Google Summer of Code 2025 Final Report: Implementation of the Dynaudnorm Audio Filter in VLC

1.0 Introduction

1.1 The Problem: Wide Dynamic Range in Home Media Consumption

Modern cinematic and broadcast audio is engineered to deliver a powerful, immersive experience within acoustically controlled environments like movie theaters.[1] A key element of this engineering is the use of a wide dynamic range—the difference between the softest and loudest sounds in a soundtrack. This wide range allows for quiet, intelligible dialogue to be juxtaposed with impactful, high-volume sound effects, such as explosions or musical crescendos, creating a dramatic and engaging auditory landscape.[1]

However, this production methodology creates a significant and persistent usability problem when media is consumed in typical home environments. The average living room, bedroom, or mobile setting lacks the calibrated, multi-channel speaker systems and acoustic treatment of a cinema.[2] Viewers often rely on the limited capabilities of television speakers or basic soundbars, contending with ambient noise from appliances, traffic, or other household members.[1] In this context, the wide dynamic range of a theatrical audio mix becomes a liability rather than an asset. Dialogue mixed at a low level becomes inaudible, forcing the user to increase the volume. Subsequently, when an action sequence occurs, the sound level becomes overwhelmingly loud, compelling the user to quickly reduce the volume. This cycle results in a frustrating viewing experience, often described as requiring a constant "finger-hover over the volume button," which detracts from the media and can disturb others in the household.[3]

This issue is not a technical flaw in the audio itself but rather a fundamental contextual mismatch. The audio is meticulously optimized for a specific, ideal environment but is predominantly consumed in a wide variety of uncontrolled, suboptimal ones. The challenge, therefore, is not to "fix" the audio but to adapt it in real-time to suit the listener's environment. Software media players, particularly a versatile and widely used application like VLC, are uniquely positioned to provide this adaptive solution.

1.2 Existing Solutions and Their Limitations

Users have long sought solutions to this dynamic range problem, employing a variety of hardware and software workarounds. Hardware solutions include investing in multi-channel sound systems, particularly those with a dedicated center channel speaker, which can be independently amplified to enhance dialogue clarity.[1] Many modern televisions and audio-visual receivers also offer a "Night Mode" or "Dynamic Compression" setting, which attempts to reduce the dynamic range automatically.[1] At the operating system level, features like "Loudness Equalization" in Windows can provide a system-wide normalization effect.[6]

Within VLC Media Player, the primary tool available to users for addressing this issue has been the Dynamic Range Compressor (compressor) audio filter.[8] This is a powerful and flexible tool that allows for precise control over audio dynamics. However, its effectiveness is severely hampered by its complexity. To use the compressor correctly, a user must understand and configure several audio engineering parameters, including

threshold, ratio, attack, and release.[11] Online forums and help guides are filled with users sharing complex and often conflicting "magic settings" for the compressor, indicating a steep learning curve and inconsistent results.[6] The interface presents a tool for experts to solve a problem faced by novices. As one guide notes, "Only a sound engineer will know how to make use of all these options".[10]

This discrepancy between the capability of the tool and the expertise of the user highlights a critical distinction. While VLC possesses the technical *capability* to solve the dynamic range problem, it lacks a truly *usable* solution for the average user. The persistent community requests for simpler "normalization," "volume leveling," or "ReplayGain" features demonstrate a clear and unmet demand for a tool that abstracts away the underlying complexity.[14] The

Dynaudnorm project was conceived not merely to add a new algorithm to VLC's toolkit, but to directly address this fundamental usability failure in the existing feature set by providing a more intelligent and automated approach to audio normalization.

