LUMIA: A Handheld Vision-to-Music System for Real-Time, Embodied Composition

Connie Cheng[*1]**, Chung-Ta Huang**[*1]**, and Vealy Lai**[2]

[1]Harvard University, Cambridge, MA 02138,
connie_cheng@gsd.harvard.edu, chungta_huang@gsd.harvard.edu
[2]Massachusetts Institute of Technology, Cambridge, MA 02139, laiv@mit.edu

Abstract

Most digital music tools emphasize precision and control, but often lack support for tactile, improvisational workflows grounded in environmental interaction. Lumia addresses this by enabling users to "compose through looking"—transforming visual scenes into musical phrases using a handheld, camera-based interface and large multimodal models. A vision-language model (GPT-4V) analyzes captured imagery to generate structured prompts, which, combined with user-selected instrumentation, guide a text-to-music pipeline (Stable Audio). This real-time process allows users to frame, capture, and layer audio interactively, producing loopable musical segments through embodied interaction. The system supports a co-creative workflow where human intent and model inference shape the musical outcome. By embedding generative AI within a physical device, Lumia bridges perception and composition, introducing a new modality for creative exploration that merges vision, language, and sound. It repositions generative music not as a task of parameter tuning, but as an improvisational practice driven by contextual, sensory engagement 1.

1 Introduction

Recent advances in generative AI have enabled creative tools across text, image, and audio, yet most remain screen-based and prompt-driven, limiting physical engagement and real-time improvisation in music production. *Lumia* is a handheld, camera-based device for real-time music generation from visual input. It uses GPT-4 Vision to analyze captured scenes and construct structured prompts from objects, context, and inferred mood, which are then passed to Stable Audio to synthesize short loops [8, 16]. Users influence generation by framing images, selecting instruments, and layering clips via a browser or physical controller, while the model introduces interpretive variation. *Lumia* extends the Large Language Object (LLO) concept [6], as introduced in VBox [13], which embedded generative models in materially expressive systems. Whereas VBox enabled tactile navigation of latent audio spaces, *Lumia* shifts to composition, linking visual perception with multimodal generation. It treats the visual world as a source of sonic material, enabling a co-creative workflow where user intent and model inference converge, resulting in an embodied, improvisational interface for AI-driven sampling and composition grounded in environmental context.

[0]Project page & code: https://github.com/KidaGSD/LLOv2

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Creative AI Track.

Figure 1: Lumia Device

2 Related Work

2.1 Multimodal Generative Models

Recent models such as DALL·E, and GPT-4 Vision have shown strong performance in cross-modal tasks, including image captioning, scene understanding, and prompt generation [17]. In the audio domain, models like MusicLM, AudioLDM, and Stability AI's Stable Audio have demonstrated the feasibility of text-to-music generation with varying degrees of control over structure, instrumentation, and genre [1, 8, 15]. While these systems demonstrate strong generative performance, they are typically accessed via prompt-based or batch-processing interfaces, and offer limited support for interactive scenarios such as live composition, exploratory sampling, or iterative refinement. Most lack mechanisms for real-time control over generation parameters, continuous multimodal input [10], or feedback-driven adaptation, which are critical for workflows that depend on responsiveness, improvisation, and embodied engagement.

2.2 Creative AI and Co-Creation

Work in creative AI has increasingly emphasized tools that position the human as an active collaborator rather than a passive consumer. Systems such as Magenta Studio, Jukebox, and Soundify offer musicians new modes of engagement, but often assume technical familiarity or require traditional DAW integration [7, 9]. Lumia differs by embedding creative interaction into a physical object, making it more accessible and intuitive, especially for non-experts. It aligns with ongoing research into co-creative systems, where human input and machine generation occur in tandem, influencing one another in real time[2].

2.3 Tangible and Embodied Interfaces

There is a long-standing tradition in HCI of exploring tangible and embodied interfaces for creative expression. Systems such as Reactable and Bela-based musical hardware [12**?**] have shown that hands-on interaction can enhance improvisation, flow, and expressive control in music-making. These systems often operate in fixed environments and rely on predefined spatial mappings. More recent work, such as Be the Beat[5], extends this paradigm by embedding generative AI into a beatbox-shaped device that responds to dancers' movements, using physical form language to cue interaction and support situated, improvisational performance. Lumia builds on this direction by introducing a mobile, camera-inspired interface for visual exploration and sound generation, further emphasizing physical context and environmental engagement as central components of co-creative AI systems.

