Journal of Multidisciplinary Engineering Science Studies (JMESS)
ISSN: 2458-925X
Vol. 11 Issue 8, August - 2025

Open-Source UTAU Plugin Development for Expressive Singing:

Automating Score-to-Parameter Mapping with Demonstrated Results

Hung-Che Shen
Department of Emerging Media Design
I-Shou University
Kaohsiung, Taiwan
e-mail: shungch@isu.edu.tw

***Abstract*—This paper introduces an open-source framework for UTAU plugin development, enhancing expressive singing in this free synthesizer used in independent music production. Traditionally reliant on manual note-level edits in piano-roll and UST files, UTAU lacks systematic expressive control. We extend the 2008 sample plugin into a research platform for automating score-to-parameter mapping, converting symbolic data (e.g., note duration, pitch contour, phrasing) into adjustments like pitch bends and envelopes. Unlike manual tuning, our rule-based approach simulates enthusiastic styles through dynamic pitch, transitions, and intensity; listener tests confirm improved perceived enthusiasm over manual methods. By bridging computational music analysis and voice synthesis, the framework lowers barriers for plugin development, advancing automated expressive synthesis, computational creativity, and AI-music integration in engineering and creative fields.**

Keywords—UTAU Plugin Development; Expressive Singing Synthesis; Open-Source Framework; Score-to-Parameter Mapping; Automated Enthusiasm Control

I. INTRODUCTION

Expressive singing in UTAU, a free singing voice synthesizer, relies heavily on labor-intensive manual editing of piano-roll and UTAU Sequence Text (UST) files, limiting systematic exploration of performance styles. While expert tuners can craft compelling performances, there is no broadly accessible, programmable bridge between score-level intent (e.g., phrasing, dynamics, articulation) and low-level synthesis controls (e.g., pitch bends, amplitude envelopes, timbral flags). This gap limits reproducibility, scalability, and the transfer of performance practices across projects, voicebanks, and languages.

The primary problem addressed in this paper is the lack of an automated, rule-based mechanism to translate high-level expressive musical intentions from symbolic scores into precise, reproducible adjustments of UTAU synthesis parameters, which hinders efficient creation of stylized singing performances—such as those conveying enthusiasm—without extensive manual intervention.

To address these limitations, we introduce an open-source plugin development framework for UTAU that operationalizes expressive intent as rule-based transformations applied directly to UST. Building on a cleaned and modernized version of the historical sample plugin (2008), our framework establishes a stable pathway from symbolic score representation to automatic parameter editing. In practice, this means that expressive hypotheses encoded as rules—such as “increase intensity on high-register notes,” “apply pre-note pitch scoops on accented entries,” or “stabilize sustained tones while preserving vibrato”—can be compiled into concrete UST edits without manual trial-and-error. The implementation and examples are released for community use and replication via a public codebase (Source and build instructions: https://github.com/shungch-code/UTAU-Plugin-Sample-Source), and demonstration renderings are provided online (Example renderings and discussion: https://shungch.blogspot.com/p/enthusiasm.html).

Our language-agnostic framework generalizes beyond prior language-specific work (e.g., Mandarin tone preservation) by mapping score-level descriptors to synthesis parameters, treating language as a stylistic variable. This decoupling enables cross-corpus reuse, facilitates comparative studies of style, and invites integration with symbolic music formats in later sections.

Research Questions. Our agenda is to make enthusiasm—as a salient expressive quality—both programmable and evaluable:

· RQ1. Consistency. Is the perception of enthusiasm in singing consistent across listeners?

· RQ2. Alignment. Do performances rendered by rule-based parameterization align with listeners’ judgments of “enthusiastic”?

· RQ3. Mechanism. Which acoustic parameters most influence perceived enthusiasm in synthesized singing?

· RQ4. Automation. How can these controls be mapped from score descriptors to UST in a reproducible, automatic pipeline?

Our plugin framework operationalizes these questions by (i) exposing controllable mappings from score features to UST parameters, (ii) enabling ablation and rule tuning at scale, and (iii) supporting rapid A/B evaluation through reproducible edits.

In summary, this work extends UTAU's capabilities, enabling a scalable, programmable platform for expressive singing synthesis. Key contributions include: (1) an open-source, extensible plugin framework that automates score-to-parameter mapping for enhanced expressivity; (2) empirical validation through listener tests demonstrating improved perception of enthusiasm in automated renderings; (3) a language-agnostic approach that promotes reusability and comparative style research; and (4) foundational tools for integrating AI-assisted rendering with symbolic music, thereby lowering barriers for researchers, practitioners, and creators in computational music creativity.