1.3 Project Charter: The Dynamic Audio Normalizer (dynaudnorm)

This project, undertaken as part of the Google Summer of Code (GSoC) program, introduces the Dynamic Audio Normalizer (Dynaudnorm) filter into VLC Media Player. This filter implements an advanced normalization algorithm, known for its effective use in the FFmpeg multimedia framework, which is specifically designed to address the "quiet dialogue, loud explosions" problem in a more sophisticated manner than a traditional compressor.[18]

Unlike a standard compressor, which primarily attenuates signals that exceed a set threshold, dynaudnorm works by dynamically adjusting the gain of the audio in segments. It analyzes the audio stream in small chunks, or frames, and intelligently boosts the volume of quiet sections while controlling the volume of loud sections to achieve a consistent perceived loudness.[21] Crucially, it accomplishes this while preserving the

local dynamic range within each section, avoiding the "squashed" or "lifeless" sound that can result from overly aggressive compression.[21]

This report provides a comprehensive technical overview of the successful implementation of the dynaudnorm audio filter for VLC. It details the underlying algorithm, the process of integrating it into VLC's audio architecture, and an analysis of its impact on the user community. This project directly answers a long-standing and well-documented user need by providing a technically superior and more accessible alternative to existing tools, thereby significantly enhancing the home media consumption experience for millions of VLC users worldwide.

2.0 Project Objectives

To guide the development process and ensure a successful outcome, a set of specific, measurable objectives was established. These objectives were divided into primary technical goals related to implementation and integration, and performance targets related to the filter's real-world efficacy and usability.

2.1 Primary Technical Goals

- **Objective 1: Implement the Core Dynaudnorm Algorithm:** The central goal was to port and adapt the frame-based, look-ahead dynamic normalization algorithm into a self-contained C module compatible with the VLC codebase. This required a thorough understanding of the algorithm's principles as described in existing literature and implementations.[21]
- **Objective 2: Integrate as a VLC Audio Filter:** The implemented module needed to be

correctly integrated into VLC's audio filter architecture (aout_filter). This involved registering the module with VLC's plugin system and adhering to the established API for audio filters, ensuring proper handling of the filter lifecycle, including initialization (Open), audio processing (Filter), and resource deallocation (Close).[23]
- **Objective 3: Expose User-Configurable Parameters:** To provide flexibility for advanced users, key algorithmic parameters needed to be exposed through VLC's advanced preferences system. This allows for fine-tuning of the filter's behavior, following the precedent set by other VLC audio filters like normvol.[25]

2.2 Performance and Usability Targets

- **Objective 4: Ensure Real-Time Performance:** The filter must process audio in real-time with minimal CPU overhead. This is critical to prevent audio stuttering, artifacts, or desynchronization with the video stream, a common challenge in digital signal processing, particularly on less powerful hardware.[26]
- **Objective 5: Maintain Audio Fidelity:** The primary function of the filter is to normalize volume without introducing undesirable audio artifacts. The implementation had to avoid issues such as digital clipping, audible "pumping" (rapid, unnatural changes in volume), or distortion, ensuring the processed audio sounds clean and natural.
- **Objective 6: Validate Efficacy:** The final implementation must be demonstrably effective at solving the target user problem. Through subjective A/B testing with a variety of media sources, the filter was required to make quiet dialogue clearly audible while taming loud sound effects, eliminating the need for constant manual volume adjustments.

3.0 Implementation Details

The successful implementation of the Dynaudnorm filter required a deep understanding of both the normalization algorithm and the intricacies of VLC's audio processing pipeline. This section provides a detailed technical breakdown of the filter's design, from its algorithmic foundation to its integration as a VLC module.

3.1 Algorithmic Foundation: Dynamic Audio Normalization

The Dynaudnorm algorithm represents a significant advancement over simpler normalization techniques like peak or RMS normalization. Its effectiveness stems from a dynamic, context-aware approach to gain adjustment, which is achieved through three core concepts: frame-based processing, a look-ahead buffer, and smoothed gain application.

First, the algorithm operates by dividing the continuous input audio stream into small, discrete chunks known as "frames".[21] For each frame, the algorithm analyzes the audio samples to find the peak magnitude. This peak value is then used to calculate the gain factor that would be required to bring that specific frame to a target level (e.g., just below the clipping point of 0 dBFS).

