# Exploring Jazz Standards with Web Visualisation for Improvisation Training

Maximos Kaliakatsos-Papakostas
School of Music Studies
Aristotle University of Thessaloniki
maxk@mus.auth.gr

Konstantinos Velenis
School of Music Studies
Aristotle University of Thessaloniki
kvelenis@mus.auth.gr

Konstantinos Giannos
School of Music Studies
Aristotle University of Thessaloniki
kongiannos@mus.auth.gr

Emilios Cambouropoulos
School of Music Studies
Aristotle University of Thessaloniki
emilios@mus.auth.gr

## ABSTRACT

The availability of music data is increasing, along with the increase in computer capabilities, internet bandwidths and enhancement of web-related technologies. This increase creates an overload of information. While methods that allow exploration and retrieval of music evolve to compensate for this overload, such methods focus on commercial services and mainly involve community-driven data processing. This paper presents a methodology that focuses on the direction of content-based exploration of music material in symbolic form, for facilitating the retrieval of Jazz standards for music improvisation training. The objective is to allow users to visualise a large dataset of Jazz standards with colourful 3D representations, while setting checkpoints on specific pieces for making the exploration more efficient. The underlying methods employ harmonic similarity metrics, which is an important factor of Jazz improvisation training, and the presented implementation incorporates custom web technologies that improve the educational perspective of the music playback experience. Evaluation of the UI and UX provides pointers for future improvements.

## 1. INTRODUCTION

Improvisation is an integral part of the typical Jazz studies curriculum. Learning chord types and chord progression patterns, scales and tonalities are only some of the key harmonic elements that become useful for creating a palette with available tools for a Jazz (or any) improvisor. It is often suggested to listen to music and study genre-specific pieces to familiarise oneself with the style in question [18]. Having a tool that assists with this exploratory process of expanding one's repertoire would greatly improve the learning curve of a Jazz student in improvisation. This is the challenge addressed by the paper at hand: to take advantage of the benefits offered by the increase of available data in music and create important opportunities for making music education more engaging, effective and adapted to the specific needs and interests of both music teachers and students. In some cases, however, data availability becomes overwhelming and methods become necessary for helping users to retrieve data they need.

In huge datasets, identifying the user needs is not insignificant; some starting point of user preferences can be helpful for developing algorithms that present users with portions of data that might be relevant. Typically, music recommendation systems implement these functions through community-based similarity metrics, e.g., using collaborative filtering [15], and/or techniques for calculating similarity distances from the pieces' content, e.g., using feature extraction with neural networks [25]. Such systems usually operate under music streaming services, like Spotify or Pandora, and they are tailored to suggest tracks. Approaches based on community-driven aggregations, however, are not necessarily effective for music education, because in some cases the numbers of users and data entries are not always sufficiently large – but neither can data be too little. Such an example is the case of jazz standards and improvisation training. Here, it is often that musicians with diverse and individual backgrounds, become students at jazz, with already obtained specializations in playing other styles of music. In this example, it is important for them or their teachers to be able to explore available data and find the jazz standards that are best suited for each individual student, based on the content of each piece.

The use case scenario is that the user (student or teacher) can quickly browse through data, having as checkpoints one or two jazz standards that they are familiar with. During the exploration, the users need to be able to quickly select and listen to a real-time rendered jazz trio accompaniment piece. Additionally, all the pieces need to be available during the exploration, as well as organized visually in such a way, where "similar" pieces are placed closer together. "Similarity" is a commonly used concept in the discourse of music recommendation or large music databases. It often requires



*Web Audio Conference WAC-2022, July 6–8, 2022, Cannes, France.*

a context in order to be able to determine claim whether two objects are similar [10, 9]. When dealing with musical pieces, determining similarity is a very complex process, as it can be a very demanding task to recall musical pieces with a commonality, especially for a music student who is inexperienced.

