Auto-Generated Beatmaps and Song Feature Visualization for a Rhythm Game

Wong Wing Yan

1155143285

Abstract **-** Melody Quest is a rhythm game that has been developed utilizing the PyGame framework in Python3. It allows users to import any desired song and generate a beatmap to play with. The present report aims to explore the principles of music information retrieval employed in Melody Quest. This will begin with an introduction to the game's user interface (UI). Subsequently, the report will delve into the techniques used for beatmap generation, which includes onset detection, melody extraction and separation, and pitch estimation. The second feature of the game, mood detection and waveform visualization, will be discussed in detail. This will involve a thorough investigation of the machine learning algorithms implemented to train a mood classifier.

Keywords – Rhythm game, PyGame, Music Information Retrieval, Beatmap Generation, Onset Detection, Melody Extraction, Pitch Estimation, Mood Detection, Waveform Visualization, Machine Learning

1. Introduction

The project is highly relevant to the course music information retrieval as it employs various techniques of music information retrieval for generating the beatmap and detecting the mood of a song. The project involves onset detection, melody extraction and separation, and pitch estimation, which are essential techniques of music analysis. These techniques are used to identify and extract significant musical features from a song, which can then be used to generate the beatmap. Furthermore, the project also incorporates machine learning algorithms, including Linear Regression, Naïve Bayes, SVM and Random Forest, to train a mood classifier for an imported song. Overall, the project demonstrates a practical application of music information retrieval techniques to create a rhythm game that is highly customizable and offers a personal and immersive experience.

The inspiration for the Melody Quest project stems from my interest in game development and my previous experience playing rhythm games. While I enjoy playing rhythm games, I found it limiting that they do not allow custom music. As someone who has studied music information retrieval this semester, I saw an opportunity to apply this knowledge to create a game that would allow players to import and play with their own music. Therefore, Melody Quest was developed in this project.

AutoRhythm [1] is an open source rhythm game that allows users to generate song maps automatically based on audio signal

processing. The game utilizes artificial intelligence to create unique notemaps for every song uploaded. It is mentioned that the future ideas for the game include rate modifiers, noteskins, visual and algorithmic improvements, machine learning algorithms, key configuration and export to other rhythm game formats. It is also stated that the biggest challenge of AutoRhythm is balancing the quiet and loud parts of a song to avoid either ending up empty or overly dense.

The paper "Music Emotion Recognition: Toward new, robust standards in personalized and context-sensitive applications" [2] provides a comprehensive review of the challenges and limitations of Music Emotion Recognition (MER). The authors propose a new framework for MER, highlighting important directions for future research such as open data and experimental reproducibility, model explainability and interpretability and ethical implications for MER applications. The complementary website for the article offers an extensive bibliography and a detailed overview of music and emotion datasets, making it valuable for research in the MER.

2. Methodology

2.1. Programming Language

The Melody Quest project has been implemented using Python3 and libraries such as Pygame and Tkinter. Pygame is a Python module that facilitates the creation of games. It provides fundamental components for game development like managing graphics, sound, and user input. Pygame has been utilized to develop the game's user interface as well as to handle the audio playback. Furthermore, Tkinter is a standard Python library that provides a graphical user interface toolkit for building desktop applications.

2.2. User Interface (UI)

Fig. 1, fig. 2 shows the user interface (UI) of Melody Quest. When the game is launched, the user is prompted to load a song from local. The game enters an introduction page after song selection. The introduction page shows "Press enter to Start" in the middle of the screen, which allows the user to start the main game at their desire. When the main game is started, the game's beatmap is displayed on the screen as some notes that fall down from each column. The notes of the beatmap are synchronized with the rhythm of the song. The player needs to hit the corresponding keys on their keyboard at the right timing. Also, the audio waveform is displayed on the upper part of the screen to provide a visual representation of the song with color matching to the mood of the song detected by mood classifier. At the top of the screen, the game displays two important metrics: combo and score. Combo represents the number of consecutive successful hits the player has achieved without missing a note. The score reflects the player's performance, taking into account the accuracy and timing of the hits. Overall, the UI has been designed to give an immersive and engaging playing experience.

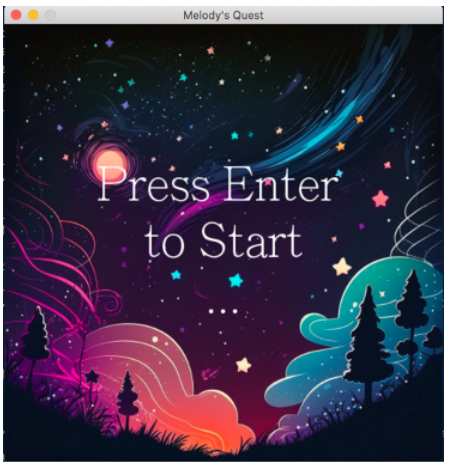

Fig. 1. The introduction page.