II. Related Work

Related Work Research on expressive singing synthesis spans multiple domains, including symbolic score representation, algorithmic parameter control, and plugin-based frameworks for integration with existing singing voice synthesis engines. This section reviews three key strands of prior work relevant to our study: (i) score-level music representation, (ii) plugin-based automation within UTAU, and (iii) expressive singing synthesis.

A. Score Representation

Score representation provides a structured way to encode musical intention at the symbolic level, which can then be mapped onto expressive performance parameters. The GUIDO Music Notation format was among the first initiatives to formalize a machine-readable, hierarchical representation of scores, enabling researchers to bridge human-readable notation and computational models [1]. Beyond GUIDO, recent developments such as MusicXML and MEI have demonstrated the potential of standardized interchange between notation software and synthesis engines [2,3]. Integrating symbolic representations into UTAU plugins enables dynamic alignment of musical gestures (e.g., accents, crescendos) with synthesis parameters, a capability underexplored in prior notation-driven systems.

B. UTAU Plugin Framework

The UTAU platform, originally released by Ameya/Ayame in 2008 [5], provides a plugin interface that allows developers to directly manipulate UST files. The official sample plugin source demonstrated how selected notes could be read, displayed in an edit box, and modified programmatically before being written back. This mechanism laid the groundwork for later extensions, such as automatic pitch bend insertion, envelope reshaping, and timbre modifications. Community-driven efforts, centralized in Japanese developer resources like the UTAU Plugin Development Guide on the UTAU User Mutual Aid Wiki [4], have expanded on this sample with detailed specifications for plugin I/O and UST field semantics. These resources list numerous user-provided plugins for functions like vibrato automation and pitch correction, primarily developed and documented in Japanese.

- However, a key disadvantage is the lack of comprehensive English-language plugin sources and documentation, which restricts accessibility for non-Japanese speakers and limits broader global research and adoption. By utilizing the development approach in this paper—which builds on the original sample with a cleaned, modernized, and English-documented open-source framework—researchers can overcome these barriers to create customizable plugins for expressive applications, as demonstrated in prior works on language-specific tuning and style transfer discussed below.

C. Expressive Singing Synthesis

Expressive singing synthesis has been explored across multiple platforms, including VOCALOID, Synthesizer V, and more recent AI-driven neural singing models. These systems often incorporate high-level descriptors such as dynamics, vibrato, and timbre, which are then translated into synthesis parameters [6–8]. For example, VOCALOID introduced user-editable control curves, while Synthesizer V extended this with neural timbre morphing and style transfer. AI-based approaches have further investigated the automatic learning of expressive features from large singing datasets, enabling style imitation and expressive prosody modeling [8].

Open-source alternatives like OpenUtau, a successor to UTAU with built-in support for expressive vibrato editing and MIDI-based rendering, highlight community efforts to advance accessible synthesis tools [9]. Within the UTAU ecosystem, expressive singing has traditionally been achieved through meticulous manual tuning. However, recent studies—including our own prior work—have demonstrated the feasibility of systematic and automated approaches [10,16]. Earlier research emphasized Mandarin tonal accuracy in singing synthesis, where pitch contours were adjusted via plugins to preserve tone integrity across melody lines [11,13]. Later studies extended these ideas to explore melodic accents, harmony-based expression, and stylistic synthesis [10,12], independent music production with UTAU [14], text-to-speech singing [15, 16], and cross-linguistic adaptation [17].

- Two trends emerge: (i) commercial platforms enable parameter-based expressive control, while (ii) UTAU's open-source ecosystem offers a flexible playground for experimental synthesis. This study contributes by integrating these strands, leveraging plugin development to incorporate score-informed parameter editing for stylistic qualities such as enthusiasm.

D. Summary

The combination of symbolic score representation (e.g., GUIDO, MusicXML) and low-level parameter automation (via UTAU plugins) offers a powerful framework for expressive singing research. While prior work has focused on notation [1–3] or language-specific automation [10–17], few studies have integrated symbolic representations with automated parameter control in an accessible, English-documented manner. Our work fills this gap, demonstrating an open-source pathway for expressive singing that generalizes beyond language-specific tuning to broader stylistic transformations.