Some approaches to incorporate content-based exploration of large musical datasets include Every Noise at Once project by EchoNest [1], a scatterplot is plotted where all possible musical genres found in Spotify are organised in a 2-dimensional map. They describe one dimension as organic versus mechanical/electric and the other as dense/atmospheric versus spiky/bouncy. Other attempts visualise a database on a 2D space with regards to assigned moods and genres [27]. A different approach proposes a Self-Organising Map constructed by a neural network following perceived sound similarity patterns [23]. An innovative interface arranges music databases in 3-dimensional virtual landscapes [16, 17]. According to a clustering of music pieces, virtual islands are generated each dedicated to a musical genre; close distances correspond to similar pieces belonging to the same style, and different ones are separated by large distances. They offer several modes, including the possibility for the user to listen to some specific songs, or to see typical words describing the music and explicit genre tags, or related images. A similar visualisation user interface has been recently proposed, alternating the virtual islands with city tower blocks [24].

Apart from the features related to the musical signal that are detrimental for similarity associations, there are elements found in the music score that are also relevant in this process. Such data ranges from information about the used notes, pitches, rhythm, tonalities, to chords and chords' functions, among others. In the harmony domain, the chord representation schema that is the most prevalent is the one used in Music Information Retrieval Evaluation eXchange (MIREX) [1] following the systematic syntax proposed by Harte [13, 12]. More recently, another representation has been developed which tackles more complicated pitch combinations accounting for idiom-independency [3, 2] and more detailed consonance/dissonance gradations [8], the General Chord Type (GCT). Additionally, there are studies with regards to modelling the functional element of tonal harmony [20, 4]. All this encoded information can be translated as features for harmonic similarity purposes [22, 5]. Chord progressions have been compared in terms of their distance to some tonic [6, 7], in terms of their alignment [11], or a generative grammar approach [5].

Although a great variety of exploring large music datasets exists, most of these methods are tailored to listeners and not students that learn how to perform a piece or some style of music. Typically, in Jazz music practices, it is expected to know a large array of standards, however, for an inexperienced student, this task is complicated and difficult to delve into. The purpose of this paper is to examine an architecture which enables a visually feasible exploration of Jazz standards. It supports setting checkpoints on specific pieces, familiar to the user, by employing harmonic similarity metrics, the user will be allowed to effortlessly browse the dataset of Jazz standards. Also, this architecture can be generalised to specialised large (musical or other) databases that can incorporate sequential data. This paper describes a 3D and RGB visualisation method that is based on combination of existing methods for sequential data and dimensionality reduction, emphasizing on the architecture, along with a custom CSV protocol accompanied by a custom web player.

[1] https://everynoise.com

# 2. METHOD DESCRIPTION

An overview of the processes and systems that lead to the audiovisual results is illustrated in Figure 1. Three distinct parts are involved: a) *Preprocessing* is the first step, where chord chart data are rendered to 3D coordinates and RGB values and to a CSV protocol based on MIDI that preserves chord information (root, type and position); b) *Server* is the system that hosts and serves the visualisation and CSV-MIDI to the UI; and c) *Client* is the web interface that hosts visualisation and CSV-MIDI playback. Red arrows in Figure 1 show user interaction for selecting a piece, based on 3D and RGB visualsation of data, and system responses to it, which results in listening to the selected piece.

## 2.1 Preprocessing

The preprocessing step is necessary in the current implementation for avoiding intense computation on generating graphics and music data. Regarding the visualisation part, except from the computation intensity aspects, there are also open scientific questions regarding the reduction of representation of entire chord charts to 3D data. The paper at hand incorporates and presents a novel method for such a reduction, however, the focus is not on evaluating the reduction per se; this paper focuses on the framework, i.e., the "3D and RGB Rendering" should be considered as a generic placeholder for such methods. The same holds for the "Jazz Trio CSV Rendering": any method can be considered that complies with the protocol presented herein.