2.4 Gap and Positioning

While prior work has addressed text-to-music generation, multimodal synthesis, and tangible instruments, few systems combine these elements into a real-time, embodied workflow grounded in

physical and environmental interaction. Most existing tools remain screen-based or prompt-driven, limiting their use in improvisational or situated creative practices. Lumia fills this gap by framing vision as a form of sampling and embedding generative AI into a portable, physically expressive device. It enables users to compose music through visual exploration, positioning AI as a responsive collaborator in a context-aware, co-creative process(see Figure 2).

3 Methodology

Lumia was designed to enable music creation from visual input while preserving user agency, with large multimodal models forming the core of its visual-to-audio pipeline. To keep interaction intuitive and accessible, it generates fixed-length audio clips that users can blend and sequence into tracks. Inspired by DJ turntables, early prototypes focused on live layering and looping, exploring interaction styles where clips were added sequentially, like a conductor introducing instruments. This led to experiments with single-instrument samples for flexible mixing, while a physical turntable informed timing and blending dynamics. Rapid prototyping and informal testing shaped the camera-inspired form and audio-themed visual language. Iterative development revealed key insights for coherence: image color strongly influenced genre inference, prompting the addition of physical color filters for simple visual genre control; overemphasis on literal objects degraded audio, so prompts were biased toward contextual and atmospheric descriptors, with genre, tempo, and key kept consistent across segments. Technical challenges included harmonizing new samples with prior material—the generation model's lack of temporal awareness made full-track regeneration disruptive. This was addressed with a sequential composition model, appending each new section in time to preserve responsiveness and continuity.

3.1 Physical Form and Interaction

The device resembles a compact camera with five buttons: four for selecting instruments (keys, guitar, bass, percussion) and one for capture and playback. LEDs indicate the current instrument state, and an onboard display shows tempo, genre, section role, and audio levels. A built-in speaker handles real-time loop playback, while a microcontroller manages I/O. A camera module continuously streams video to the frontend; still frames are sent to a vision-language model (VLM) for analysis.

3.2 Hardware Protocol and Microcontroller Firmware

The Arduino firmware is implemented as a finite-state loop. It debounces button inputs, updates the display, and adjusts LEDs and audio level indicators. LCD updates reflect session metadata, while LEDs respond to both software state and user input. The firmware is stateless beyond its display responsibilities, making all playback and audio logic frontend-driven.

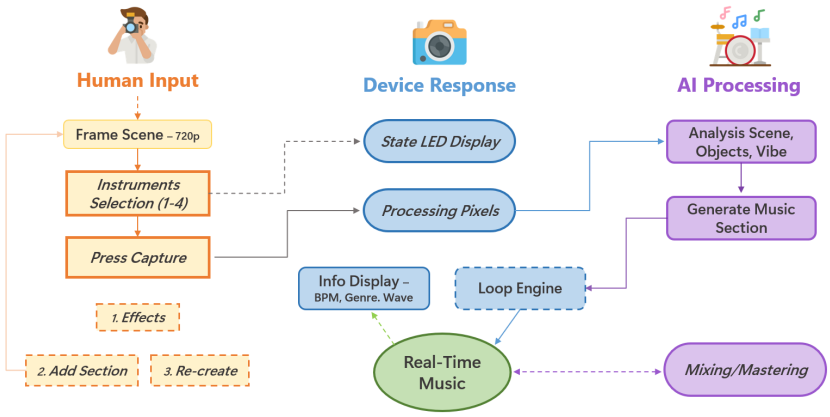

Figure 2: User Flow

4 System Architecture

Lumia's system architecture (see Figure 3) is designed as a modular, decoupled system comprising three primary pillars: (1) a tangible hardware controller for physical interaction, (2) a browser-based frontend application that serves as the central orchestrator, and (3) a suite of cloud-based AI services for content generation and processing. This architecture is designed for instantaneous physical feedback to the user while managing the high-latency, asynchronous nature of generative AI calls in the background, ensuring a natural creative experience. Full details of hardware configuration, API service specifications, latency, and cost are provided in Appendix to support reproducibility.

