evaluations on both subjective audio metrics and human perceptual studies demonstrate that SmartDJ outperforms prior methods, and preserves spatial fidelity in complex scenes.

Limitations. Supporting a new task-specific editing operation on the LDM requires retraining the diffusion model. However, these edits can usually be achieved through a combination of our proposed atomic steps. Besides, a future direction is to implement an end-to-end joint training strategy of ALM and LDM that combines reasoning with audio editing.

540

6 ETHICS STATEMENT

541
 542 Our work focuses on audio editing for research and creative applications such as immersive media,
 543 conferencing, and sound design. The dataset is generated from publicly available sound event libraries
 544 and synthetic mixing, without personal or sensitive recordings. We encourage responsible use aligned
 545 with academic and creative purposes.

546
 547

7 REPRODUCIBILITY STATEMENT

548
 549 We provide detailed descriptions of the model architectures, training objectives, and data generation
 550 pipeline in the main paper and appendix. Hyperparameters, training configurations, and dataset con-
 551 struction details are included to ensure reproducibility. Code, pretrained models, and the synthesized
 552 dataset will be released upon acceptance to facilitate replication of our results.

553
 554

REFERENCES

555 Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel
 556 Lenc, Arthur Mensch, Katie Millicah, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan
 557 Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian
 558 Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo
 559 Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language
 560 model for few-shot learning. In *Proceedings of the 36th International Conference on Neural*
 561 *Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.
 562 ISBN 9781713871088.

563
 564 Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image
 565 editing instructions, 2023. URL <https://arxiv.org/abs/2211.09800>.

566
 567 Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A large-scale audio-
 568 visual dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and*
 569 *Signal Processing (ICASSP)*, pp. 721–725. IEEE, 2020.

570
 571 Ziyang Chen, David F Fouhey, and Andrew Owens. Sound localization by self-supervised time delay
 572 estimation. In *European Conference on Computer Vision*, pp. 489–508. Springer, 2022.

573
 574 Ziyang Chen, Prem Seetharaman, Bryan Russell, Oriol Nieto, David Bourgin, Andrew Owens, and
 575 Justin Salamon. Video-guided foley sound generation with multimodal controls. *arXiv preprint*
 576 *arXiv:2411.17698*, 2024.

577
 578 Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and
 579 Jingren Zhou. Qwen-audio: Advancing universal audio understanding via unified large-scale
 580 audio-language models. *arXiv preprint arXiv:2311.07919*, 2023.

581
 582 Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li,
 583 Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language
 584 models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

585
 586 Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based
 587 semantic image editing with mask guidance, 2022. URL <https://arxiv.org/abs/2210.11427>.

588
 589 Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter,
 590 Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar,
 591 Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc
 592 Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied
 593 multimodal language model, 2023. URL <https://arxiv.org/abs/2303.03378>.

594
 595 Zach Evans, CJ Carr, Josiah Taylor, Scott H Hawley, and Jordi Pons. Fast timing-conditioned latent
 596 audio diffusion. In *Forty-first International Conference on Machine Learning*, 2024a.

594 Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio
 595 open. *arXiv preprint arXiv:2407.14358*, 2024b.
 596

597 Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open
 598 dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language
 599 Processing*, 30:829–852, 2021.

600 Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding
 601 instruction-based image editing via multimodal large language models, 2024. URL <https://arxiv.org/abs/2309.17102>.
 602

603 Zhiqi Ge, Hongzhe Huang, Mingze Zhou, Juncheng Li, Guoming Wang, Siliang Tang, and Yueting
 604 Zhuang. WorldGPT: Empowering LLM as multimodal world model. In *ACM Multimedia 2024*,
 605 2024. URL <https://openreview.net/forum?id=G1tsqarGAw>.
 606

607 Zigang Geng, Binxin Yang, Tiansai Hang, Chen Li, Shuyang Gu, Ting Zhang, Jianmin Bao, Zheng
 608 Zhang, Han Hu, Dong Chen, and Baining Guo. Instructdiffusion: A generalist modeling interface
 609 for vision tasks, 2023. URL <https://arxiv.org/abs/2309.03895>.
 610

611 Sreyan Ghosh, Zhifeng Kong, Sonal Kumar, S Sakshi, Jaehyeon Kim, Wei Ping, Rafael Valle, Dinesh
 612 Manocha, and Bryan Catanzaro. Audio flamingo 2: An audio-language model with long-audio
 613 understanding and expert reasoning abilities. *arXiv preprint arXiv:2503.03983*, 2025a.
 614

615 Sreyan Ghosh, Zhifeng Kong, Sonal Kumar, S Sakshi, Jaehyeon Kim, Wei Ping, Rafael Valle, Dinesh
 616 Manocha, and Bryan Catanzaro. Audio flamingo 2: An audio-language model with long-audio
 617 understanding and expert reasoning abilities, 2025b. URL <https://arxiv.org/abs/2503.03983>.
 618

619 Yuan Gong, Hongyin Luo, Alexander H Liu, Leonid Karlinsky, and James Glass. Listen, think, and
 620 understand. *arXiv preprint arXiv:2305.10790*, 2023.
 621

622 Jiarui Hai, Yong Xu, Hao Zhang, Chenxing Li, Helin Wang, Mounya Elhilali, and Dong Yu. Eza-
 623 dio: Enhancing text-to-audio generation with efficient diffusion transformer. *arXiv preprint
 624 arXiv:2409.10819*, 2024.

625 Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-
 626 to-prompt image editing with cross attention control, 2022.
 627

628 Mojtaba Heydari, Mehrez Souden, Bruno Conejo, and Joshua Atkins. Immersediffusion: A generative
 629 spatial audio latent diffusion model. *arXiv preprint arXiv:2410.14945*, 2024.
 630

631 Mojtaba Heydari, Mehrez Souden, Bruno Conejo, and Joshua Atkins. Immersediffusion: A generative
 632 spatial audio latent diffusion model. In *ICASSP 2025-2025 IEEE International Conference on
 633 Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.
 634

635 Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*,
 636 2022.
 637

638 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,
 639 Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
 640

641 Jiawei Huang, Yi Ren, Rongjie Huang, Dongchao Yang, Zhenhui Ye, Chen Zhang, Jinglin Liu, Xiang
 642 Yin, Zejun Ma, and Zhou Zhao. Make-an-audio 2: Temporal-enhanced text-to-audio generation.
 643 *arXiv preprint arXiv:2305.18474*, 2023a.
 644

645 Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao
 646 Zhou, Chao Dong, Rui Huang, Ruimao Zhang, and Ying Shan. Smartedit: Exploring complex
 647 instruction-based image editing with multimodal large language models, 2023b. URL <https://arxiv.org/abs/2312.06739>.
 648

649 Mude Hui, Siwei Yang, Bingchen Zhao, Yichun Shi, Heng Wang, Peng Wang, Yuyin Zhou, and
 650 Cihang Xie. Hq-edit: A high-quality dataset for instruction-based image editing. *arXiv preprint
 651 arXiv:2404.09990*, 2024.

648 Yuhang Jia, Yang Chen, Jinghua Zhao, Shiwan Zhao, Wenjia Zeng, Yong Chen, and Yong
 649 Qin. Audioeditor: A training-free diffusion-based audio editing framework. *arXiv preprint*
 650 *arXiv:2409.12466*, 2024.