### 2.1.1 Visualisation data preperation

Purpose of the visualisation step is to generate visualisations of a large music data base of jazz standard chord charts (1065 items) where each individual chart is represented by a coloured point in a 3D space. The position and color of point needs to be musically relevant, taking under consideration all aspects of music represented in a chart, e.g., chords, position of chords, time signature, tempo, tonality, etc. The available chart data are represented in string format as in the following example:

```
style~Open Swing,tempo~160,tonality~C,bar~4/4,c⌋
↪ hord~C\u03947@0.0,bar~4/4,chord~B\u00f87@0.⌋
↪ 0,chord~E7b9@2.0,bar~4/4,chord~Am7@0.0,chor⌋
↪ d~D7@2.0,bar~4/4,chord~Gm7@0.0,chord~C7@2.0⌋
↪ ,bar~4/4,chord~F7@0.0,bar~4/4,chord~Fm7@0.0⌋
↪ ,chord~B-7@2.0,bar~4/4,chord~Em7@0.0,chord~⌋
↪ A7@2.0,bar~4/4,chord~E-m7@0.0,chord~A-7@2.0⌋
↪ ,bar~4/4,chord~Dm7@0.0,bar~4/4,chord~G7@0.0⌋
↪ ,bar~4/4,chord~C\u03947@0.0,chord~A7b9@2.0,⌋
↪ bar~4/4,chord~Dm7@0.0,chord~G7@2.0,end
```

The method that has been developed for this application, integrates all information of the chart strings. An overview of the method is shown in Figure 2. This method involves a neural network with:

- two consecutive dense layers (256 and 512 units, Selu activation) connected to

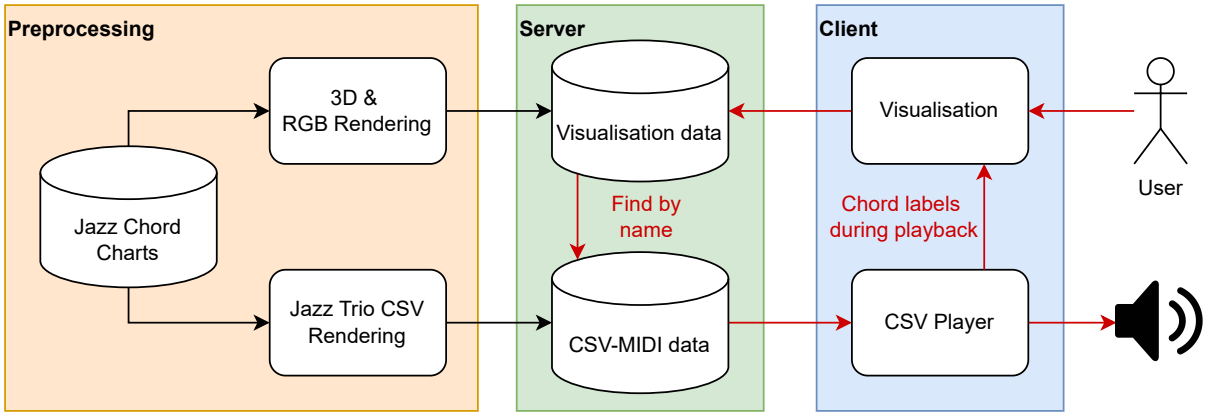


**Figure 1: Overview of systems and processes. Red arrows indicate forwarding of events and data after user interaction.**

- one Long Short-Term Memory (LSTM) [14] layer with 128 units connected to
- two consecutive dense layers (first with 128 units and the second with size-of-dictionary units).

This network learns to generate all chart strings in the database autoregressively, character by character. After training on a string that comprises a concatenation of all chart string, each chart string is passed separately from the network. After that, the internal cell states (128 values) and the hidden states (128 values) are concatenated, leading to a 256-dimensional representation of each chart string.