4.1 Music Generation: Vision-to-Music Pipeline

Image Captioning: When the capture button is pressed, a frame from the live camera feed is sent to GPT-4 Vision for extracting meaningful scene information. The vision model returns a structured JSON caption describing the image, with fields for: 1) overall description of the scene, 2) a list of salient objects, 3) the overall mood(adjectives), 4) section role $\in \{intro, verse, chorus, bridge, outro\}$, 5) a music genre(based on the first section), 6) suggested BPM(beats per minute). This metadata is designed to capture both the content and the atmosphere of the photo, providing a basis for music generation. (If section inference is uncertain, system would default to *verse*)

4.2 Prompt Construction:

The system constructs a music generation prompt by combining structured GPT-4 Vision output with user-defined instrumentation. Users select 1 to 3 instruments (keys/synth, guitar, bass, drums) to define the sonic range; this cap promotes perceptual stream segregation and reduces masking in novice-facing settings [3]. The parsed image caption containing scene description, mood, genre, BPM, and section role (e.g., intro, verse, chorus)—is programmatically merged with these inputs. To scaffold musical structure, section-specific modifiers (e.g., "higher energy, catchy hookc for chorus; "winding down" for outro) and, for non-initial segments, a variation tag (e.g., "motif development", "steady groove") are appended to promote temporal progression and thematic continuity. All elements are concatenated into a single sentence-level prompt optimized for the Stability AI audio generation model. For example:

```
keys, guitar section, purple neon street light sign at night, moody,
lush, ambient chill, steady groove, subtle variation, same sound
palette as previous section
```

This structured prompt encodes intended instrumentation, musical character, temporal role, and stylistic direction to condition the audio model effectively.

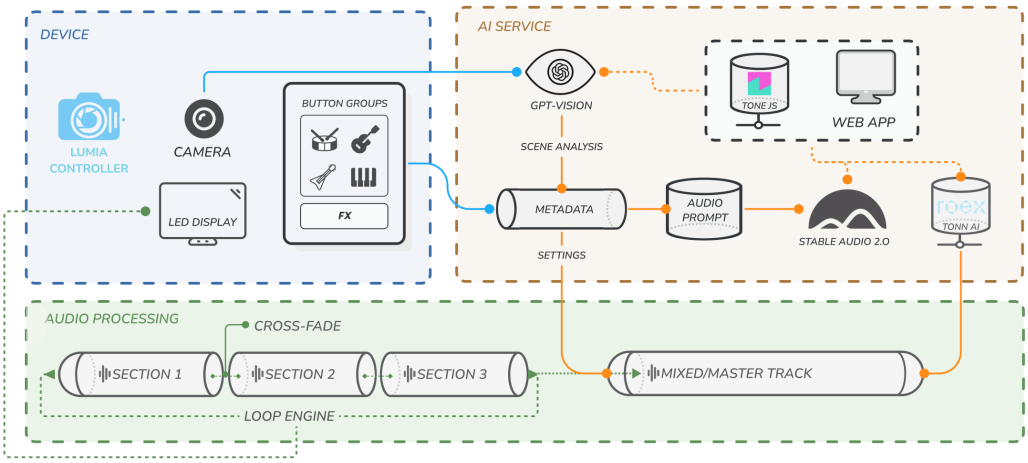

Figure 3: System Diagram

4.3 Audio Generation

The composed prompt is sent to the Stability AI-Stable Audio API(2.0) for music generation. Each section is generated as a 15 seconds audio clip with a target tempo derived from the caption. The generation parameters (stereo output at 44.1kHz) are held constant for all sections to ensure uniform audio quality. The Stable Audio model interprets the text prompt to produce an audio waveform matching the described scene, mood, instruments, and genre. The returned audio WAV file is immediately decoded into an AudioBuffer and added to the playback loop.