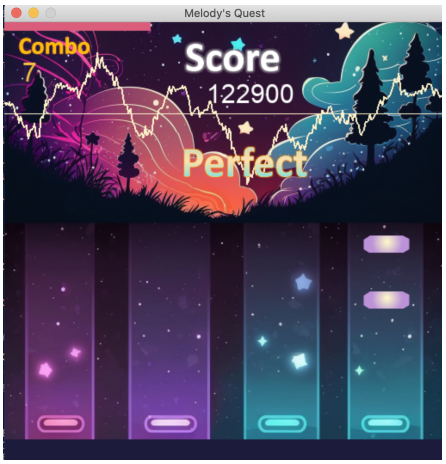

Fig. 2. The main game.

To create the background for the interface of the rhythm game, the Mid Journey AI tool is used. The first prompt asked for a vibrant, starry night sky with shooting stars and pastel, soft, colorful, ornate, intricate colors for a dreamy effect [3]. However, I felt that the design lacked a clear recognition of the beat, so I generated a second background using a prompt that included four columns of simple rectangles that light up and animate to the beat of the music [4]. I combined the first background image with the second image with four columns to create the final background image for the main game. Overall, the background design was intended to be amusing and give a twinkling effect.

2.3. Beatmap Generation

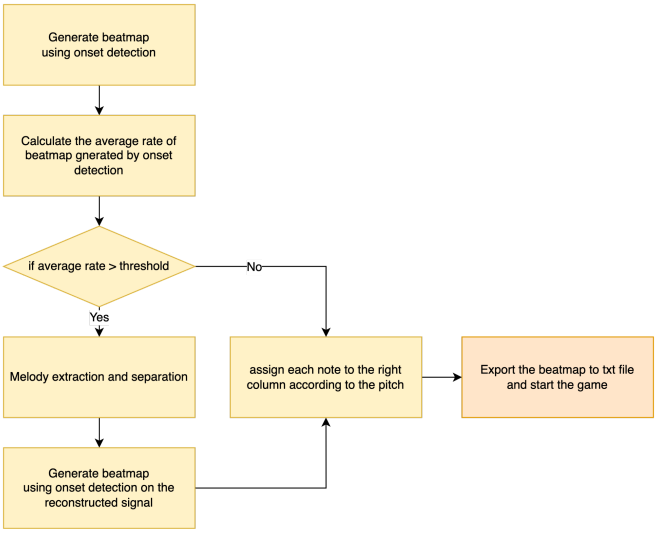

Fig. 3. The proposed methodology of beatmap generation

Fig. 3 shows the proposed methodology of beatmap generation using techniques in music information retrieval, including onset detection, music extraction and separation and pitch estimation.

Onset detection is a method that identifies the precise time at which each note is played in a musical composition. This technique is particularly effective for analyzing monophonic music, such as that produced by solo instruments. However, it is limited when it comes to more complex music involving multiple instruments playing different parts. In such cases, onset detection may generate beatmaps that are excessively fast to play. For instance, a beatmap that follows the accompaniment or harmony, rather than the ear-catching main melody, may be a poor gaming experience.

To address the problem of impractical beatmap generated, the average rate of notes appearing in the beatmap is calculated. If the value exceeds a predetermined threshold, melody extraction and separation will be performed to isolate the main melody from accompanying harmonies. Melody extraction and separation involves deriving the fundamental frequency (F0) of the main melody. Based on these F0 values and their harmonics, a binary mask for the melodic component is constructed. The binary mask of the accompaniment is defined to be the complement of the melodic component. [6]

Then, the onset detection can be performed on the reconstructed signal. This facilitates the generation of a beatmap that flows seamlessly along with the rhythm of the main melody.

After using onset detection to identify the exact timing of each note, pitch estimation is performed. The location of maximum autocorrelation is determined using the time information. The range of pitch in the entire song is then determined and each beat in the beatmap is assigned to the corresponding column based on its pitch. [7] This allows for a precise representation of the music in the generated beatmap.

The generated beatmap will be exported to a text file. Once the beatmap txt file is saved, it can be loaded into the game.

In the implementation, the algorithm of onset detection [5], melody extraction and separation [6] and pitch estimation [7] is based on the Python Notebooks for Fundamentals of Music Processing and musicinformationretrieval.com.

Remark: However, it is noted that performing melody extraction and separation on an entire song can be very time-consuming (taking up to 10 minutes). The waiting time for the melody extraction and separation process may discourage users from playing the game. To address this issue that the beatmap is excessively fast to play, a simple method is introduced where notes are randomly discarded when their time is too close to the previous notes, providing a more feasible solution for generating beatmap for complex music.