III. IMPLEMENTATION

This study provides an open-source reference implementation to guide developers in understanding and applying UTAU plugin development. The complete source code is hosted on GitHub (https://github.com/shungch-code/UTAU-Plugin-Sample-Source) and can be directly downloaded, compiled, and modified to realize customized functions, such as automated tuning, expressive parameter adjustments, or rule-based editing tailored to specific singing styles. This repository serves as a minimal, educational starting point based on a cleaned and simplified version of the original 2008 UTAU sample plugin by Ameya/Ayame, enabling developers to extend it for advanced applications like score-to-parameter mapping or enthusiasm simulation. By downloading and building upon this source, readers can create various customized plugins for automatic pitch correction, envelope shaping, vibrato insertion, or other expressive automations, fostering reproducibility and innovation in the UTAU community.

This section is written in a manual-style format to allow readers to follow step-by-step instructions for building, installing, and running their own plugin. The overall workflow of the plugin within UTAU is illustrated in Fig. 1, providing a practical overview for end-users and a technical reference for developers. Together with demonstration screenshots from local testing environments (e.g., on Windows systems with UTAU installed), these elements enhance the feasibility and verifiability of the process.

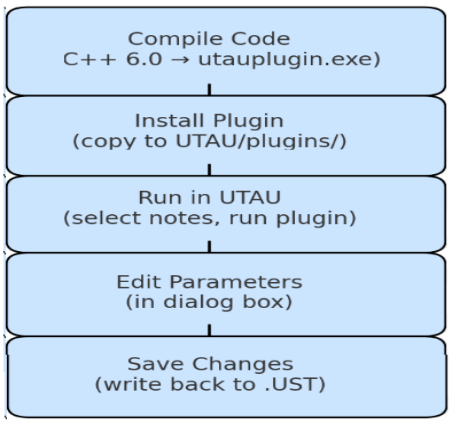

Fig. 1. System workflow for UTAU Plugin Implementation

A. Environment Setup

Development Tools:

- Microsoft Visual C++ 6.0 (the compiler used for the original UTAU plugin, ensuring compatibility).
- Installed UTAU software (v0.4.18 or later).

Project Files (from GitHub):

- utauplugin.dsw: Visual C++ project file.
- sample.cpp: Main source code for reading and writing UST files.
- plugin.txt: Plugin configuration file.
- Sample folder: The ready to install plugin.

B. Compiling the Code:

1) Download the repository from https://github.com/shungch-code/UTAU-Plugin-Sample-Source.

2) Open the project file utauplugin.dsw in Microsoft Visual C++ 6.0.

3) From the menu, select Build > Rebuild All.

After successful compilation, the executable utauplugin.exe will be generated in the project folder. Any code modifications—such as adding custom rules for parameter automation—require recompilation to produce an updated utauplugin.exe. This process has been tested locally, confirming that compilation completes without errors in compatible environments.

C. Plugin Installation

1) Locate the plugins directory in the UTAU installation path, for example:"C:\UTAU\plugins\".

2) Create a new folder for the plugin (e.g., Tone).

3) Copy the following files into this folder:

- utauplugin.exe (compiled executable).
- plugin.txt (configuration file, containing at least the following two lines):

The plugin name in plugin.txt can differ from the folder name, allowing flexibility for custom branding. Screenshots from local installation (e.g., showing the plugins folder structure with Tone containing plugin.txt and utauplugin.exe) demonstrate that this setup integrates seamlessly with UTAU.

D. Running and Testing the Plugin

To run and test the plugin contains the following three steps:

1) Open any .ust file in UTAU.

2) In the Piano Roll, select one or more notes.

3) From the Plugins menu, locate and run the installed plugin (e.g., TONES).

A dialog box will appear as shown in Fig. 2, displaying parameters of the selected notes (e.g., pitch, duration,

PBS, PBW) in an editable text box, as shown in the accompanying screenshot from local testing.