4.4 Loop Playback Engine

Timing and scheduling: Once the first section is generated, the engine enters a continuous loop. Let b denote the session tempo (BPM). A beat lasts $T_{\text{beat}} = 60/b$ seconds and a bar is $T_{\text{bar}} = 4T_{\text{beat}}$. We generate fixed-length sections $L_k = m_k T_{\text{bar}}$, $m_k \in \mathbb{N}$. To ensure smooth overlaps we define a tempo-adaptive crossfade window (Eq. 1) and schedule the next section start time (Eq. 2):

$$T_{\text{cf}}(b) = \max\Big(\tfrac{120}{b},\, 0.3\Big)\ \text{s} \qquad (1) \qquad\qquad t_{k+1} = t_k + L_k - T_{\text{cf}} \qquad (2)$$

Crossfade envelopes: To avoid clicks and maintain perceived loudness during the overlap, we default to equal-power(eq) crossfades. For outgoing buffer $x[n]$ and incoming buffer $y[n]$ we set

$$g_{\text{out}}(n) = \cos\Big(\tfrac{\pi n}{2N}\Big), \quad g_{\text{in}}(n) = \sin\Big(\tfrac{\pi n}{2N}\Big), \quad z[n] = g_{\text{out}}(n)x[n] + g_{\text{in}}(n)y[n], \qquad (3)$$

Since $g_{\text{out}}^2(n) + g_{\text{in}}^2(n) = 1$, the instantaneous power remains approximately constant. For sparser, ambient sections with small energy changes, we optionally use a single power-law(poly) envelope

$$g_{\text{out}}(n) = \Big(1 - \tfrac{n}{N}\Big)^{\alpha_0}, \qquad g_{\text{in}}(n) = \Big(\tfrac{n}{N}\Big)^{\alpha_0}, \qquad (4)$$

with hyperparameter $\alpha_0 \approx 2.5$, providing a smoother, less hazy transition. Alternative shapes can be swapped in using the same scheduling logic.

Context-to-envelope policy: We map a lightweight context vector $\mathbf{c} = (\Delta P,\ \text{section role})$ to a fade family and parameter. Conceptually, envelope selection can be framed as minimizing a loudness mismatch objective,

$$(f^\star, \theta^\star) = \arg\min_{f \in \{\text{eq, poly}\},\, \theta} \sum_{n=0}^{N-1} \big(|z[n]|^2 - P_{\text{target}}\big)^2 + \lambda\, \mathcal{C}_{\text{transient}}, \qquad (5)$$

where P_{target} is the running power target and $\mathcal{C}_{\text{transient}}$ penalizes pre/post-splice transients.

Look-ahead and hot-swap: A short look-ahead scheduler quantizes all splice times to the nearest bar boundary. This preserves the groove and allows "hot-swaps": when an asynchronous preview mix or mastered version becomes available (see Section 5), the engine atomically switches sources at the next bar boundary with uninterrupted, beat-aligned playback.

5 Automatic AI Mixing and Integration

Mixing: When at least two sections are ready *and auto-mix* is enabled, Lumia triggers an asynchronous preview mix job while playback continues. The backend integrates the Tonn AI [18] for multitrack mixing; all calls are non-blocking and results are inserted at bar boundaries in the Loop Engine. First, section WAVs and the Loop Engine files are uploaded as stems. The system then calls *mixpreview* with per-stem metadata, including `instrumentGroup`, `presenceSetting`, `panPreference`, `reverbPreference`, and `musicalStyle` derived from the session genre. A *task ID* is returned, and completion is detected via webhook. Once ready, the preview mix is downloaded, decoded to an `AudioBuffer`, and scheduled for a hot-swap at the next loop boundary with crossfade.

Mastering. For export-quality output, sections are first concatenated into a single stereo WAV using *pydub*'s *"AudioSegment"*. This file is then submitted to Tonn AI's album-style mastering service using the *mastering preview* request, which specifies the musical style, desired loudness, and sample rate (44.1 kHz). The system retrieves the resulting preview master, and the final master version. The mastered audio is decoded and, like preview mixes, is inserted into the loop at a bar boundary. If additional sections are added while a mastered preview is playing, the system appends the new material and re-runs the mastering process asynchronously without interrupting playback [11, 18].