The next step is to generate a 3D reduction of the 256-dimensional representation of each chart; this is performed with the t-SNE algorithm [26]. The resulting 3D representations create clusters of charts that are not, however, clearly visible in the resulting web page. To this end, an additional step is proposed, the color clustering of points with the aid of distinct colors differentiation; these clusters are computed with the $k$-means algorithm [19]. Additionally, it is proposed to allow the user to adjust the granularity of clustering, and therefore the size of the similarly colored clusters, by having prepared clustering results from iterative applications of $k$-means for incremental numbers of clusters (from 2 to 20). The RGB values of the color that represents each cluster, is computed as a function of the 3D coordinates of the cluster centroid. All visualisation data (i.e., 3D coordinates from t-SNE and RGB cluster color values from $k$-means) a computed on the preprocessing stage and are readily available on the server. The implementation details on how the UI is built, are given in Section 2.2.1

### 2.1.2 *CSV-MIDI rendering*

Chart information needs to be rendered to a jazz trio orchestration, i.e., to specific notes for drums, bass and piano. This is performed by employing an implementation the Genius Jam Tracks (GJT)[2] music generation algorithm. GJT is an iOS application that generates backing tracks of given or user-generated jazz chord charts. Those algorithms are using AI and human played or annotated music data for generating jazz trio renderings in different harmonic and rhythm complexities. Harmonic complexity changes the underlying chords in a given jazz chart; similarly to the way a jazz musician performs substitutions of chords, GJT increases the harmonic pluralism in a given chord progression as the harmonic complexity increases, leading to more unexpected chord substitution. Increasing rhythm complexity changes the palette, the duration and the frequency that polyrhythms occur, i.e., the engagement of instruments in rhythmic patterns that "drift" from the tactus related to the time signature of the piece. Harmonic and rhythm complexities are defined by two respective integers with a minimum value 1 and maximum 5.

The exact algorithm of GJT is proprietary and, also, outside the scope of this paper. Regarding the implementation, which is the subject of interest in this paper, an implementation of the GJT trio rendering module has been integrated in a python wrapper that has endpoints for receiving requests about the title of the piece to be rendered, along with two integers for the desired harmonic and rhythm complexity. The responses to such requests are CSV files that contain information about notes, chords, bar-related information including time signature, style (e.g., swing, mambo, bossa nova etc.), section (e.g., A or B) and score-related information including tonality and tempo. A custom protocol has been designed and proposed for specifying the form of the information included in this CSV file for the application under examination. This protocol expands the standard MIDI protocol to include information about aspects of the chart that are necessary for conveying to the user chart-related information in real-time during playback; e.g., to show what chord corresponds to the notes that are being played at any instance.

Table 1 presents the specifications of the protocol (called CSV-MIDI) with examples. The identifiers in note event information refer to the instrument and/or their function in the piece. The available values (in the context of the presented jazz trio application) are: `Piano`, `Bass`, `Drums`, `Metro` and `Precount`. `Metro` refers to metronome events that are rendered as drums notes, if the user selects to activate metronome ticks, while `Precount` is also rendered as drums notes. `Precount`, if activated by the user, is an introductory drums part that starts before the actual piece, preparing the user for the exact initiation time of the piece. Overall chart information, always in the first row of the CSV file, includes information about the precount length (in quarter note length units); if the user deactivates precount, playback