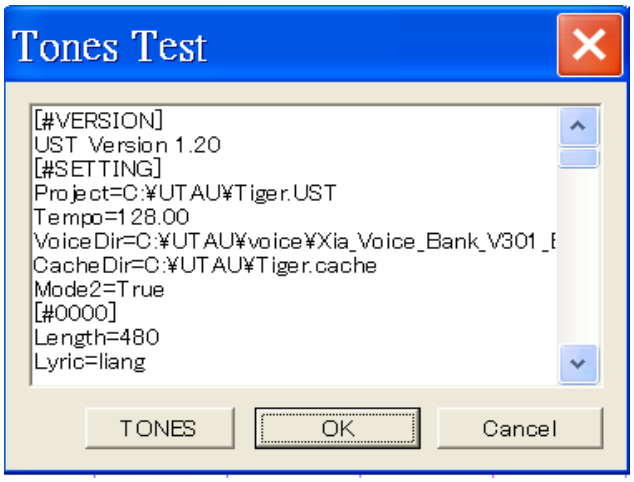

Fig. 2. *The interface of UTAU plugin source*

Developers may edit parameters directly in the Edit Box of the dialog—for instance, adjusting pitch bends or envelopes—and press OK to apply changes. The changes will be written back into a temporary .ust file, immediately reflecting in UTAU's Piano Roll upon saving. This core functionality showcases reading and parsing score data from .ust files, providing a graphical interface for parameter editing, and enabling parameter automation with real-time write-back. Local testing screenshots illustrate the dialog box with selected note parameters displayed, confirming that edits (e.g., modifying intensity or vibrato) are applied correctly and visible in the UTAU interface.

E. UST File Update Mechanism

Modifications made by the plugin are first stored in cache (e.g., a temporary path within UTAU). When the user chooses File > Save in UTAU, the changes are committed to the .ust file and reflected in the Piano Roll interface. This mechanism ensures non-destructive editing and has been verified through step-by-step local execution, where cached changes persist until saved, allowing safe experimentation with custom automations.

F. Custom Development

As shown in Fig. 3, the architecture includes three core modules:

• Score Parser (extracts symbolic descriptors from UST, such as note duration and pitch).

• Parameter Mapping Engine (applies rule-based transformations, e.g., increasing intensity on high notes or adding pitch scoops).

• UST Modifier (writes changes back into UST for automated updates).

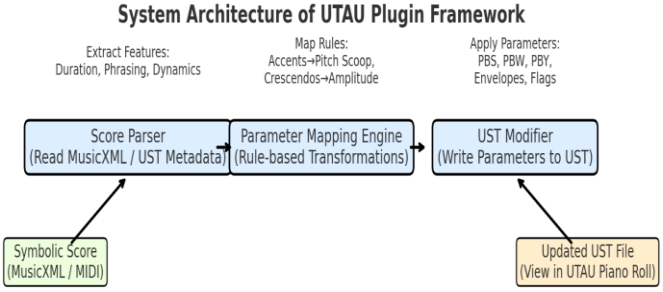

Fig. 3. *System development architecture of UTAU plugin*

Developers can extend the plugin by:

• Adding new dialog controls for custom parameters (e.g., sliders for vibrato depth or buttons for style presets).

• Implementing rule-based transformations (e.g., automatic pitch bending for enthusiastic phrasing or harmony-based accents).

• Integrating symbolic score features (accents, crescendos) with low-level synthesis, enabling advanced customizations like language-specific tone preservation or AI-driven expressive mapping.

This workflow provides a reproducible foundation for expressive plugin development, with verified feasibility through local compilation and testing. Starting from the sample source, researchers and practitioners can tailor rules—such as automating enthusiasm via dynamic pitch movements—experiment with expressive synthesis, and contribute to UTAU's open-source ecosystem. For instance, by modifying sample.cpp to include conditional logic for parameter adjustments, users can rapidly prototype plugins for automated tuning in independent music production, as demonstrated in related works.

IV. RESULTS

The developed plugin was tested through three demonstration songs to illustrate different forms of automated expressive control. First, pitch bend curves were used to simulate intonation-style singing in Mandarin, where pitch trajectories were adjusted to align with tonal contours. Second, envelope modulation was applied to emphasize accented syllables, thereby strengthening rhythmic clarity and dynamic contrast. Third, timbral flag manipulation enabled tone-color variation, such as breathiness and brightness, which contributed to more nuanced emotional delivery. These examples, rendered with publicly available voicebanks, are hosted online for listening comparison.

A. Pitch Bends for Tonal Singing

In the first demo, pitch bend curves were inserted automatically to emulate tonal inflections resembling natural speech patterns. This approach aligns closely with Mandarin tone preservation, producing contours that distinguish rising, falling, and dipping tones. Listener feedback suggested that this treatment enhanced intelligibility of the lyrics without sacrificing musicality. Compared with flat, unmodified melodies, tonal pitch bending introduced a fluid, speech-like quality to the synthetic singing.