6 Experiments

We conducted a formative evaluation with three professional audio engineers (4–6 years experience). Each participant used *Lumia* to compose a 120–150 s multi-section track by framing scenes, selecting instruments, and layering loops on-device. Sessions lasted $\sim$25–30 min and concluded with a short survey and semi-structured interview. The survey assessed five constructs: *co-creation/agency*, *musical quality*, *audio mapping*, *interaction/flow*, and *value/fit*, measured with multi-item Likert-type scales (1–7). Two additional 0–10 sliders measured *authorship share* and *expectation match*. Construct scores were computed as item means, following standard practice for summated ratings [4, 14]. The instrumented workflow and several survey items were adapted from Creative-AI studies such as PAGURI, a user-experience study of creative interaction with text-to-music models [19], and modified for *Lumia*'s real-time photo-to-music loop.

Mean construct scores are provided in (Figure 10: Collected numerical evaluation results10). On the 0–10 scales, *authorship* averaged 4.0 and *expectation match* averaged 6.3. Participants cited speed ("Is different started from images rather than a DAW template, this gives faster vibe-finding") and nuanced control ("Chorus lift with subtle bass change", "macro close-up calmed the drums"). Improvement requests included: (i) optional genre/BPM locks, (ii) micro-nudge layer edits, (iii) a visual-to-sonic mapping legend, and (iv) reduced latency. Reported use-cases ranged from rapid idea sketching to mood-boarding and creating backing tracks for short-form video.

7 Discussion and Future Work

Lumia demonstrates that multimodal language models, traditionally used for captioning or classification, can operate in real-time generative contexts, translating vision into structured musical prompts while preserving user intent. Informal use suggests it behaves more like an instrument than a static tool, supporting exploration, intuition, and creative ownership. Its camera-inspired physical design plays a critical role in shaping interaction, encouraging playful experimentation and environmental engagement. Users valued the balance of control and surprise in composing through looking, noting patterns in visual-to-audio mappings. Although Lumia offers a real-time generative workflow, several limitations remain. Planned hardware updates include extending the FX button for real-time modulation of pitch, duration, reverb, and volume, mapped to gestures such as shakes, strikes, or pressure input. The current reliance on open-ended scene analysis introduces variability and interpretability challenges; future versions may add intermediate prompt-editing layers to let users refine or constrain descriptions before generation.

Lumia's accessibility positions it as a tool for inclusive, co-creative music-making, education, and rapid prototyping, expanding opportunities for interdisciplinary practice. Embedding generative AI in cultural production carries risks such as overreliance on machine output, erosion of local styles through model bias, and dependence on proprietary services. These risks can be mitigated through transparency in how Lumia's visual-to-audio mappings and prompts are constructed, robust user control over generative parameters, and active community participation in shaping system evolution to ensure representation, cultural diversity, and genuine human–AI co-creation.

8 Conclusion

We presented Lumia, a camera-inspired handheld system for real-time, vision-driven musical composition. Its core contribution is a real-time vision-to-audio pipeline that uses GPT-4 Vision to transform captured imagery into structured musical prompts, rendered into coherent audio segments by Stability AI's text-to-music model. This is embedded in a camera-inspired physical interface that uniquely integrates scene framing, instrument selection, and audio layering into a continuous creative loop, enabling responsive, context-aware composition outside of screen-based workflows. The system's design philosophy prioritizes human–AI co-creation, supporting improvisation, expressive control, and cultural diversity while leveraging the generative capabilities of multimodal models. Together, these elements establish Lumia as the first portable, vision-driven music generation device to merge multimodal AI inference, tangible interaction, and live compositional practice into a unified, user-centered workflow.

9 Acknowledgements

We would like to thank Prof. Marcelo Coelho for his guidance and support throughout the development of this project. We are also grateful to our teaching assistants Sergio Mutis, Chenyue Dai, and Quincy Kuang for their feedback and technical insight as well. Special thanks to William McKenna for his mentorship and help with tools around the shop, and to the MIT MAD program and the 4.043/4.044 Design Studio spaces, where the 4.044 community of peers and instructors fostered a wonderful space for experimentation, iteration, and creative risk-taking.

References

[1] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. TagliasFacchi, M. Sharifi, N. Zeghidour, and C. Frank. Musiclm: Generating music from text, 2023. URL https://arxiv.org/abs/2301.11325.

[2] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–13, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300233. URL https://doi.org/10.1145/3290605.3300233.

[3] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press, 1990.

[4] J. Carifio and R. J. Perla. Resolving the 50-year debate around using and misusing likert scales. *Medical Education*, 42(12):1150–1152, 2008.