[2]https://geniusjamtracks.com/

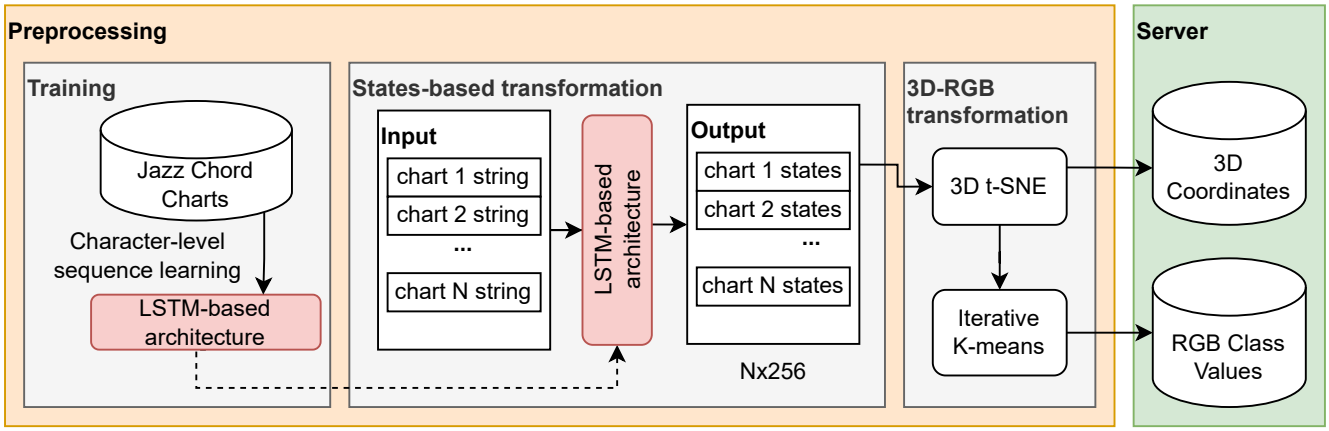


**Figure 2: Preprocessing pipeline for generating 3D and RGB visualisation data for the server out of jazz chord string sequences.**

**Table 1: CSV-MIDI protocol and examples.**

| | | | | | |
|---|---|---|---|---|---|
| **Overall chart information - first row** | | | | | |
| Spec. | Tonality | Style | Time sig. | precount lgth. | Tempo |
| Example | `C` | `Swing` | `4/4` | `8` | `130` |
| **Bar event information** | | | | | |
| Spec. | Id.∼Index@onset in piece | Style | Time sig. | Section | Tempo |
| Example | `Bar∼0@0.00` | `Swing` | `4/4` | `A` | `130` |
| **Chord event information** | | | | | |
| Spec. | Identifier | Label | Root-type expression | Onset in piece | Onset in chart |
| Example | `Chord` | `Dm7` | `D[0_3_7_10]` | `496` | `112` |
| **Note event information** | | | | | |
| Spec. | Identifier | Pitch (MIDI) | Onset in piece | Duration | Velocity (MIDI) |
| Example | `Piano` | `84` | `486` | `0.639583` | `71` |

begins directly at the beginning of the piece, i.e., from the time instance indicated by precount length. Details about the implementation of the player that reads and plays back CSV-MIDI files are given in Section 2.2.2.

## 2.2 Client-Server

The user is able to explore music pieces on the visualisation component of the client, select a piece to listen (having selected desired complexity levels for harmony and rhythm)

### *2.2.1 Visualisation implementation*

UI was developed using javascript, HTML and CSS with freely available additional libraries for optimised interaction (i.e., selectize.js to add search options, Bootstrap 4 framework for styling the menu and plotly.js for interactive plotting of the data). Visualisation and music playback (CSV-MIDI) data are fetched from the server using native javascript (`XMLhttpRequest()`) which is wrapped with a function `send_request_get_response(url, return_function)` that is used for flexible parametrization of requests to the server. The `url` argument represents the url where the desired JSON object is being served and the `return_function` argument represents the function that processes the JSON object; declaring the return function for each request is necessary, since different manipulation was necessary for different requested data (e.g., for updating the plot or the playback file).

The data plotting is based on the plotly.js library, providing the user with interaction capabilities such as zooming, rotation, scrolling and presenting information for each song via a tool-tip on the mouse hover event. Upon successful completion of the data plotting, the user, by clicking on the desirable song on the plot, can choose a song to load on the player and to retrieve useful music information such as the song's tempo, style, rhythm and tonality. The UI also provides piece selection from a dropdown menu for highlighting two different songs into the plot thus creating checkpoints for facilitating the exploration of graph areas around and/or between two jazz standards that the user feels comfortable in terms of improvisation. The player UI consist of the play/stop button and five boxes that render in real-time the current chord and the next four chord sequences, giving the user the possibility of playing along with the chosen song.

### *2.2.2 CSV-MIDI player*

A custom player has been implemented that is based on a metronome mechanism that activates events according to their onset in piece (according to the protocol described in Section 2.1.2). Note events are played using proper soundfonts for each instruments from the WebAudioFont[3] while chord events, when triggered by the metronome, are redirected to the chord display component (see Section 2.2.1). In the current implementation, no information is processed from bar events. The metronome is a simple clock mechanism that transforms broadcasts a `CustomEvent` approxi-

[3]https://github.com/surikov/webaudiofont

mately every 100ms (sensitive to the inherent inaccuracy of `setTimeInterval`). A timer starts with zero time when the user presses the "Play" button and the `CustomEvent` broadcasts the time that elapsed from the beginning of playback continuously, in quarter length multiples, computed as seconds times tempo value over 60.