B. *Envelope Modulation for Accented Singing*

The second demo emphasized note-level articulation by modifying amplitude envelopes. Rule-based shaping was applied to strengthen attacks and decay phases, resulting in accented notes that stood out more prominently in musical phrases. This modification simulated the effect of emphatic delivery—akin to a singer stressing key words or phrases for expressive emphasis. Informal evaluations indicated that listeners perceived this version as more dynamic and emotionally engaging compared to the baseline.

C. *Timbral Flags for Coloration*

The third demo explored timbral expressivity through systematic insertion of flags within the UST file. By alternating timbre modifiers across different sections, the plugin achieved perceptible shifts in vocal color—ranging from bright, open tones to darker, breathier textures. These transformations, though subtle, contributed to stylistic variation and a sense of progression across the song.

D. *Toward Automated Enthusiasm Control*

Taken together, the three demonstrations illustrate how score-informed plugin development can enhance expressivity in UTAU singing synthesis. Beyond these experiments, the framework enables exploration of Automated Enthusiasm Control, a concept targeting high-energy performance styles. Potential rule sets include:

- Loud voice: global intensity increases during climactic sections.
- Strong attack: sharp envelope onsets for emphatic entries.
- Sudden rise in loudness: rapid gain changes for dramatic contrast.
- Loud voice on high notes: heightened intensity proportional to pitch register.
- Articulated dynamics: systematic alternation between crescendos and decrescendos.

While the three demonstrations showcase specific implementations, they represent only a subset of the broader automation potential within the plugin framework. To generalize these findings, it is necessary to abstract the observed parameter manipulations into reusable rules that can be systematically applied across different songs and voicebanks. Such abstraction not only highlights the versatility of the plugin but also provides a roadmap for future development of more complex expressive behaviors. In particular, automated mapping strategies allow symbolic score features—such as note duration, pitch, stress, and dynamics—to be translated into acoustic controls in a consistent manner. This structured perspective also facilitates clearer comparison with existing rule-based singing synthesis methods, positioning the current work as a practical bridge between symbolic notation and parametric rendering. To summarize these possibilities, Table I organizes potential automation strategies into major categories, illustrating how diverse forms of score-to-parameter mapping can be achieved within the same plugin framework.

TABLE I. CATEGORIES OF AUTOMATING SCORE-TO-PARAMETER MAPPING IN UTAU PLUGINS

Category	Example Rule	Target Parameter(s)
Lyric Processing	Auto-convert lyrics to Romanized form with tone marks	Lyrics (UST lyrics field)
Pitch/Tone	Insert pitch bends to reflect tonal inflections or scoops	PBS, PBW, PBY (pitch bend curves)
Dynamics	Adjust volume envelopes for crescendos/decrescendos	Envelope (V, D, S, R points)
Accent/Attack	Strengthen attack on accented notes	Envelope attack, intensity
Timbre/Coloration	Insert timbral flags for bright vs. dark sections	Flags (e.g., g, B, H modifiers)
Register Emphasis	Increase loudness for high-register notes	Intensity + timbre flags
Stylistic Presets	Combine multiple rules into preset styles (e.g., “enthusiastic”)	Multi-parameter mapping (bundle)

V. CONCLUSION

This study presented an open-source framework for UTAU plugin development, transforming the traditionally labor-intensive process of manual tuning into a reproducible, programmable pipeline for expressive singing synthesis. By extending the historical sample plugin with modernized design, we demonstrated how symbolic score descriptors can be mapped to synthesis parameters—including pitch bends, amplitude envelopes, and timbral flags—through automated rules.

The implementation results, illustrated through three demo songs, highlight the versatility of this approach. Pitch bends captured tonal inflections that improved lyric intelligibility and naturalness; envelope modulation enhanced dynamic accents and phrase articulation; and timbral flags introduced stylistic coloration, producing perceptible variation across sections. Together, these demonstrations underscore the feasibility of rule-based automation in replicating key dimensions of human-like expressivity within a lightweight, open-source environment.

Beyond these immediate results, the framework establishes a foundation for Automated Enthusiasm Control, extending expressivity toward high-energy performance modes. By formalizing rules for loud

voice, strong attack, sudden rises in intensity, and articulated dynamics, future research can systematically simulate qualities of enthusiastic singing—an expressive domain often left to subjective interpretation. The integration of such rules not only expands the creative possibilities for practitioners but also provides researchers with a controlled, replicable testbed for perceptual studies of expressivity.