651

652 Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. Audiocaps: Generating
 653 captions for audios in the wild. In *Proceedings of the 2019 Conference of the North American*
 654 *Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*
 655 *1 (Long and Short Papers)*, pp. 119–132, 2019.

656 Jaeyeon Kim, Heeseung Yun, and Gunhee Kim. Visage: Video-to-spatial audio generation. *arXiv*
 657 *preprint arXiv:2506.12199*, 2025.

658

659 Jing Yu Koh, Daniel Fried, and Ruslan Salakhutdinov. Generating images with multimodal language
 660 models, 2023. URL <https://arxiv.org/abs/2305.17216>.

661

662 Zhifeng Kong, Arushi Goel, Rohan Badlani, Wei Ping, Rafael Valle, and Bryan Catanzaro. Audio
 663 flamingo: A novel audio language model with few-shot learning and dialogue abilities. *arXiv*
 664 *preprint arXiv:2402.01831*, 2024a.

665

666 Zhifeng Kong, Arushi Goel, Rohan Badlani, Wei Ping, Rafael Valle, and Bryan Catanzaro. Audio
 667 flamingo: A novel audio language model with few-shot learning and dialogue abilities, 2024b.
 URL <https://arxiv.org/abs/2402.01831>.

668

669 Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-
 670 fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing*
 671 *Systems*, 36:27980–27993, 2023.

672

673 Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng
 674 Gao. Multimodal foundation models: From specialists to general-purpose assistants, 2023a. URL
<https://arxiv.org/abs/2309.10020>.

675

676 Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming
 677 Liu, and Hao Dong. Manipllm: Embodied multimodal large language model for object-centric
 678 robotic manipulation, 2023b. URL <https://arxiv.org/abs/2312.16217>.

679

680 Jinhua Liang, Huan Zhang, Haohe Liu, Yin Cao, Qiuqiang Kong, Xubo Liu, Wenwu Wang, Mark D
 681 Plumbley, Huy Phan, and Emmanouil Benetos. Wavcraft: Audio editing and generation with large
 682 language models. *arXiv preprint arXiv:2403.09527*, 2024.

683

684 Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and
 685 sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of*
 686 *computer vision*, pp. 5404–5411, 2024.

687

688 Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and
 689 Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv*
 690 *preprint arXiv:2301.12503*, 2023a.

691

692 Haohe Liu, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Qiao Tian, Yuping Wang, Wenwu
 693 Wang, Yuxuan Wang, and Mark D Plumbley. Audioldm 2: Learning holistic audio generation with
 694 self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*,
 695 2024a.

696

697 Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023b. URL
 698 <https://arxiv.org/abs/2304.08485>.

699

700 Huadai Liu, Tianyi Luo, Qikai Jiang, Kaicheng Luo, Peiwen Sun, Jialei Wan, Rongjie Huang, Qian
 701 Chen, Wen Wang, Xiangtai Li, et al. Omniaudio: Generating spatial audio from 360-degree video.
 702 *arXiv preprint arXiv:2504.14906*, 2025.

703

704 Xubo Liu, Qiuqiang Kong, Yan Zhao, Haohe Liu, Yi Yuan, Yuzhuo Liu, Rui Xia, Yuxuan Wang,
 705 Mark D Plumbley, and Wenwu Wang. Separate anything you describe. *IEEE/ACM Transactions*
 706 *on Audio, Speech, and Language Processing*, 2024b.

702 Xinji Mai, Zeng Tao, Junxiong Lin, Haoran Wang, Yang Chang, Yanlan Kang, Yan Wang, and
 703 Wenqiang Zhang. From efficient multimodal models to world models: A survey, 2024. URL
 704 <https://arxiv.org/abs/2407.00118>.

705 Hila Manor and Tomer Michaeli. Zero-shot unsupervised and text-based audio editing using ddpm
 706 inversion. *arXiv preprint arXiv:2402.10009*, 2024.

708 Xinhao Mei, Chutong Meng, Haohe Liu, Qiuqiang Kong, Tom Ko, Chengqi Zhao, Mark D Plumbley,
 709 Yuxian Zou, and Wenwu Wang. Wavcaps: A chatgpt-assisted weakly-labelled audio captioning
 710 dataset for audio-language multimodal research. *IEEE/ACM Transactions on Audio, Speech, and
 711 Language Processing*, 2024.

712 Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon.
 713 Sdedit: Guided image synthesis and editing with stochastic differential equations, 2022. URL
 714 <https://arxiv.org/abs/2108.01073>.

715 Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for
 716 editing real images using guided diffusion models, 2022. URL <https://arxiv.org/abs/2211.09794>.

719 Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhui Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng
 720 Dai, Yu Qiao, and Ping Luo. EmbodiedGPT: Vision-language pre-training via embodied chain of
 721 thought. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL
 722 <https://openreview.net/forum?id=IL5zJqfxAa>.

723 Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM
 724 international conference on Multimedia*, pp. 1015–1018, 2015.

725 Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-
 726 resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF confer-
 727 ence on computer vision and pattern recognition*, pp. 10684–10695, 2022.

729 Robin Scheibler, Eric Bezzam, and Ivan Dokmanić. Pyroomacoustics: A python package for audio
 730 room simulation and array processing algorithms. In *2018 IEEE international conference on
 731 acoustics, speech and signal processing (ICASSP)*, pp. 351–355. IEEE, 2018.

732 Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Inter-
 733 national Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=St1giarCHLP>.

735 Peiwen Sun, Sitong Cheng, Xiangtai Li, Zhen Ye, Huadai Liu, Honggang Zhang, Wei Xue, and Yike
 736 Guo. Both ears wide open: Towards language-driven spatial audio generation. *arXiv preprint
 737 arXiv:2410.10676*, 2024.

739 Andrew Szot, Bogdan Mazoure, Harsh Agrawal, R Devon Hjelm, Zsolt Kira, and Alexander T
 740 Toshev. Grounding multimodal large language models in actions. In *The Thirty-eighth Annual
 741 Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=0G15WxY6es>.

743 Qwen Team. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

744 Yuancheng Wang, Zeqian Ju, Xu Tan, Lei He, Zhizheng Wu, Jiang Bian, et al. Audit: Audio editing
 745 by following instructions with latent diffusion models. *Advances in Neural Information Processing
 746 Systems*, 36:71340–71357, 2023.

748 Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye HAO, and Mingsheng Long.
 749 ivideoGPT: Interactive videoGPTs are scalable world models. In *The Thirty-eighth Annual
 750 Conference on Neural Information Processing Systems*, 2024a. URL <https://openreview.net/forum?id=4TENzBftZR>.

752 Jiannan Wu, Muyan Zhong, Sen Xing, Zeqiang Lai, Zhaoyang Liu, Zhe Chen, Wenhui Wang,
 753 Xizhou Zhu, Lewei Lu, Tong Lu, Ping Luo, Yu Qiao, and Jifeng Dai. VisionLLM v2: An end-
 754 to-end generalist multimodal large language model for hundreds of vision-language tasks. In
 755 *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL
<https://openreview.net/forum?id=nvYDPF4LJK>.

756 Jianzong Wu, Xiangtai Li, Chenyang Si, Shangchen Zhou, Jingkang Yang, Jiangning Zhang, Yining
757 Li, Kai Chen, Yunhai Tong, Ziwei Liu, and Chen Change Loy. Towards language-driven video
758 inpainting via multimodal large language models, 2024c. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2401.10226)
759 2401.10226.