[5] E. Chang, Z. Chen, J. Labrune, and M. Coelho. Be the beat: Ai-powered boombox for music suggestion from freestyle dance. In *Proceedings of the Nineteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711978. doi: 10.1145/3689050.3705995. URL https://doi.org/10.1145/3689050.3705995.

[6] M. Coelho and J.-B. Labrune. Large language objects: The design of physical ai and generative experiences. *Interactions*, 31(4):43–48, June 2024. ISSN 1072-5520. doi: 10.1145/3672534. URL https://doi.org/10.1145/3672534.

[7] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, I. Sutskever, and J. Knight. Jukebox: A generative model for music. *arXiv:2005.00341*, 2020. URL https://arxiv.org/abs/2005.00341.

[8] Z. Evans, J. D. Parker, C. Carr, Z. Zukowski, J. Taylor, and J. Pons. Stable audio open, 2024. URL https://arxiv.org/abs/2407.14358.

[9] Google Brain Team. Magenta: Music and art generation with machine intelligence. https://magenta.tensorflow.org/, 2016.

[10] A. Guzhov, F. Raue, J. Hees, and A. Dengel. Audioclip: Extending clip to image, text and audio, 2021. URL https://arxiv.org/abs/2106.13043.

[11] J. R. (jiaaro). pydub: Manipulate audio with a simple Python api. https://github.com/jiaaro/pydub, 2011.

[12] S. Jordà. The reactable: tangible and tabletop music performance. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, page 2989–2994, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589305. doi: 10.1145/1753846.1753903. URL https://doi.org/10.1145/1753846.1753903.

[13] D. Liang, A. Laptiev, and M. Coelho. Vbox: Ai-powered radio for musical exploration. https://doi.org/10.1145/3672534, 2024. Referenced in LLO feature.

[14] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, (140):1–55, 1932.

[15] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. 2023. URL https://arxiv.org/abs/2301.12503.

[16] OpenAI, J. Achiam, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024. URL https://arxiv.org/abs/2303.08774.

[17] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. 2022. URL https://arxiv.org/abs/2204.06125.

[18] RoEx Audio. Tonn api documentation. https://tonn.roexaudio.com/, 2024.

[19] F. Ronchini, L. Comanducci, G. Perego, and F. Antonacci. Paguri: a user experience study of creative interaction with text-to-music models. 2024. URL https://arxiv.org/abs/2407.04333.