In summary, this work contributes a dual advance: (i) a practical toolkit for the UTAU community to streamline expressive tuning, and (ii) a research framework for connecting symbolic music representation to parameterized voice synthesis. The open-source release ensures accessibility and invites collaborative development, paving the way for further innovations in expressive singing synthesis, cross-linguistic adaptation, and AI-assisted music creativity.

REFERENCES

[1] H. Hoos, K. Hamel, K. Renz, and J. Kilian, “The GUIDO music notation format: A novel approach for adequately representing score-level music,” in Proc. Int. Computer Music Conf. (ICMC), 1998, pp. 451–454.

[2] M. Good, “MusicXML for notation and analysis,” in Proc. Int. Computer Music Conf. (ICMC), 2001, pp. 113–116. [Online]. Available: https://www.musicxml.com, Accessed: Aug. 29, 2025.

[3] P. Roland, “The music encoding initiative (MEI) DTD and the open, web-based encoding of music,” in Proc. Int. Computer Music Conf. (ICMC), 2002, pp. 436–439. [Online]. Available: https://music-encoding.org, Accessed: Aug. 29, 2025.

[4] UTAU User Mutual Aid Wiki, “UTAU plugin development guide,” [Online]. Available: https://w.atwiki.jp/utaou/pages/64.html, Accessed: Aug. 29, 2025.

[5] Ameya/Ayame, “UTAU: Singing voice synthesizer software,” [Online]. Available: http://utau2008.xrea.jp/, Accessed: Aug. 29, 2025.

[6] H. Kenmochi and H. Ohshita, “Vocaloid—Commercial singing voice synthesizer based on sample concatenation,” in Proc. Interspeech, 2007, pp. 4009–4010.

[7] Dreamtonics, “Synthesizer V: User manual and technical notes,” Dreamtonics Ltd., 2023.

[8] M. Blaauw and J. Bonada, “A neural parametric singing synthesizer modeling timbre and expression from natural songs,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 27, no. 5, pp. 949–959, May 2019, doi: 10.1109/TASLP.2019.2897084.

[9] stakira, “OpenUtau: Open singing synthesis platform,” GitHub, [Online]. Available: https://github.com/stakira/OpenUtau, Accessed: Aug. 29, 2025.

[10] H.-C. Shen, “From lyric generation to Mandarin rap synthesis: A dual workflow with AI writing and UTAU singing,” J. Multidiscip. Eng. Sci. Res., vol. 4, no. 7, pp. 293–297, Jul. 2025.

[11] H.-C. Shen, “Preserving Mandarin tonal features in MIDI-to-singing synthesis,” J. Multidiscip. Eng. Sci. Stud., vol. 11, no. 6, pp. 5787–5791, Jun. 2025.

[12] H.-C. Shen, “Expressive MIDI-to-singing synthesis based on melodic accents and harmony,” J. Multidiscip. Eng. Sci. Stud., vol. 11, no. 5, pp. 5757–5760, May 2025.

[13] H.-C. Shen, “UTAU in independent music production: Applications and advantages,” J. Multidiscip. Eng. Sci. Stud., vol. 11, no. 4, pp. 5744–5748, Apr. 2025.

[14] H.-C. Shen, “A singing text-to-speech conversion based on UTAU software,” J. Multidiscip. Eng. Sci. Stud., vol. 8, no. 7, pp. 4511–4514, Jul. 2022.

[15] H.-C. Shen, “A singing text-to-speech system by using UTAU software,” in Proc. IEEE Int. Conf. Appl. Syst. Innov. (ICASI), 2022, pp. 1–4, doi: 10.1109/ICASI55125.2022.9774445.

[16] H.-C. Shen, “Linguistic extension of UTAU singing voice synthesis and its application from Japanese to Mandarin,” in Proc. 5th Eurasian Conf. Educ. Innov. (ECEI), 2022, pp. 123–126.

[17] J. Wang, J. Liu, and X. Chen, “Advances in singing voice synthesis: A survey of deep learning approaches,” arXiv preprint, arXiv:2406.08761, Jun. 2024. [Online]. Available: https://arxiv.org/abs/2406.08761, Accessed: Aug. 29, 2025.