The player listens to the events broadcasted by the metronome and keeps track of the row index in the CSV-MIDI file. When the time received by the metronome quarter-note time has progressed further that the time indicated in the "Onset in piece" column of the event of the current CSV-MIDI row, the row index starts increasing incrementally and events of these rows are activated until an event with greater onset value than the current received time is reached. Activation of note events is implemented by playing the note included in the "Pitch" column, with the indicated "Velocy" and with the defined "Duration", using the proper soundfont indicated by the "Identifier" column. Activation of chord events is implemented by redirecting the chord "Label" to the chord display component of the UI.

# 3. RESULTS

The final application, being in a prototype version, has been evaluated based on the Nielsen's usability heuristics [21]. According to these, the software must comply with ten fundamental principals discussed below. It will be necessary to evaluate a more mature version of the system, when it becomes available, in real-world educational conditions. The reader can examine the implementation on the web [4], where the method is applied in a set of 1065 jazz standards that are included in the Genius Jam Tracks commercial iOS application. The code can be found on github [5].

*Visibility of system status*: When all components are loaded, every component of the system is ready to use. The system presents no significant changes in its status, whatever the user input. Even in music playback mode, all components can be used; in this case, the play button switches to a stop button, clearly showing that the system status is in playback mode and the user is forced to press stop before pressing play again (possibly for another song). Furthermore, the system contains explanatory tags for each interaction element.

*Match between system and the real world*: The UI components involve commonly used terms in English (e.g. "Harmonic Complexity"), or commonly known music jargon (e.g., bpm - beats per minute - for tempo) and describes its functionality in plain words, familiar to the user's language e.g. "Select."

*User control and freedom*: Allowed user actions include the selection of two points/pieces on the map and playback of a piece and each action the user perform is rewritable. The ways that the user can change the outcome for such actions are trivial (e.g., select another piece on the map or "None" and press stop), while there is always the widely-known browser refresh button, for bringing the system back to its initial status.

*Consistency and standards*: Consistency is preserved by incorporating short and explanative descriptions in English on the UI, about the necessary actions to perform allowed tasks. Standard practices in mouse-based navigation of 3D graphs (e.g., scroll to zoom in/out) are inherited directly from the employed library (plotly), while standard UI components are used for selecting (dropdown menus) and playing (buttons) pieces. The user interface describes it's functions with common verbal phrases.

*Error prevention*: The only errors that can occur is when the user tries to press the play button when no piece is selected and when the user try to change the song, the harmonic or rhythmic complexity while a song is playing. These actions create a clear alert message with instructions.

*Recognition rather than recall*: The role of the two dropdown menus for piece selection is to mark specific positions on the map as checkpoints, for relieving the user from the need to painstakingly keep track of where specific pieces are when rotating or zooming in/out.

*Flexibility and efficiency of use*: The goal of the tool is to allow all users, regardless of their experience level in using the system, to explore the graph of jazz standards. The user interface has shortcuts for the advanced user e.g. zoom and turn the chart by rolling the mouse pad. A shortcut is introduced for increased efficiency regarding mouse-based navigation: by shift-clicking on a point/piece, playback starts immediately.

*Aesthetic and minimalist design*: The user interface is designed in a minimal manner allowing the user to concentrate to the function of the application.

*Help users recognize, diagnose, and recover from errors*: The system informs the user with verbal error messages that are easily understandable, explaining the nature if the problem.