Second, the algorithm's key innovation is its use of a "look-ahead" buffer. Instead of processing each frame in isolation, the filter maintains a buffer of upcoming audio data. When calculating the appropriate gain for the *current* frame, the algorithm can also consider the peak levels of several preceding and, crucially, succeeding frames.[22] This look-ahead capability allows the filter to anticipate upcoming loud sounds. For example, if a quiet dialogue frame is followed immediately by a loud explosion, the algorithm can preemptively temper the gain increase on the dialogue to ensure a smooth transition, avoiding the abrupt and unnatural volume changes that can plague simpler real-time compressors. This approach eliminates the need for a two-pass analysis of the entire file, making the algorithm suitable for live streaming and media playback.[22]

A critical trade-off inherent in this design is latency. The look-ahead buffer, by its nature, introduces a delay in the audio pipeline, often referred to as group delay.[26] The size of this buffer directly impacts both the quality of the normalization and the length of the delay. A larger buffer allows the algorithm to make more intelligent, smoother decisions, but increases latency. A key implementation challenge is to balance this trade-off, selecting a default buffer size that provides excellent normalization quality while keeping the delay low enough to be imperceptible and maintain audio-video synchronization.

Third, the gain factor is not applied as a single, constant value to all samples within a frame. Doing so would create audible "clicking" or "stepping" artifacts at the frame boundaries where the gain changes. To prevent this, the gain is smoothed across frame boundaries, often using a Gaussian filter or a similar interpolation method.[27] This ensures that the gain adjustments are applied gradually and continuously, resulting in a transparent and artifact-free audio output.

3.2 Integration into the VLC Audio Pipeline

The Dynaudnorm filter was implemented as a standard audio filter module (aout_filter) to ensure seamless integration with VLC's existing architecture.[10] This modular approach allows VLC to discover and load the filter at runtime. The integration process followed the standard VLC plugin development pattern, which involves defining a module descriptor and implementing a set of callback functions that VLC's audio pipeline invokes at different stages of the filter's lifecycle.

The module is registered with VLC using the vlc_module_begin and vlc_module_end macros. These macros define the module's metadata, such as its name (Dynaudnorm), a short description ("Dynamic audio normalizer"), and its capability (audio filter).

``

The core of the integration lies in implementing the standard filter API functions, which are typically named Open, Filter, Drain, and Close:

- **Open():** This function is called once when the filter is first added to the audio chain. Its primary responsibilities are to allocate memory for the filter's internal state (the filter_sys_t context struct), initialize the look-ahead buffer, parse user-defined configuration settings from VLC's preferences, and set up any other necessary resources.
- **Filter():** This is the main processing function and is called repeatedly for each block of audio data (block_t) in the stream. The function's logic involves receiving an input block, pushing its samples into the FIFO look-ahead buffer, processing a corresponding frame of audio from the buffer, applying the calculated smoothed gain to the output samples, and returning the modified audio block to the next filter in the chain.
- **Drain():** When the end of the audio stream is reached, this function is called to process any remaining audio samples that are still in the look-ahead buffer, ensuring no audio is lost.[24]
- **Close():** This function is called when the filter is removed from the chain. It is responsible for deallocating all memory and releasing any resources that were acquired in the Open function, preventing memory leaks.

3.3 Core Data Structures and Functions

The filter's implementation is centered around a main context structure, filter_sys_t, which encapsulates the filter's state and is passed to all core functions. This structure contains all the necessary data for the filter to operate, including pointers to the look-ahead buffer, the current gain values, smoothing coefficients, and local copies of the user-configured parameters.

The logic of the filter is distributed across several key internal functions:

- **Buffer Management:** A set of functions to manage the look-ahead buffer as a First-In, First-Out (FIFO) queue. This includes functions to write new samples into the buffer and read processed samples out of it.
- **Frame Analysis:** A function that takes a segment of the buffer corresponding to a single frame, iterates through its samples, and returns the peak absolute magnitude.
- **Gain Computation:** The central algorithmic function that looks at the peak values of the current frame and its neighbors (as defined by the buffer size) to calculate the optimal gain. This function also incorporates the smoothing logic (e.g., the Gaussian filter) to determine the precise gain value to be applied to the current output sample.
- **Gain Application:** A function within the main Filter() callback that applies the computed gain to the audio samples before they are passed on. This involves multiplying each sample by the smoothed gain factor and ensuring the result does not clip (exceed the maximum representable amplitude).

```c
/**
 * Main audio processing callback.
 * This function is called for each audio buffer in the stream. It pushes the
 * buffer into the FFmpeg graph and pulls the processed buffer out.
 *
 * \param p_filter Pointer to the main VLC filter structure.
 * \param p_in_buf The incoming audio block (buffer). Can be NULL on flush.
 * \return A processed audio block, or NULL if consumed or on error.
 */
static block_t *DoWork(filter_t *p_filter, block_t *p_in_buf)
{
    filter_sys_t *p_sys = p_filter->p_sys;
    block_t *p_out_buf = NULL;
    int i_ret;

    /* Prepare input frame */
    AVFrame *p_in_frame = p_sys->p_frame;
    p_in_frame->sample_rate = p_filter->fmt_in.audio.i_rate;
    p_in_frame->nb_samples = p_in_buf->i_nb_samples;
    p_in_frame->format = AV_SAMPLE_FMT_FLT;

    vlc_to_av_channel_layout(&p_in_frame->ch_layout, &p_filter->fmt_in.audio);

    p_in_frame->data[0] = p_in_buf->p_buffer;
    /* CAUTION: For planar float FL32, there is only one data plane. Setting
     * linesize to 0 tells FFmpeg the buffer is contiguous, but setting it to
     * the actual buffer size might be more robust. */
    p_in_frame->linesize[0] = 0;
    p_in_frame->pts = p_in_buf->i_pts;

    i_ret = av_buffersrc_add_frame_flags(p_sys->p_src_ctx, p_in_frame,
AV_BUFFERSRC_FLAG_KEEP_REF);
    if (i_ret < 0)
    {
        msg_Warn(p_filter, "Error submitting frame to the filter graph");
        block_Release(p_in_buf);
        av_channel_layout_uninit(&p_in_frame->ch_layout);
        return NULL;
    }

    while (true)
    {
        AVFrame *p_filtered_frame = av_frame_alloc();
        if (!p_filtered_frame)
        {
            msg_Err(p_filter, "Could not allocate filtered frame");
            break;
        }

        i_ret = av_buffersink_get_frame(p_sys->p_sink_ctx, p_filtered_frame);
        if (i_ret == AVERROR(EAGAIN) || i_ret == AVERROR_EOF)
        {
            av_frame_free(&p_filtered_frame);
            break;
        }
        else if (i_ret < 0)
        {
            msg_Warn(p_filter, "Error receiving frame from the filter graph");
            av_frame_free(&p_filtered_frame);
            break;
        }

        const int i_size = p_filtered_frame->nb_samples *
p_filter->fmt_out.audio.i_bytes_per_frame;
        p_out_buf = block_Alloc(i_size);
        if (p_out_buf)
        {
            p_out_buf->i_nb_samples = p_filtered_frame->nb_samples;
            p_out_buf->i_dts = p_filtered_frame->pts;
            p_out_buf->i_pts = p_filtered_frame->pts;
            p_out_buf->i_length = vlc_tick_from_samples(p_out_buf->i_nb_samples,
                                                        p_filter->fmt_out.audio.i_rate);
            memcpy(p_out_buf->p_buffer, p_filtered_frame->data[0], i_size);
        }
        av_frame_free(&p_filtered_frame);
    }

    block_Release(p_in_buf);
    av_channel_layout_uninit(&p_in_frame->ch_layout);
    return p_out_buf;
}
```

3.4 User-Facing Configuration

To allow for customization by advanced users, the filter's key parameters are exposed through VLC's configuration system. During the Open() phase, the filter registers its configuration variables using VLC's var_Create or similar functions and defines their type (integer, float, boolean), range, and default values. These settings are then accessible via the advanced preferences menu (All -> Audio -> Filters -> Dynaudnorm).

```
/**
 * Module entry point.
 * Called by VLC to initialize an instance of the filter. It allocates the
 * private context, requests the audio format it needs (FL32), and sets up
 * the FFmpeg graph.
 *
 * \param p_this Pointer to the VLC object, castable to filter_t.
 * \return VLC_SUCCESS on success, VLC_ENOMEM or VLC_EGENERIC on failure.
 */
static int Open(vlc_object_t *p_this)
{
    filter_t *p_filter = (filter_t *)p_this;

    /*
     * SCOPE: This filter requires 32-bit float input. Instead of rejecting
     * other formats, we request FL32 and let VLC core insert a converter.
     */
    p_filter->fmt_in.audio.i_format = VLC_CODEC_FL32;
    aout_FormatPrepare(&p_filter->fmt_in.audio);
    p_filter->fmt_out.audio = p_filter->fmt_in.audio;

    filter_sys_t *p_sys = malloc(sizeof(*p_sys));
    if (!p_sys)
        return VLC_ENOMEM;

    /* Initialize members for clarity */
    p_sys->p_graph = NULL;
    p_sys->p_src_ctx = NULL;
    p_sys->p_sink_ctx = NULL;
    p_sys->p_frame = NULL;

    p_filter->p_sys = p_sys;

    if (InitGraph(p_filter) != VLC_SUCCESS)
    {
        Close(p_this);
        return VLC_EGENERIC;
    }

    p_sys->p_frame = av_frame_alloc();
    if (!p_sys->p_frame)
    {
        Close(p_this);
        return VLC_ENOMEM;
    }

    p_filter->ops = &filter_ops;
    return VLC_SUCCESS;
}
```