10 Appendix

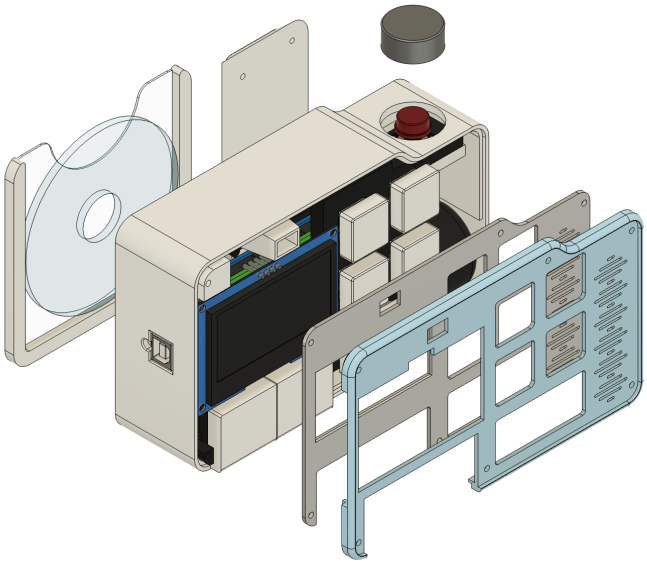

Figure 4: Interface exploded view

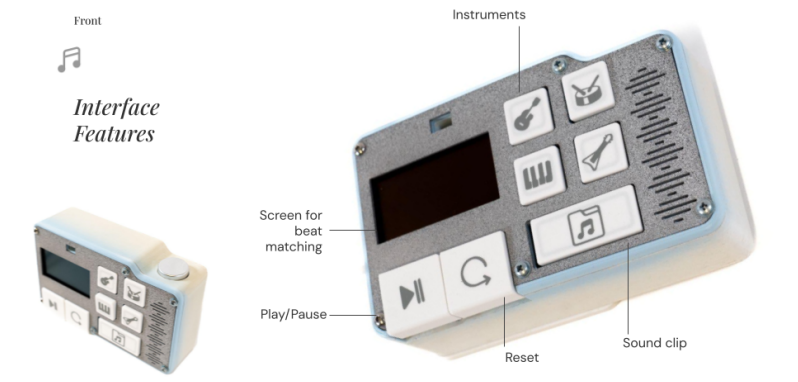

Figure 5: Front of LUMIA

Figure 6: Back of LUMIA

Figure 7: Prototyped Color Filter Disk Designs

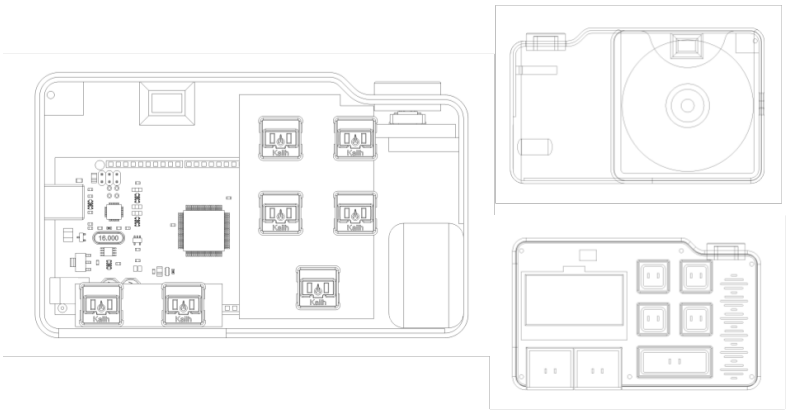

Figure 8: Schematics and Component Layout

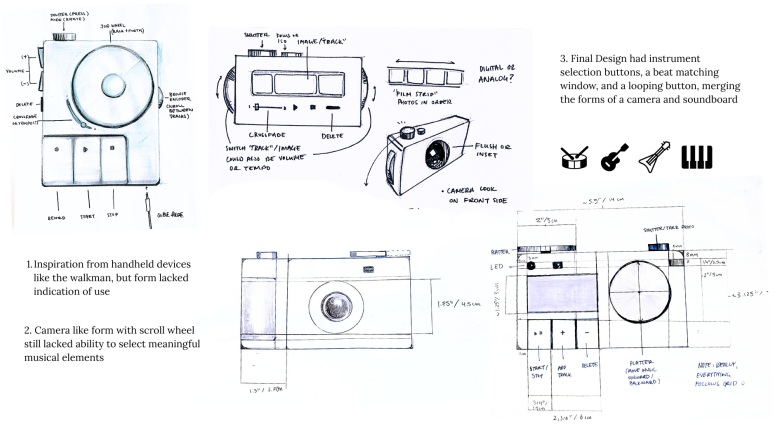

Figure 9: Form design evolution, driven by functionality and useability

10.1 Compute Resources and Latency

Lumia's music generation pipeline integrates cloud-based multimodal AI services accessed via REST APIs, orchestrated by a local browser-based frontend. All experiments were conducted using the following configuration:

Hardware:

- **Frontend:** MacBook Pro (Apple M2 Pro, 16 GB RAM) running Chrome 126
- **Microcontroller:** Arduino Nano 33 IoT (Cortex-M0+, 256 KB SRAM) handling physical I/O
- **Network:** 1 Gbps wired Ethernet or 300 Mbps Wi-Fi 6 connection

Cloud Services:

- **Image captioning** — OpenAI GPT-4 Vision API Input: 1280×720 px JPEG frame ($\sim$120 KB) Average latency: 1.2 ± 0.3 s Cost: $\sim$ \$0.002 per request ($\approx$120 input tokens, 200 output tokens)
- **Music generation** — Stability AI Stable Audio 2.0 API Input: single-sentence structured prompt ($\sim$35 tokens) Output: 15 s stereo WAV at 44.1 kHz Average latency: 3.8 ± 0.6 s Cost: $\sim$ \$0.14 per generation (1 API credit)
- **Automated mixing & mastering** — Tonn Audio Mixing API Input: up to 4 stereo stems, 15 s each Average latency: 5.2 ± 0.9 s (mixpreview), 8.6 ± 1.1 s (full master) Cost: $\sim$ \$0.05 per stem for mixpreview; $\sim$ \$0.15 for final master