# 4. CONCLUSIONS

This paper has presented an implementation for the exploration of a collection of jazz standards, with a focus on jazz improvisation training. Jazz standards are presented in a 3-dimensional mapping that results from a machine learning procedure, which processes harmonic annotations extracted from jazz standard charts to compute similarities between pieces. The visual component is integrated with a music player, which employs a custom representation for rendering chart chords to jazz trio music (piano, bass and drums). The developed representation incorporates note and chord-related information, allowing the presentation of the active chart chord on the user interface while the music plays. The user interface is evaluated according to heuristic criteria.

A necessary future step is the evaluation of the user interface, along with the effectiveness of an overall approach to jazz standard exploration in the context of jazz improvisation training. Furthermore, alternative visualisation techniques could be explored, including not only different data illustrations (e.g., 2D instead of 3D) but also altogether different methods for data processing. Such methods include, among others, the development of similarity metrics that offer coherent insight, since the LSTM-based approach that presented in this paper, functions as a black box.

## 4.1 Acknowledgments

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE. Project Acronym:

[4]http://155.207.188.7:5000/

[5]https://github.com/maximoskp/jazz_standards_visualization.git

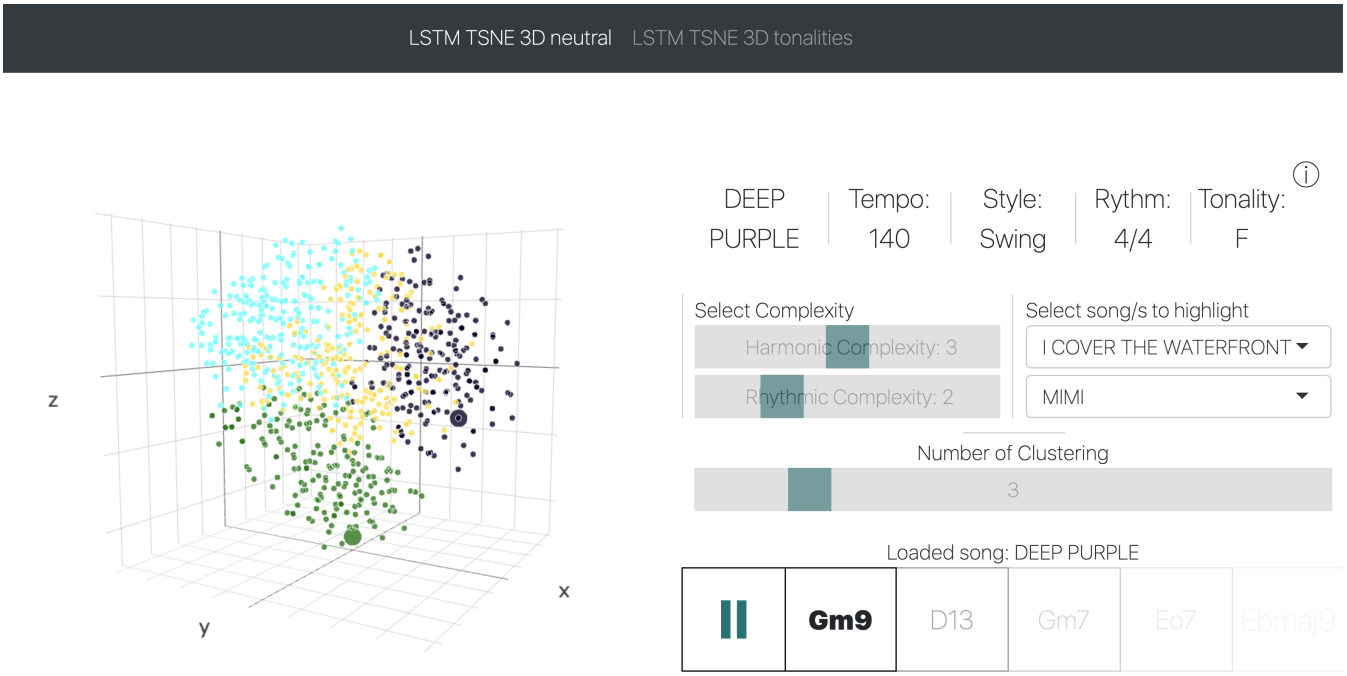


**Figure 3: Overview of the software UI.**

MusiCoLab, Project Code:T2EDK-00353

## 5. REFERENCES


[1] A. M. Barbancho, I. Barbancho, L. J. Tardón, and E. Molina. *Database of Piano Chords: An Engineering View of Harmony*. Springer, 2013.