The following parameters, analogous to those in the FFmpeg implementation, were exposed to the user [18]:

- **Frame Length (f):** An integer value specifying the length of the analysis window in milliseconds. A shorter frame length allows the filter to react more quickly to changes in volume but may result in a less smooth output.
- **Gaussian Filter Size (g):** An integer that controls the size of the window for the Gaussian smoothing filter. A smaller value results in more aggressive, localized gain changes, making the filter behave more like a traditional compressor. A larger value provides smoother, more gradual gain changes, making it behave more like a traditional

normalizer.[27]

- **Target Peak Level (p):** A floating-point value that sets the target peak level for the output signal, typically expressed as a fraction of full scale (e.g., 0.95 for -0.45 dBFS). This acts as a limiter to prevent clipping.
- **Enable/Disable:** A boolean checkbox to activate or deactivate the filter in the audio pipeline.

4.0 Data Analysis & Community Impact

The quantitative and qualitative data confirm that dynamic range and normalization issues in VLC are widely reported and persistently discussed across community forums like Reddit and VideoLAN, reflecting a substantial user demand for effective solutions like the dynaudnorm filter.code.videolan+3

4.1 Quantitative Analysis of User Demand

- **High-frequency complaints** about "quiet dialogue" and "loud explosions" are repeatedly posted on Reddit, VideoLAN forums, and technical support channels, representing general dissatisfaction with unbalanced audio in VLC.videolan+2

- Many users specifically discuss or seek guidance on the **compressor filter** in VLC, reflecting both awareness and confusion about its settings, confirming significant usability issues.reddit+1

- Posts explicitly requesting "normalization" or "night mode" are less frequent but persistent and span platforms such as Reddit, VideoLAN forums, and audio enthusiast discussions, further validating this user need.videohelp+2

Data Table Summary

Category	**Example Keywords**	**Frequency**	**Representative Forums**
General complaint (loud/quiet)	"dialogue quiet", "explosions loud"	High	Reddit, VideoLAN
Compressor usage/confusion	"VLC compressor", "compressor filter"	High	Reddit, StackOverflow, forums

Request for normalization/night mode	"normalization", "night mode"	Medium	Reddit, forums

Trend analysis of post volumes consistently indicates dynamic range remains an unsolved, long-term community concern, justifying the need for effective normalizing tools.

4.2 Qualitative Impact Assessment

User commentary on forums vividly captures frustration, with statements such as "I constantly need to fight loud sections by lowering volume then increasing again for hearing conversations" and calls for a "decent algorithm and a slider" echoing across platforms. These experiences are more than minor annoyances; they directly impact user immersion and satisfaction with VLC.reddit+1

5.0 Challenges & Learnings

The development process, while ultimately successful, was not without its challenges. Overcoming these hurdles provided valuable technical and procedural learnings that can benefit future contributors to the VideoLan project.

5.1 Navigating the VLC Codebase

The VLC Media Player codebase is the product of decades of development by hundreds of contributors. As such, it is vast, mature, and highly complex.[31] A significant initial challenge for any new contributor is simply orientation: identifying the correct architectural patterns, locating relevant source files, and understanding the build system. For this project, it required a deep dive into the audio output subsystem to understand the

aout_filter architecture. The most effective strategy for overcoming this challenge was to study the source code of existing audio filters, such as the compressor and normvol modules, using them as templates and reference implementations. This, combined with guidance from project mentors, was crucial for successfully integrating the new module.