760 Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov.
761 Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption
762 augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and*
763 *Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

764

765 Bin Xia, Shiyin Wang, Yingfan Tao, Yitong Wang, and Jiaya Jia. Llmga: Multimodal large language
766 model based generation assistant. *arXiv preprint arXiv:2311.16500*, 2023.

767

768 Manjie Xu, Chenxing Li, Dan Su, Wei Liang, Dong Yu, et al. Prompt-guided precise audio editing
769 with diffusion models. *arXiv preprint arXiv:2406.04350*, 2024.

770

771 Xuenan Xu. Audiocaption. <https://github.com/wsntxxn/AudioCaption>, 2024.
772 GitHub repository, accessed 2025-11-21.

773

774 Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating
775 text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810 A DATASET CURATION PROCESS
811812 A.1 DATASET PREPARATION
813814 We construct our training corpus by merging several publicly-available audio dataset, including VGG-
815 Sound, AudioCaps, WavCaps, ESC-50, and FSD50K. Since some of these sets provide audio captions
816 rather than discrete audio label, we first convert every caption to audio labels with GPT-4o-mini
817 API. We only retain single-labeled audio clip and any clip whose caption maps to multiple events is
818 discarded. A CLAP model scores the semantic correspondence between the audio and its new label.
819 Samples with a similarity score below 0.3 are filtered out. The remaining clips from all sources are
820 finally mixed into one large dataset that we use for subsequent data curation.821 A.2 DETAILED IMPLEMENTATION OF SPATIAL AUDIO
822823 We simulate room impulse responses (RIRs) using PyRoomAcoustics (Scheibler et al., 2018) to
824 generate stereo audio in diverse environments. Rectangular rooms are created with horizontal
825 dimensions sampled from
826

827
$$R = [R_0, R_1, R_2], \quad R_0, R_1 \sim U(4, 10) \text{ m}, \quad R_2 = 3 \text{ m}, \quad (6)$$

828

829 All microphone and source positions are sampled uniformly inside the room with a minimum 0.2m
830 to each wall, and the microphone azimuth orientation is drawn at random.
831832 The reverberation time of RIR is sampled as $RT60 \sim U(0.3, 1.2) \text{ s}$, covering different level of
833 reverberation conditions.
834835 **Directional sampling.** We consider semantic source directions (left, front-left, front, front-right,
836 right) and map each label to an azimuth angle $\theta \in \{180^\circ, 135^\circ, 90^\circ, 45^\circ, 0^\circ\}$. To introduce variability,
837 we perturb the azimuth with Gaussian noise (standard derivation of 5°). This θ represents the relative
838 angle between the microphone orientation and the sound source.839 **Binaural rendering with cardioid gain.** We use a two-microphone array with spacing $d = 17\text{cm}$
840 to simulate human ears. Let $x(t)$ denotes the dry source signal, and let α be the incident angles
841 of the source relative to the left or right microphone normals. We use a cardioid gain pattern
842 $G(\alpha) = \frac{1}{2}(1 + \cos \alpha)$. The rendered left and right ear signals are
843

844
$$y_L(t) = x(t) * h_L(t), \quad (7)$$

845

846
$$y_R(t) = x(t) * h_R(t), \quad (8)$$

847

848 where $h_L(t)$ and $h_R(t)$ are the RIR synthesized at the left and right ear position, $*$ denotes convolution.
849 The synthesized RIR considers the directional dependent gain pattern when combining the acoustic
850 ray coming from different directions. As a result, the interaural time and level difference are encoded
851 within the synthesized RIR to create spatial audio experience.
852853 A.3 ATOMIC EDIT ACTIONS
854855 We explain the details on creating the single-step atomic edit data pairs. For this single-step audio
856 editing, we have an original audio a_{i-1} , a single atomic edit operation s_i . Base on atomic edit
857 operation s_i , we can generate the edited audio a_i .
858859 **Add.** Assume the original audio is a mix of audio content $A+B+C$. To *add* a new content into this
860 original audio, we sample a new audio content D from the database and mix the D with the original
861 audio $A+B+C+D$. The atomic template is "*Add the sound of {dog barking} at {right} with {3} db*".
862 The contents inside the {} can be changed to other sound events or sound attributions. We support
863 various sound direction (left, front and right) and dynamic volume adjustment.
864865 **Remove.** Given an original mix $A+B+C$, let B be the undesired source. We suppress B so the output
866 becomes $A+C$. The atomic template is Atomic template: "*Remove the sound of {bird chirping} {at*
867 *right}*". The directional phrase in braces is optional. If there are similar audio contents in the same
868 clip, the spatial features enables to manipulate it precisely.
869

864 **Extract.** Starting from the same mix $A+B+C$, we isolate one target source A and mute everything
 865 else, yielding only A . The atomic template is "*Extract the sound of {speaking} {at the right}*". The
 866 direction is optional.

867 **Turn up/down** To change loudness of a specific source B , we scale it by α , where $\alpha > 1$ is for "up"
 868 and $0 < \alpha < 1$ is for "down". The resulted audio clip is $A + \alpha B + C$. The atomic template is "*Turn*
 869 *{up / down} the sound of {engine rev} by {2} dB*". We also support a dynamic range of volume
 870 adjustment.

871 **Change sound direction.** We alter only the spatial cues of a specific source C , producing an edited
 872 version C' while leaving the other tracks untouched: $A+B+C'$. The atomic template used is "*Change*
 873 *the sound of {baby crying} {from front} to {right}*". The "from" clause could also be omitted.

874 **Time shift.** Given an original mix $A+B+C$, let B be the target source. We apply time shifting to B
 875 so the output becomes $A+B'+C$. The atomic template is Atomic template: "*Shift time of the sound*
 876 *of {bird chirping} {by 3 seconds}*".

877 **Add reverberations.** Given a mix $A+B+C$, we apply reverberation to a specific target source
 878 (e.g., B) by convolving it with a room impulse response (RIR). The resulting edited clip becomes
 879 $A + B' + C$, $B' = B * \text{RIR}_{\text{level}}$. We support three reverberation levels, each corresponding to an
 880 RIR with a different decay time (RT60):

- Low reverberation: $\text{RT60} \approx 0.4$ s
- Mid reverberation: $\text{RT60} \approx 0.8$ s
- High reverberation: $\text{RT60} \approx 1.2$ s

881 The atomic template is "*Add reverberation to the sound of {violin} {at left} of {low / mid / high}*
 882 *level*". A directional qualifier (e.g., "at left") is optional. The reverberation is applied only to the
 883 selected source, while all other sources remain unchanged.

884 **Timbre adjustment.** We modify the spectral characteristics of a specific source C while preserving
 885 its spatial cues and temporal structure. The resulting signal C' is obtained via frequency-domain
 886 filtering, and the output becomes $A+B+C'$. We support several interpretable timbre transformations
 887 commonly used in audio production:

- Bright: boost high frequencies > 3 kHz
- Dark: attenuate high frequencies > 3 kHz
- Warm: boost mid-high components (300 Hz–4 kHz)
- Cold: boost > 6 kHz and attenuate < 300 Hz
- Muffled: attenuate frequencies > 1.5 kHz

888 The atomic template is "*Change the timbre of the sound of {acoustic guitar} to {bright / dark /*
 889 *warm / cold / muffled}*". These adjustments operate at the source level, allowing precise sound-color
 890 manipulation even when multiple acoustically similar sources appear within the same audio clip.