Execution time:

- End-to-end latency (capture $\rightarrow$ audio in loop): 5.0–6.5 s
- End-to-end latency (capture $\rightarrow$ mixed audio update): 10–13 s

Compute load:

- All ML inference performed on provider-side infrastructure (no GPU required locally)
- Frontend CPU load $\leq 8\%$ during playback; memory footprint $\sim$400 MB with 4 concurrent AudioBuffers loaded

This configuration ensures reproducibility of performance results and can be replicated with comparable local hardware and a standard API subscription for the listed services.

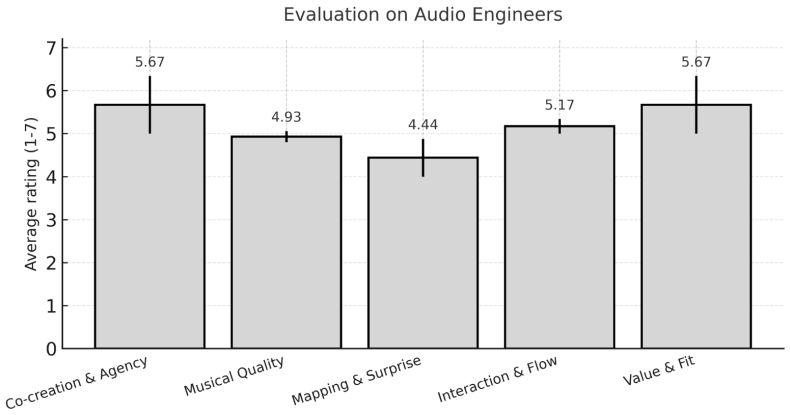

Figure 10: Collected numerical evaluation results

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:The abstract and introduction clearly state the system's technical contributions, including the real-time vision-to-audio pipeline, the handheld interface design, and the focus on human–AI co-creation. These claims align directly with the methods, experiments, and results presented in the paper, without overstating scope or novelty beyond what is substantiated in the body.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The abstract and introduction accurately describe Lumia's contributions, a real-time photo-to-music generation interface integrating GPT-4 Vision and Stable Audio, embedded in an embodied, camera-like device. The claims match the described methods, scope, and results without overstatement (Sections 1, 3–5).

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient methodological and implementation details to reproduce the pipeline, including hardware configuration (Section 3.1–3.2), system architecture (Section 4), vision-to-music pipeline parameters, audio generation settings, and evaluation design (Section 6). A link to the github repository containing written code is also included in the appendix, which contains more detailed instructions on how to run the pipeline.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code, data, and instructions are publicly available via the linked GitHub repository, system architecture and other details are described in the main paper.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [NA]

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: All compute requirements are documented in Appendix, including local hardware (Arduino, browser frontend), cloud APIs used (GPT-4V, Stable Audio 2.0, Tonn), average API latency, and per-generation credit cost. No high-performance compute required beyond a standard laptop.

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research complies with the NeurIPS Code of Ethics. All participants in the user study were informed of the research goals, data usage, and their right to withdraw; no personally identifiable information was collected or stored. The system poses no foreseeable harm, avoids biased or discriminatory model outputs by using generic prompts and instrumentation, and all AI services are used within their terms of service.

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper addresses positive societal impacts, such as lowering barriers to music creation, enabling non-experts to engage in composition, and supporting new modes of co-creative expression. Potential negative impacts are also acknowledged, including possible overreliance on AI-generated content, the risk of homogenization of musical style due to model biases, and dependency on third-party AI services. These considerations are discussed in the context of responsible deployment and future work.

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All external models, APIs, and datasets used in Lumia (GPT-4V, Stability AI's Stable Audio, and Tonn mixing API) are credited in the paper with proper citations. Their use complies with the providers' licensing and terms of service. No unlicensed or restricted assets were incorporated.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The paper explicitly details the use of GPT-4 Vision, a large multimodal language model, as a core component of the vision-to-music pipeline. Its role in structured scene analysis, prompt construction, and genre/mood inference is described in the Methodology section, with technical specifications and integration details provided.