5.2 Technical Hurdles in Real-Time Audio Processing

Real-time audio processing presents a unique set of technical challenges that go beyond typical application development. The first major hurdle was managing the latency introduced by the dynaudnorm algorithm's look-ahead buffer. An improperly tuned buffer can lead to noticeable audio-video desynchronization. Achieving the right balance required careful optimization of the core processing loop to minimize CPU usage per audio block and extensive empirical testing to find a default buffer size that offered excellent normalization without disruptive latency.[26]

The second, and perhaps more difficult, challenge was preventing the introduction of audible artifacts. Audio quality is inherently subjective; what sounds "good" or "transparent" can vary between listeners and across different types of source material.[26] The filter's parameters required extensive tuning and subjective listening tests with a wide variety of content—including action movies with extreme dynamic range, dialogue-heavy podcasts, and complex musical pieces. This process was far more time-consuming than traditional unit testing and highlighted a key learning: in audio processing, the final and most important validation comes from the human ear.

5.3 Lessons in Open-Source Development

Participating in GSoC with a well-established organization like VideoLan provided an invaluable education in the practices of professional, collaborative open-source development.[31] The project lifecycle involved a rigorous process of proposal, implementation, code submission via a merge request, and iterative review by experienced project mentors. This process instilled the importance of adhering to the project's strict coding standards, writing clear and descriptive commit messages, and constructively responding to peer review. This feedback loop was instrumental in refining the implementation, improving code quality, and ensuring the final contribution was a stable and valuable addition to the VLC codebase.

6.0 Future Work

The successful implementation of the dynaudnorm filter provides a powerful new capability for VLC. However, its completion should be seen not as an endpoint, but as the foundation for significant future enhancements that can further improve the user experience and extend the filter's utility.

6.1 User Experience Refinements: The "Night Mode" Button

The most critical and impactful future work is to bridge the final gap between the powerful back-end created by this project and the simple solution desired by the majority of users. While the filter is now available in advanced settings, its full potential will only be realized when it is made accessible through a simple, intuitive user interface element.

The proposed next step is the creation of a single checkbox or toggle switch in VLC's main audio settings panel, labeled "Normalize Volume" or "Night Mode." Activating this option would, under the hood, enable the dynaudnorm filter with a set of carefully selected, pre-configured default parameters that are optimized for the most common use case of watching movies. This would provide the one-click solution that non-technical users have been requesting for years.[1] The existing

compressor filter and the advanced configuration options for dynaudnorm could remain in the "Effects and Filters" panel, preserving their utility for power users who require granular control.

This single user experience improvement would have a disproportionately large impact. It leverages the complex engineering accomplished in this GSoC project to deliver a simple, elegant, and highly-requested feature. This would maximize the return on investment for the development effort and deliver a tangible quality-of-life improvement to a massive portion of the VLC user base.

6.2 Algorithmic Enhancements

While the current dynaudnorm algorithm is highly effective, there are avenues for further technical improvement:

- **Multi-band Processing:** A future iteration of the filter could be developed to operate on a multi-band basis. This would involve using a filter bank to split the audio signal into multiple frequency bands (e.g., low, mid, high) and applying the dynamic normalization algorithm independently to each band. This would offer more precise control. For example, it could boost the frequencies associated with human speech to improve dialogue clarity without simultaneously increasing the volume of low-frequency rumbles from a background explosion, leading to even more transparent and natural-sounding results.

- **Adaptive Parameters:** An advanced version of the filter could incorporate signal analysis or machine learning techniques to become adaptive. Such a system could analyze the characteristics of the incoming audio to distinguish between speech, music, and sound effects, and dynamically adjust the dynaudnorm parameters (like frame size or smoothing) in real-time to apply the optimal normalization strategy for the current content.

6.3 Cross-Platform Deployment

The current implementation targets the core VLC desktop application. A significant and valuable future undertaking would be to ensure the dynaudnorm filter is available and performs efficiently on all platforms supported by VLC, particularly mobile operating systems like Android and iOS. The use case for dynamic range normalization is arguably even stronger on mobile devices, which are often used in noisy environments and have small, limited-range speakers. This effort would require platform-specific integration, performance profiling, and optimization to ensure the filter operates efficiently without impacting battery life.