903 A.4 COMPLEX AUDIO EDITING DATASET CURATION

904 In the dataset curation process, we first sample 2-5 audio labels in the database, with LLM randomly
 905 assigned volume and sound directions. We then call the GPT-4o batch API with sound sources and
 906 attributions. For each API call, we provide 15 sets of sound sources. Through API call, each set
 907 of sound sources will return a single data pair containing Declarative editing instruction and the
 908 corresponding atomic editing steps. We follow these atomic editing steps to manually generate the
 909 step-by-step target edited audio with the rules in A.3. We generate 50K data pairs for complex audio
 910 editing for training. We generate 1K data pairs from AudioCaps test/validation set for evaluation.

911 Starting from these 50K complex audio editing pairs, we collect the generated corresponding step-
 912 by-step atomic actions and produce about 200K single step audio editing pairs (s_i, a_{i-1}, a_i) . We
 913 further scale up the this dataset size to 1M to train the LDM audio editor and we also generate another
 914 1K extra audio data pairs to evaluate the performance of single step audio editing. This scaled-up
 915 single step audio editing dataset keeps some instructions with audio captions, improving the LDM's
 916 robustness to different audio contents in the atomic edit instructions. Tab. 4 shows a comparison

Dataset	Task	Hours	Number	Data Type	Format
Audio Generation Datasets					
ImmerseDiffusion	Text to Audio	6000	1.1M	Text + Audio	FOA
YT-Ambigen	Video to Audio	142	102K	Video + Audio	FOA
BEWO-1M (text)	Text to Audio	2800	1M	Text + Audio	Binaural
BEWO-1M (image)	Image to Audio	54	2.3K	Image + Audio	Binaural
Audio Editing Datasets					
Audit		1605	578K	Text + Audio	Mono
	Add	197	71k		
	Remove	197	71k		
	Replacement	139	50k		
	Inpainting	536	193k		
	Super-resolution	536	193k		
SmartDJ Single-Step Editing		2778	1M	Text + Audio	Binaural
	Add	763	275K		
	Remove	330	119K		
	Extract	305	110K		
	Volume adjust	239	86K		
	Direction adjust	305	110K		
	Timing adjust	278	100K		
	Add Reverb.	278	100K		
	Timbre adjust	278	100K		
SmartDJ Declarative Editing	Declarative editing	140	50K	Text + Audio	Binaural

Table 4: Summary of current audio generation and editing datasets.

between current audio generation and editing dataset, where ImmerseDiffusion (Heydari et al., 2025) and YT-Ambigen (Kim et al., 2025) are two current spatial audio generation work.

We provide the details of our *Base Prompt* for dataset curation as follow:

Prompt

You are an expert in spatial audio editing and sound design.

Your task is to generate complex audio editing instructions based on a given list of sound sources (labels). The sound sources will be provided as a list of full sentences (as strings), not character lists. Treat each sentence as a single atomic sound unit. Do not tokenize or split the sound sources into characters.

For example, if you are given: "a baby crying and a man talking; a bird is chirping; dog barking". You should consider "a baby crying and a man talking" as a complete sound source. "a bird is chirping" is another complete sound source and "dog barking" is another sound source. You then generate the step-by-step audio editing instructions based on the given complex instructions.

Task: you need to first brainstorm a complex audio editing instruction

- Imagine a realistic and creative soundscape editing for the given audios.
- You are not limited to the provided audio contents for the editing instruction. And the complex editing instructions should be brief.
- The complex editing instructions could be a soundscape transformation. For example:
 - Make this sound like it was recorded in a bookstore
 - Make this sound like a busy coffee shop
 - Make this sound like a train station
 - Make this sound like a forest at night
 - Make this sound like a beach
 - Make this sound like a sunny day
 - Craft this sound to feel like a park
 - Make this audio sound like a quiet farm

972 - - Make this audio sound like a firework show
 973
 974 - Your generated complex instructions can be broader than these provided examples. Use your
 975 imaginations!
 976 - But remember to keep them brief, the complex editing instructions ideally should not contain the actual
 977 sound sources.
 978 - Then, based on the given sound event, give me the detailed editing instructions.
 979
 You then generate detailed editing instructions based on the complex audio editing instruction.
 980 - Ensure the editing makes sense (e.g., no waves in a desert, no sheep indoors, rustling leaves in the forest,
 981 seas waves in the beach, no raining in the sunny day).
 982 - You are encouraged to remove the original audio contents, but you are required to maintain at least one
 983 sound source, not removing all of them.
 984 - Do not split or partially reference a sound source when applying operations.
 985 - Use a combination of simple operations (but NOT necessarily all of them). For example:
 986 - Add (e.g., thunderstorm, cat meowing) (for the add operation, you should add up to two additional
 987 sources that best align with the complex editing instructions)
 988 - - Remove (For remove, you should remove at least one sound sources (up to two sound sources), but
 989 you must keep at least one sound source!)
 990 - - Turn up/down (e.g., turn up/turn down the sound of xxx by xxx db (between 0 and 6db))
 991 - - Change sound direction.
 992 - Each "target" in remove/turn up/turn down/change must exactly match one of the provided "sound
 993 sources"
 994 - For the "add" operation:
 995 - - The "target" must clearly describe the new sound being added (e.g., "crowd chatter", "rain",
 996 "footsteps on gravel").
 997 - - The "effect" should specify the volume and direction (e.g., "at front by 4dB").
 998 - - Do NOT use "none", "null", or placeholder values as the target. The target must always be a
 999 descriptive label of the added sound.
 1000 - - The added sound should not duplicate any original sound source.
 1001 - The same operation can be repeated for multiple targets.
 1002 - When doing add, you can also have volume and direction attributes
 1003 - When editing existing sound sources, you can also have mixed attributes in terms of the volume and
 1004 sound directions.
 1005 - You must ensure each step logically contributes to the final transformation.
 1006
 Return the output in the following structured JSON format:
 1007 {
 1008 "sound sources": [..., ...], here you should put the sound sources you are given
 1009 "complex editing instruction": "...",
 1010 "atomic editing steps": [
 1011 {"operation": "add", "target": "...", "effect": "at xxx(left, front, right) by xxxdB"},
 1012 {"operation": "remove", "target": "...", "effect": "None"},
 1013 {"operation": "turn up/turn down", "target": "...", "effect": "xxxdB"},
 1014 {"operation": "change", "target": "...", "effect": "to xxx (left, right, or front)"},
 1015]
 1016 }
 1017
 1018 Do NOT break down sound sources into individual words or characters. Sound sources are complete strings
 1019 and must remain so in the JSON.
 1020
 1021
 1022
 1023
 1024
 1025