[2] E. Cambouropoulos. The harmonic musical surface and two novel chord representation schemes. In *Computational music analysis*, pages 31–56. Springer, 2016.

[3] E. Cambouropoulos, M. A. Kaliakatsos-Papakostas, and C. Tsougras. An idiom-independent representation of chords for computational music analysis and generation. In *ICMC*, 2014.

[4] T.-P. Chen, L. Su, et al. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In *ISMIR*, pages 90–97, 2018.

[5] W. B. De Haas, M. Robine, P. Hanna, R. C. Veltkamp, F. Wiering, et al. Comparing harmonic similarity measures. In *7th International Symposium on Computer Music Modeling and Retrieval*, pages 299–315, 2010.

[6] W. B. De Haas, R. C. Veltkamp, and F. Wiering. Tonal pitch step distance: a similarity measure for chord progressions. In *ISMIR*, pages 51–56, 2008.

[7] W. B. De Haas, F. Wiering, and R. C. Veltkamp. A geometrical distance measure for determining the similarity of musical harmony. *International Journal of Multimedia Information Retrieval*, 2(3):189–202, 2013.

[8] K. Giannos and E. Cambouropoulos. Symbolic encoding of simultaneities: Re-designing the general chord type representation. In *8th International Conference on Digital Libraries for Musicology*, pages 67–74, 2021.

[9] R. L. Goldstone. The role of similarity in categorization: Providing a groundwork. *Cognition*, 52(2):125–157, 1994.

[10] N. Goodman. Seven strictures on similarity. reprinted in m. douglas and d. hull. *How classification works: Nelson Goodman among the social sciences*, pages 12–23, 1972.

[11] P. Hanna, M. Robine, and T. Rocher. An alignment based system for chord sequence retrieval. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 101–104, 2009.

[12] C. Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, 2010.

[13] C. Harte, M. B. Sandler, S. A. Abdallah, and E. Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR*, volume 5, pages 66–71, 2005.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] D. Kluver, M. D. Ekstrand, and J. A. Konstan. Rating-based collaborative filtering: algorithms and evaluation. *Social Information Access*, pages 344–390, 2018.

[16] P. Knees, M. Schedl, T. Pohle, and G. Widmer. An innovative three-dimensional user interface for exploring music collections enriched. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 17–24, 2006.

[17] P. Knees, M. Schedl, T. Pohle, and G. Widmer. Exploring music collections in virtual landscapes. *IEEE multimedia*, 14(3):46–54, 2007.

[18] D. Liebman, P. Markowitz, V. Juris, and B. Reich. *A chromatic approach to jazz harmony and melody*.

Advance music Rottenburg am Neckar, 1991.

[19] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[20] J. P. Magalhaes and W. B. de Haas. Functional modelling of musical harmony: an experience report. *ACM SIGPLAN Notices*, 46(9):156–162, 2011.

[21] J. Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 152–158, 1994.

[22] K. S. Orpen and D. Huron. Measurement of similarity in music: A quantitative approach for non-parametric representations. *Computers in music research*, 4:1–44, 1992.

[23] A. Rauber, E. Pampalk, and D. Merkl. The som-enhanced jukebox: Organization and visualization of music collections based on perceptual models. *Journal of New Music Research*, 32(2):193–210, 2003.

[24] M. Schedl, M. Mayr, and P. Knees. Music tower blocks: Multi-faceted exploration interface for web-scale music access. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pages 388–392, 2020.

[25] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. *Advances in neural information processing systems*, 26, 2013.

[26] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[27] R. van Gulik, F. Vignoli, and H. van de Wetering. Mapping music in the palm of your hand, explore and discover your collection. In *Proceedings of the 5th International Conference on Music Information Retrieval*. Queen Mary, University of London London, 2004.