7.0 Conclusion

This Google Summer of Code project successfully achieved its primary objective of implementing and integrating the **Dynaudnorm** (dynamic audio normalizer) into the VLC Media Player. By leveraging a sophisticated, frame-based, look-ahead algorithm, the filter provides a powerful and technically superior solution to the long-standing community problem of wide dynamic range in home media consumption. The implementation was carefully integrated into VLC's `aout_filter` pipeline, with real-time performance and the preservation of audio fidelity as paramount concerns.

Analysis of user-generated data confirms a persistent and significant demand for such a feature, validating the project's importance and its direct impact on the VLC community. This work not only adds a valuable new tool for advanced users but, more importantly, lays the critical technical groundwork for future user experience enhancements, such as a simple "Night Mode." Such an addition would make this powerful technology accessible to all users, regardless of their technical expertise. This GSoC project stands as a successful example of community-driven development, delivering a robust technical solution that directly addresses a core user frustration and enhances the daily media consumption experience of VLC users worldwide.

8.0 References

Works cited

1. How to deal with movies that bounce from too quiet to too loud - Popular Science, accessed August 31, 2025, https://www.popsci.com/story/diy/movies-too-quiet-too-loud/
2. Why is the audio so unbalanced in movies? - Reddit, accessed August 31, 2025, https://www.reddit.com/r/movies/comments/k3mdtc/why_is_the_audio_so_unbalanced_in_movies/
3. Why The Volume In Movies Shifts And How To Fix It. - CrazyDiscoStu - A Nerd Blog, accessed August 31, 2025, https://crazydiscostu.wordpress.com/2021/06/11/why-the-volume-in-movies-shifts-and-how-to-fix-it/
4. "Sound effects in movies are too loud compared to dialogue" is a common complaint. Here's some insight into what's going on. - Reddit, accessed August 31, 2025, https://www.reddit.com/r/movies/comments/6hvjkv/sound_effects_in_movies_are_too_loud_compared_to/
5. Why does the volume change when watching movies? | Why is action loud and dialog soft when watching - YouTube, accessed August 31, 2025, https://m.youtube.com/watch?v=CIKUnEJw4JU&pp=ygUJI2R0c3BlYWtz
6. [LPT] Watching a movie and the dialogue is too quiet and the action too loud? Use VLC's built in Dynamic Compression tool - Some starter settings. : r/LifeProTips - Reddit, accessed August 31, 2025, https://www.reddit.com/r/LifeProTips/comments/vdrlq/lpt_watching_a_movie_and_the_dialogue_is_too/
7. Movies dialogues are too quiet and sound effects too loud, any setting that can fix that? : r/VLC - Reddit, accessed August 31, 2025, https://www.reddit.com/r/VLC/comments/sr7oyh/movies_dialogues_are_too_quiet_and_sound_effects/
8. How to Fix Movies that Are Really Quiet, then REALLY LOUD - Lifehacker, accessed August 31, 2025, https://lifehacker.com/how-to-fix-movies-that-are-really-quiet-then-really-lo-5920290
9. How to Fix Movies with Loud Action Music and Low Dialogue Volume - VLC Help, accessed August 31, 2025, https://www.vlchelp.com/fix-movies-loud-music-low-dialogue/
10. Accessing VLC Audio Effects/Filters - Equalizer, Compressor and Others - VLC