2.4. Mood Detection

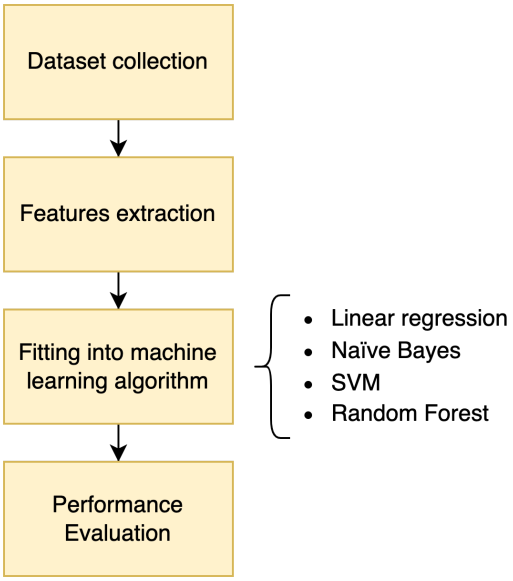

Fig. 4. The proposed methodology of mood detection

Fig. 4 shows the 4-step proposed methodology framework of mood detection.

After the first attempt to train a classifier using the dataset "Music and emotion stimulus sets consisting of film soundtracks" [8], the results were not satisfactory. The classifier achieved an accuracy of only 0.3888, which was insufficient for the purposes of this study. As a result, an alternative dataset was selected for further experimentation.

"The Emotion in Music Database (1000 songs)" [9] was then selected which fetched songs from the Free Music Archive (FMA). The dataset contains 744 songs after removing redundancies. The dataset is split into a development set (619 songs) and an evaluation set (125 songs). 45-second excerpts were extracted from the songs, which were then re-encoded to have the same sampling frequency of 44100 Hz. The dataset utilizes two fundamental dimensions, arousal and valence, to represent emotional states. Arousal refers to the intensity of an emotion. And, valence refers to the positive or negative of the emotion in the song. The annotators were instructed to rate the level of arousal or valence for the entire audio clip on a 9-point scale (from 1 to 9).

These ratings serve as the labels for the machine learning model. The model is subsequently trained to predict the degree of arousal or valence by utilizing various features extracted from the audio clips. The evaluation process shows the model's ability to predict the arousal or valence ratings for unseen audio clips.

The predictor is trained using machine learning algorithms based on a set of 57 audio features, including tempo, beat frames, chroma stft mean and variance, zero crossing rate mean and variance, spectral centroid mean and variance, spectral bandwidth mean and variance, spectral contrast mean and variance, and 20 Mel Frequency Cepstral Coefficients (MFCCs).

Table. I. Benchmark of Prediction Models

Run	Arousal		Valence	
	RMSE	R^2	RMSE	R^2
RND	0.16	0	0.15	0
BSL	0.12	0.48	0.15	0
TUM	0.10	0.59	0.11	0.42
UoA	0.10	0.63	0.12	0.35
UU	0.10	0.59	0.12	0.31

Table. I [9] presents the evaluation metrics used to test the performance of prediction models. Smaller values of Root Mean Square Error (RMSE) and larger Coefficient of Determination (R^2) indicate

better performance. The acronyms RND, BSL, TUM, UoA, and UU represent different benchmarks of prediction models, namely: random level, Baseline, TU Munich, University of Aizu, and Utrecht University.

Table. II. Performance metrics of Prediction Models

Run	Arousal		Valence	
	RMSE	R^2	RMSE	R^2
LR	0.1289	0.5997	0.1787	0.1485
NB	0.1299	0.5934	0.1750	0.1836
SVM	0.1311	0.5855	0.1815	0.1214
RF	0.1328	0.5745	0.1691	0.2377

Table. II presents the evaluation metrics of various models for predicting the arousal and valence in unseen music, based on the RMSE and R^2.

Linear regression is the most effective model for predicting arousal, performing similarly to the benchmark models. Random forest outperforms the other models for predicting valence. However, the performance of all the models for predicting valence is noticeably poorer than the benchmark.

The circumplex model posits that emotions are represented along two fundamental dimensions, arousal and valence. To visualize this two-dimensional space, emotions are plotted on a circular plane with arousal along the horizontal axis and valence along the vertical axis. By positioning an emotion in this circumplex space, its underlying mood can be better understood and characterized.