1026 **B IMPLEMENTATION DETAILS**
10271028 **B.1 BASELINES IMPLEMENTATION**
10291030 We present the following baseline methods to evaluate the complex audio editing task. One of the
1031 baselines is an end-to-end version of Audit without using ALM. All other baseline methods perform
1032 multi-step sequential editing with ALM’s atomic editing step outputs.1033 **End-to-End Audit.** We first train an end-to-end version of Audit that directly predicts the final-step
1034 edited audio a_n conditioned on the original audio a_0 and Declarative instruction \mathcal{P} in one step
1035 without ALM. We extend the mono-channel Audit to our binaural setting, where we stack the left
1036 and right channels of the mel-spectrograms as the model inputs. Follow the original implementation,
1037 we convert 10s of audio into mel-spectrograms with a size of 80×624 , using a hop size of 256, a
1038 window size of 1024, and mel-bins of size 80. We use the same model configurations in the original
1039 paper.1040 **SDEdit.** SDEdit is a zero-shot method that does not require training a new audio editing model. It
1041 uses an off-the-shelf text-to-audio (TTA) generation model, which we use Stable-Audio-Open, as it
1042 supports binaural audio generation. We use the default 200 total diffusion steps and start the reverse
1043 process from a timestep of 100. We use a classifier-free-guidance (CFG) scale of 7.5. The target
1044 caption is composed by concatenating the individual event captions after each editing step.1045 **DDIM Inversion.** Similar to SDEdit, we also use Stable-Audio-Open as the TTA generation model.
1046 We use the default 200 total diffusion steps and start the reverse process from a timestep of 100. We
1047 use a CFG scale of 7.5 for both the target and the source. The source text prompt and the target text
1048 prompt are composed by concatenating the individual event captions before and after each editing
1049 step, respectively.1050 **ZETA.** Similar to SDEdit and DDIM Inversion, we also use Stable-Audio-Open as the TTA generation
1051 model. We use the default 200 total diffusion steps and start the reverse process from a timestep of 100.
1052 We use a CFG scale of 7.5 for the target and a CFG scale of 1 for the source. The source text prompt and the target text
1053 prompt are composed by concatenating the individual event captions before and after each editing
1054 step, respectively.1055 **AudioEditor.** Specifically in AudioEditor, we replace Affusion with the spatial generator SpatialSonic
1056 in BEWO to support binaural editing. We use the default 100 total editing steps with 5 iterations
1057 per step to update text embedding for null-text inversion. For the addition task, we use the default
1058 punishment ratio (alpha) of -0.001. For the remove or extract task, we use the default punishment
1059 ratio (alpha) of 1.1060 **Audit with ALM.** To evaluate sequential editing with a trainable editor, we use an Audit baseline
1061 trained on single-step audio editing data with input audio a_{i-1} , atomic instruction s_i , and output
1062 edited audio a_i . The model and data configurations are the same with the end-to-end Audit baseline
1063 variant.1064
1065 **B.2 SMARTDJ IMPLEMENTATION**
10661067 **ALM.** Our ALM module is initialized from Audio Flamingo 2. This model contains an AF-CLAP
1068 audio encoder module to encode the mono-channel audio. The input 10-second audio is first resampled
1069 to 16KHz and transformed into a dense audio features $z_a \in \mathbb{R}^{64 \times 2048}$. The mono-channel CLAP
1070 encoder understands the audio semantics and sound events, which is enough to reason the atomic
1071 steps to make the edited audio semantically align with the Declarative instruction. This audio encoder
1072 is followed by a representation transformation layers that expand the model capacity. This module
1073 has three self-attention layers to the audio feature representation, each with 8 heads and a dimension
1074 of 2048. Following this, gated cross-attention layers are used to condition audio representations on
1075 the LLM. The LLM uses Qwen2.5-3B, a decoder-only causal LLM with 3B parameters, 36 hidden
1076 layers, and 16 attention heads. During training, we keep both the AF-CLAP and LLM frozen during
1077 training. The audio representation transformation layers are fully optimized. We apply LoRA only to
1078 the gated cross-attention layers with a rank of 16.1079 **DiT architecture** We conduct experiments at 24KHz audio sample rate. The waveform latent
representation operates at 50Hz and consists of 128 channels. Our DiT model has 24 blocks, each

1080 with a channel of 1024, 16 attention heads, and an FFN (Feed-Forward Network) dimension of 4096,
1081 resulting in total of 597M parameters. During training, we begin with a 50K linear warm-up phase
1082 and the result is training with a cosine learning rate decay. In the inference, we use 100 diffusion
1083 steps and a guidance scale of 4, with guidance rescale of 0.8.
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

1134

C MORE RESULTS

1135
1136 In this section, we provide more qualitative results on the ALM-inferred editing steps and the
1137 LDM edited audio results.
11381139

C.1 RESULTS ON ALM-INFERENCE ATOMIC EDITING STEPS

1140
1141 We show multiple examples on ALM-inferred atomic editing steps. In each example, the "audio
1142 labels" are the ground truth audio labels for reference and they are not provided to the ALM module.
1143 Only the corresponding audios are provided to the ALM. We then input the declarative instruction to
1144 the ALM and it output the atomic editing steps.
11451146 Example 1: Make this sound like a quiet afternoon in a garden
11471148 Audio labels: clock tick, bird chirp, wind
11491150 Declarative instructions: Make this sound like a quiet afternoon in a garden
11511152 Atomic editing steps:
1153

- Remove the sound of clock tick
- Turn up the the sound of bird tweet by 3dB
- Add the sound of gentle breeze at front by 2dB

1154 Example 2: Make this sound like a busy city street
11551156 Audio labels: engine rev, church bell ring
11571158 Declarative instructions: Make this sound like a busy city street
11591160 Atomic editing steps:
1161

- Remove the sound of bell ring
- Turn down the sound of engine rev by 2dB
- Add the sound of distant siren at left by 2dB
- Add the sound of traffic noise at front by 3dB

1162 Example 3: Make this sound like a bustling park on a sunny day.
11631164 Audio labels: children scream, insect buzz, bird call, chainsaw run
11651166 Declarative instructions: Make this sound like a bustling park on a sunny day.
11671168 Atomic editing steps:
1169

- Remove the sound of chainsaw run
- Add the sound of laughter at left by 3dB
- Turn down children scream by 2dB
- Change the sound of bird call to front

1170 Example 4: Make this sound like a serene beach
11711172 Audio labels: water waves, whistling, music playing
11731174 Declarative instructions: Make this sound like a serene beach
11751176 Atomic editing steps:
1177

- Remove the sound of whistling
- Turn up the sound of wave crash by 4dB
- Add the sound of seagulls calling at front by 3dB

1178 Example 5: Make this sound like a busy office
11791180 Audio labels: typewriter tap, power tool drill
11811182 Declarative instructions: Make this sound like a busy office
11831184 Atomic editing steps:
1185

- Remove the sound of drilling
- Turn up the sound of typewriter type by 2dB

1188 - Add the sound of phone ringing at right by 3dB
 1189

1190 Example 6: Make this sound like a military base
 1191

1192 Audio labels: fire truck siren, gunshot, vehicle move
 1193 Declarative instructions: Make this sound like a military base
 1194 Atomic editing steps:
 1195 - Remove the sound of fire truck siren
 1196 - Turn up the sound of gunshot by 3dB
 1197 - Add the sound of helicopter blades at right by 2dB
 1198

1199 Example 7: Make this sound like a calm night in the countryside
 1200

1201 Audio labels: snoring, machine run, wind blow
 1202 Declarative instructions: Make this sound like a calm night in the countryside
 1203 Atomic editing steps:
 1204 - Remove the sound of machine run
 1205 - Turn down the sound of wind blow by 2dB
 1206 - Add the sound of owl hooting at right by 2dB
 1207