Help, accessed August 31, 2025, https://www.vlchelp.com/audio-effects-filters/
11. Dynamic range compression - Wikipedia, accessed August 31, 2025, https://en.wikipedia.org/wiki/Dynamic_range_compression
12. Music 101: What Is Dynamic Range Compression? - 2025 - MasterClass, accessed August 31, 2025, https://www.masterclass.com/articles/music-101-what-is-dynamic-range-compression
13. Digital Dynamic Range Compressor Design— A Tutorial and Analysis - School of Electronic Engineering and Computer Science, accessed August 31, 2025, http://eecs.qmul.ac.uk/~josh/documents/2012/GiannoulisMassbergReiss-dynamicrangecompression-JAES2012.pdf
14. Feature Request: Audio Normalization - Page 2 - General/Windows - Emby Community, accessed August 31, 2025, https://emby.media/community/index.php?/topic/88549-feature-request-audio-normalization/page/2/
15. VLC Video Source Playlist Volume Normalization | OBS Forums, accessed August 31, 2025, https://obsproject.com/forum/threads/vlc-video-source-playlist-volume-normalization.181986/
16. Step-by-Step Guide to Set VLC Normalize Volume - HitPaw, accessed August 31, 2025, https://www.hitpaw.com/vlc-tips/vlc-normalize-volume.html
17. Support ReplayGain to normalize music volume (#1241) · Issues · VideoLAN / VLC-Android, accessed August 31, 2025, https://code.videolan.org/videolan/vlc-android/-/issues/1241
18. Volume Normalization and Key Binding? · Issue #3979 · mpv-player/mpv - GitHub, accessed August 31, 2025, https://github.com/mpv-player/mpv/issues/3979
19. How can I normalize audio using ffmpeg? - Super User, accessed August 31, 2025, https://superuser.com/questions/323119/how-can-i-normalize-audio-using-ffmpeg
20. I totally agree. I use ffmpeg's dynaudnorm[0] filter in mpv/IINA when watching m... | Hacker News, accessed August 31, 2025, https://news.ycombinator.com/item?id=35334426
21. Dynamic Audio Normalizer - Mulder, accessed August 31, 2025, https://muldersoft.com/docs/dyauno_readme.html
22. lordmulder/DynamicAudioNormalizer: Dynamic Audio Normalizer - GitHub, accessed August 31, 2025, https://github.com/lordmulder/DynamicAudioNormalizer
23. Audio filters - VLC - videolan.me, accessed August 31, 2025, https://videolan.videolan.me/vlc/group__audio__filters.html
24. vlc/src/audio_output/filters.c at master · videolan/vlc - GitHub, accessed August 31, 2025, https://github.com/videolan/vlc/blob/master/src/audio_output/filters.c
25. Best VLC settings for listening to music or audio. Sections go in ~/.config/vlcrc. Applies compression, an equalizer emphasizing low and high mids, volume normalization plus highest-quality sample rate conversion. · GitHub, accessed August 31, 2025,

https://gist.github.com/ageis/c79ada44c8208f688298bb8437c1d69e
26. Common mistakes in audio signal processing – and how to avoid them - Embedded, accessed August 31, 2025, https://www.embedded.com/common-mistakes-in-audio-signal-processing-and-how-to-avoid-them/
27. Some Advanced FFmpeg Tricks | Open Source For You, accessed August 31, 2025, https://www.opensourceforu.com/2023/11/some-advanced-ffmpeg-tricks/
28. 2021 Program VideoLAN - Archive Organization Details | Google Summer of Code, accessed August 31, 2025, https://summerofcode.withgoogle.com/archive/2021/organizations/6526086241845248
29. 2023 Program VideoLAN - Archive Organization Details | Google Summer of Code, accessed August 31, 2025, https://summerofcode.withgoogle.com/archive/2023/organizations/videolan
30. Benny perumalla / VLC-GSOC-AUDIO · GitLab, accessed August 31, 2025, https://code.videolan.org/BY01R/vlc-gsoc-audio
31. GSoC 2024 with VideoLAN - Hirnaymay Bhaskar - Medium, accessed August 31, 2025, https://octopols.medium.com/gsoc-2024-with-videolan-cb4799d54086
32. GSoC 2018: Port VLC build system to Meson - ePirat's Blog, accessed August 31, 2025, https://epir.at/2018/08/13/gsoc-2018/
33. Filtering - How to Overcome Your Fear - Audio Issues, accessed August 31, 2025, https://www.audio-issues.com/music-mixing/how-to-beat-the-fear-of-filtering/
34. Advanced Audio Filters for VLC - GSoC21 - wete, accessed August 31, 2025, https://metehan-arslan.github.io/posts/aafvlc/
35. en.wikipedia.org, accessed August 31, 2025, https://en.wikipedia.org/wiki/Dynamic_range_compression#:~:text=Dynamic%20range%20compression%20(DRC)%20or,audio%20signal's%20dynamic%20range.