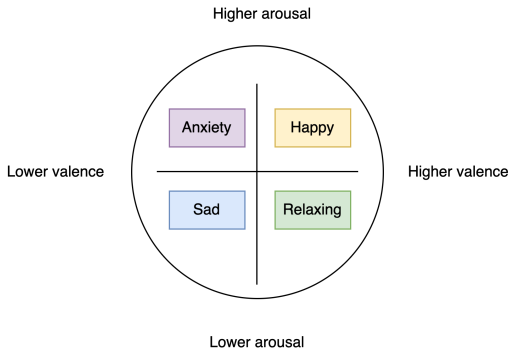

Fig. 4. The proposed classification of mood in circumplex model

Fig. 4 shows the proposed classification of mood in the circumplex model. Songs with high valence and high arousal are generally perceived as happy. Songs with low valence and low arousal are associated with sadness. Songs with high valence and low arousal are perceived as relaxing. Songs with low valence and high arousal are associated with anxiety.

2.5. Waveform Visualization

To visualize the waveform of the music played in the rhythm game, the librosa library was used in Python. Different colors were used to represent the mood detected in the music. The sound wave was synchronized with the music. The Pygame library was used to display the visualization on the game screen. The waveform plot was drawn on the screen using the Pygame draw functions.

Table. III. Mood-to-color Mapping

Mood	**Color**
Happy	Yellow
Anxiety	Purple
Relaxing	Green
Sad	Blue

Table. III displays a list of moods and the colors associated with each mood.

In the implementation, the function waveform_drawing samples audio data at regular intervals and draws lines between successive sample points to produce a waveform.

3. Conclusion

Throughout the development of the Melody Quest game, I have applied various music information retrieval techniques such as onset detection, melody extraction, and mood classification. I have successfully implemented a beatmap auto-generation feature, allowing the user to import any desired song and play. Moreover, I have also developed visualization of the song waveform and mood detection using a machine learning algorithm.

In the process, I have learned about Python programming, Pygame, and how to use them to develop a game's user interface. More importantly, I have gained knowledge in music information retrieval and machine learning. Overall, this project has been a valuable experience for applying the skills and knowledge in a practical way.

Reference

[1] C. Holt, Ø. Nordli, and W. Park., “AutoRhythm,” 2021. [Online]. Available: https://github.com/Garlov/AutoRhythm. [Accessed 28 March 2023].

[2] J. S. Gómez-Cañón, E. Cano, T. Eerola, P. Herrera, X. Hu, Y.-H. Yang, et al, “Music Emotion Recognition: Toward new, robust standards in personalized and context-sensitive applications,” *IEEE Signal Processing Magazine,* vol. 38, no. 6, pp. 106-114, 2021.

[3] MidJourney, “Prompt: Create a simple background for a rhythm game interface. The background should feature a vibrant, starry night sky with shooting stars and pastel, soft, colorful, ornate, intricate colors for a dreamy effect. Add a music element with swirling lights and sound waves. Overall, the design should be amusing and give a pulsating, twinkling effect that emphasizes the rhythm game.” [Online]. Available: https://cdn.discordapp.com/attachments/1008571088919343124/1090160385941831720/tracywong117_Create_a_simple_background_for_a_rhythm_game_inter_541dc973-3ba3-44fb-8f9a-a74be1d12517.png. [Accessed 28 March 2022].

[4] MidJourney, “Prompt: Create a simple background for a rhythm game interface with four rows of simple rectangles for easy recognition that light up and animate to the beat of the music. The background should feature a vibrant, starry night sky with shooting stars and pastel, soft, colorful, ornate, intricate colors for a dreamy effect. Add a music element with swirling lights and sound waves. Overall, the design should be amusing and give a pulsating, twinkling effect that emphasizes the rhythm game.” [Online]. Available: https://cdn.discordapp.com/attachments/1008571088919343124/1090156053519413248/tracywong117_Create_a_simple_background_for_a_rhythm_game_inter_a59c61a7-6103-4969-bbc5-4d4b5b1f6a1b.png. [Accessed 28 March 2022].

[5] M. Müller, "Onset Detection," 2015. [Online]. Available: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C6/C6S1_OnsetDetection.html. [Accessed: 29 Mar 2023].

[6] M. Müller and S. Rosenzweig, "Melody Extraction and Separation," 2015. [Online]. Available: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S2_MelodyExtractSep.html. [Accessed: 29 Mar 2023].

[7] "Autocorrelation," 2016. [Online]. Available: https://musicinformationretrieval.com/index.html. [Accessed: 29-Mar-2023].

[8] T. Eerola and J. K. Vuoskoski, "A comparison of the discrete and dimensional models of emotion in music," in Psychology of Music, vol. 39, no. 1, pp. 18-49, 2011, doi: 10.1177/0305735610362821.

[9] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, "1000 Songs for Emotional Analysis of Music," in Proceedings of the 2nd ACM International Workshop on Crowdsourcing for Multimedia (CrowdMM '13), Barcelona, Spain, 2013, pp. 1-6. doi: 10.1145/2506364.2506365.