1208 Example 8: Make this sound like a bustling urban street
 1209

1210 Audio labels: horn honk, cat, siren blare, male laugh
 1211 Declarative instructions: Make this sound like a bustling urban street
 1212 Atomic editing steps:
 1213 - Remove the sound of cat
 1214 - Turn down the sound of siren blare by 2dB
 1215 - Add the sound of people chatter at front by 3dB
 1216 - Add the sound of traffic noise at left by 2dB
 1217

1218 **C.2 FAILURE CASE ON ALM-INFERENCED ATOMIC EDITING STEPS.**

1219 To better illustrate how the ALM behaves under abstract or conflicting prompts such as “make it
 1220 sound eerie yet cheerful,” we conducted a test using this instruction across different audio inputs:
 1221

1222 Example 1: Make this sound like eerie yet cheerful
 1223

1224 Audio labels: child cries, wind blows
 1225 Declarative instructions: Make this sound like eerie yet cheerful
 1226 Atomic editing steps:
 1227 - Add the sound of laughing at right by 3dB
 1228 - Add the sound of person whispers at left by 2dB
 1229 - Turn up child cries by 2dB
 1230

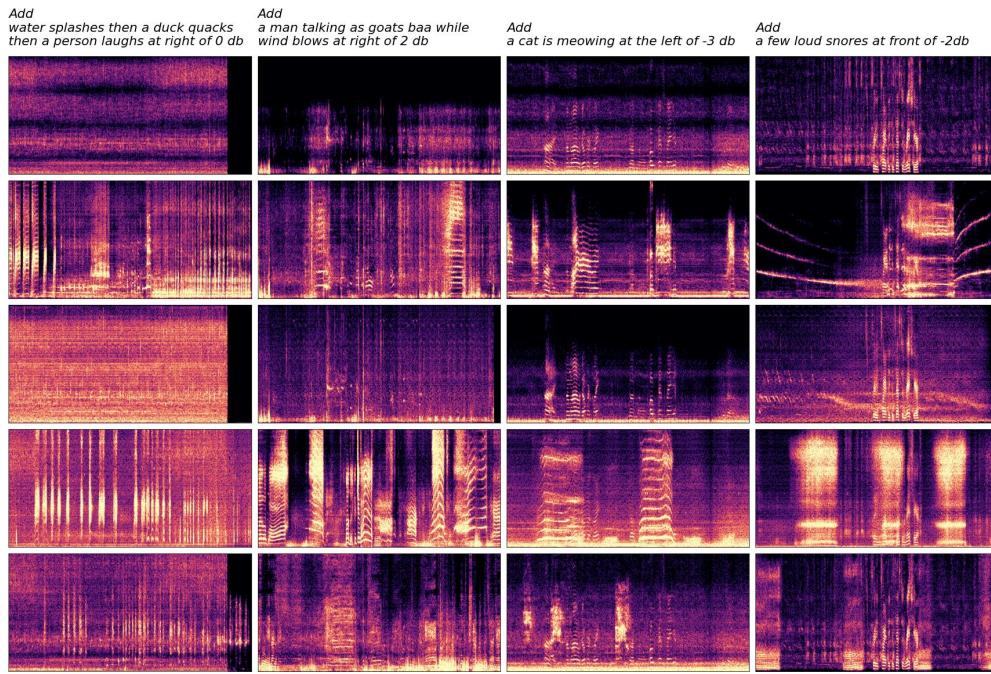
1231 Example 2: Make this sound like eerie yet cheerful
 1232

1233 Audio labels: female speech, bell ringing
 1234 Declarative instructions: Make this sound like eerie yet cheerful
 1235 Atomic editing steps:
 1236 - Turn up the sound of female speech by 2dB
 1237 - Add the sound of owl call at front by 3dB
 1238

1239 Overall, these examples show that the ALM can reliably handle atmosphere-related cues (e.g., “eerie”,
 1240 “cheerful”), but it is still a challenging task when emotional or stylistic attributes conflict or lack clear
 1241 instruction. This reflects an inherent limitation of mapping abstract emotions to concrete operations.
 Extending the ALM to better model emotional and stylistic intent is a natural direction for future
 work.

1242
1243

C.3 RESULTS ON ATOMIC EDITING STEPS

1244
1245

1265

Figure 9: Examples on *add* operation. The top row is the original audio and the rest rows are the edited results. Only SmartDJ can keep the original audio clips while adding new audio content.

1266

We provide more atomic editing results on *add* in Fig 9. As shown by the comparison with the original audio and the edited one, while the baseline method tends to replace the original clips and completely generate a new audio clip, SmartDJ can successfully keep the original contents and *add* new sound events into it.

1273

We show more editing examples on *remove* and *extract* in Fig 10 and 12. Compared with baseline method, SmartDJ can effectively either remove unwanted or extract wanted audio contents. The edited results show good alignment with the ground truth edited audios.

1276

Figure 11 presents additional qualitative results for the *change sound direction* task. The y axis in each figure is the sound direction heatmap. SmartDJ consistently relocates the source to the requested spatial direction, and its outputs align well with the ground truth edited audios. By contrast, Audit can not alter spatial effect, showing the limitations of using spectrogram audio encoder for direction-aware editing.

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

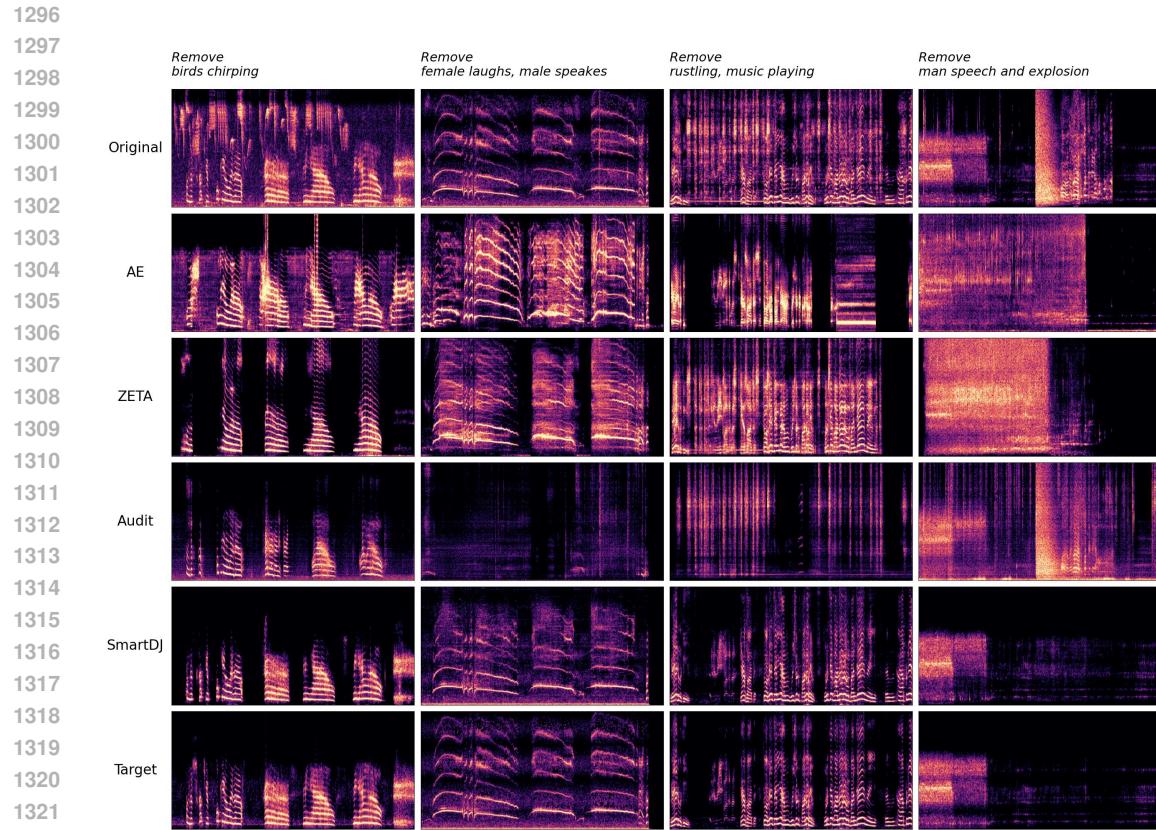


Figure 10: Examples on *remove* operation. The top row is the original audio, and the bottom row is the target audio. Only SmartDJ can completely remove unwanted audio parts and keep the remaining part unchanged.

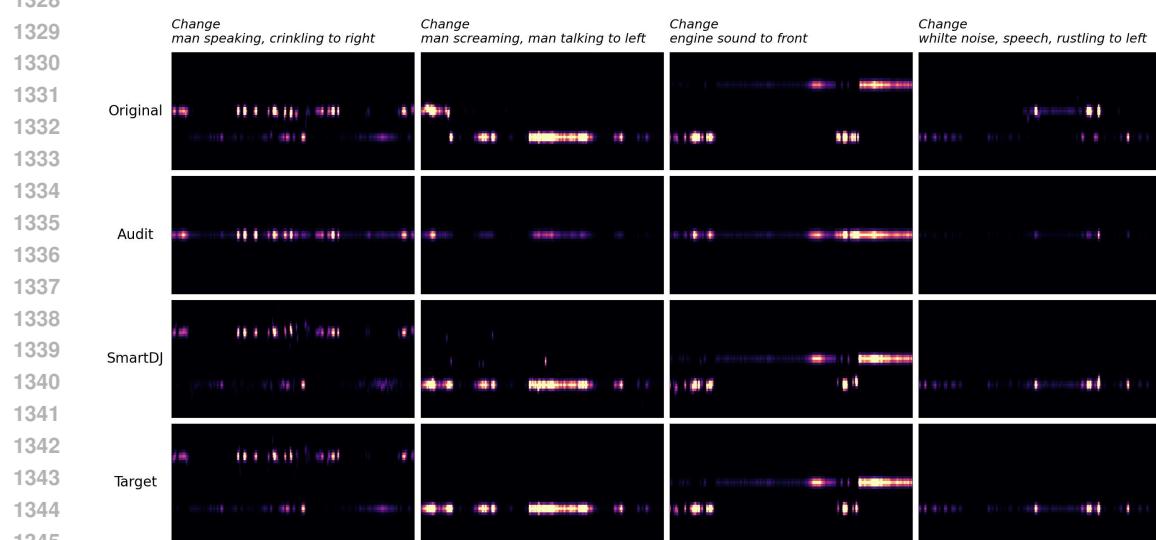
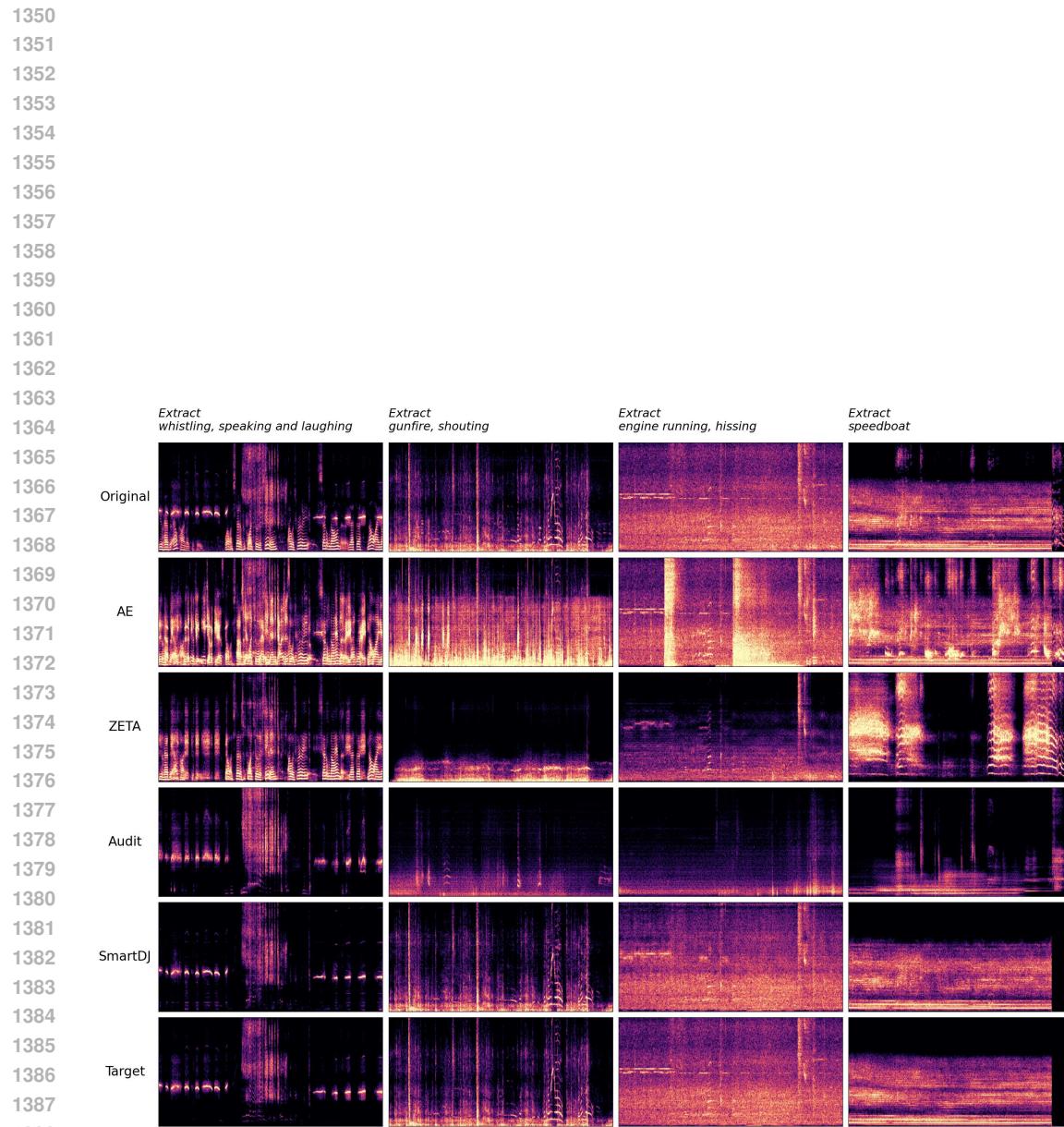


Figure 11: Examples on *change* operation. The top row is the original audio, and the bottom row is the target audio. Only SmartDJ can perfectly edit the sound directions that match closely with the target.

1346
1347
1348
1349



1389 Figure 12: Examples on *extract* operation. The top row is the original audio and the bottom row is
 1390 the target audio. Only SmartDJ can extract wanted audios that are clean and of high quality.

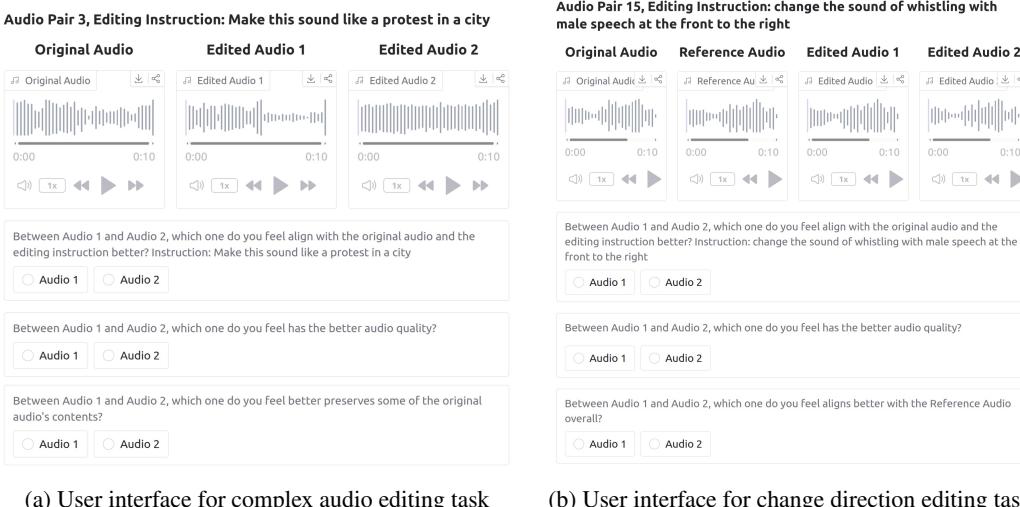
1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1399
 1400
 1401
 1402
 1403

C.4 HUMAN SUBJECTIVE STUDIES DETAILS

Task & Metric	Baseline	Preference (SmartDJ)	95% Confidence Interval	p-value
Declarative Edit Quality	ZETA	80.00%	[68.9% – 91.1%]	< 0.001
	AE	95.50%	[90.6% – 100.0%]	< 0.001
	Audit	90.40%	[83.7% – 97.2%]	< 0.001
Declarative Edit Alignment	ZETA	87.00%	[80.4% – 93.6%]	< 0.001
	AE	91.00%	[86.2% – 95.9%]	< 0.001
	Audit	93.20%	[89.1% – 97.2%]	< 0.001
Single Step Quality	ZETA	77.40%	[66.1% – 88.6%]	< 0.001
	AE	86.30%	[76.8% – 95.7%]	< 0.001
	Audit	95.30%	[90.9% – 99.8%]	< 0.001
Single Step Alignment	ZETA	88.90%	[83.4% – 94.4%]	< 0.001
	AE	84.90%	[78.7% – 91.2%]	< 0.001
	Audit	96.80%	[94.2% – 99.3%]	< 0.001

Table 5: User Preference Evaluation Results.

We provide more details on the subjective user study. We provide users with data pairs consisting of the original audio, the editing instruction, the SmartDJ edited result, and the edited results from a random competing baseline. In the case of single-step audio editing tasks *remove*, *extract*, *turn up/down*, *change sound direction* where there exists a ground-truth editing solution, we also provide it as the reference audio as shown in Fig. 13b. We conduct evaluations with 19 participants and 20 (10 complex audio editing, 10 single-step editing) data pairs per participant.



(a) User interface for complex audio editing task

(b) User interface for change direction editing task

Figure 13: The audio pairs, questions, and user interfaces for different audio editing tasks

In the 10 complex audio editing question pairs, we ask the user to select between the SmartDJ edited audio and the edited results from a random competing baseline (randomly sampled from AudioEditor, ZETA and Audit) according to the following three questions:

Question list for complex audio editing user study

- Between Audio 1 and Audio 2, which one do you feel aligns with the original audio and the editing instruction better?

1458
1459
1460
1461

- Between Audio 1 and Audio 2, which one do you feel has the better audio quality?
- Between Audio 1 and Audio 2, which one do you feel better preserves some of the original audio's contents?

1462 In the single-step editing for *add* operation (3 pairs in total), we compare SmartDJ with a randomly
 1463 sampled method from three baselines (AudioEditor, ZETA and Audit). We ask the user to answer
 1464 four questions, with one additional question listed below:

1465

Additional question for *add* operation user study

1466
1467
1468

- Between Audio 1 and Audio 2, which one do you feel the added spatial effect aligns with the text instruction?

1469
1470
1471
1472

For the single-step *remove* and *extract* tasks (4 pairs) we compare SmartDJ with a randomly chosen baseline model. For *turn-up/turn-down* and *change direction* (3 pairs) we benchmark only against AUDIT, since the other baselines cannot perform these operations. Each pair includes a ground-truth reference, and listeners answer the three evaluation questions listed below.

1473
1474
1475

Questions list for the *Remove*, *Extract*, *Turn up/down*, *Change sound direction* operation user study

1476
1477
1478
1479
1480

- Between Audio 1 and Audio 2, which one do you feel aligns with the original audio and the editing instruction better?
- Between Audio 1 and Audio 2, which one do you feel has the better audio quality?
- Between Audio 1 and Audio 2, which one do you feel aligns better with the Reference Audio overall?

1481
1482
1483
1484
1485
1486
1487
1488

Across all tasks, SmartDJ is consistently preferred over all baseline methods by analyzing its win rate against three baselines. For the complex audio editing task, SmartDJ receives at least 80% of user votes over the baselines for audio quality, and at least 87% for alignment with the Declarative editing instruction and original audio. This shows that SmartDJ faithfully performs the requested scene transformation while preserving the key elements of the original audio. In the single-step editing task, our method receives more than 77% of user votes for audio quality, and 84% for alignment with the single-step editing text prompt, spatial description, and original or reference audio (when applicable). These results demonstrate that SmartDJ achieves the highest user preference across both quality and alignment metrics, outperforming all competing methods.

1489
1490
1491
1492
1493
1494
1495
1496

Regarding the statistical significance of the user study results, we calculated it using a two-sided Z-test against a null hypothesis of random chance (50%). A p-value of < 0.001 indicates that there is less than a 0.1% probability that SmartDJ's preference rates are due to random luck. This confirms that the preference for our method is statistically significant. While the p-value confirms significance, the Confidence Interval demonstrates reliability. The 95% CI indicates the range in which the true population preference lies with 95% certainty. Because the lower bound of our confidence intervals is consistently well above 50% (ranging from 66.1% to 94.2%), we can claim with high certainty that SmartDJ is preferred over baselines in the general population.

1497
1498
1499

D USE OF LARGE LANGUAGE MODELS.

1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

Large language models (LLMs) were used to support the writing of this manuscript. In particular, we used LLMs to assist with writing fluency, polishing the presentation of technical content, and rephrasing sections to improve readability. All scientific insights, technical innovations, and claims presented in this paper are solely the work of the authors. The final content was carefully reviewed, validated, and revised by the authors to ensure both correctness and clarity.