Rhythm game beatmap generation using audio feature extraction and machine learning

CS310 Project Specification

Jai Sharma - 1409638

Background

This project is primarily based upon the free rhythm game “osu!” and in the development of new software tools to aid in the process of creating levels in the game. Typical gameplay of rhythm games consists of on-screen elements that are synchronised to game audio, which prompt the player for a specific input. The player must respond with accurate and synchronised inputs for the audiovisual cues to be rewarded with points.

Specific to “osu!”, a game level is composed of a specific audio track with placed timed visual elements, together known as a “beatmap”. The visual elements, known as “hit objects”, are placed and timed such that player input matches the rhythm of the beatmap audio track[0]. Beatmaps are manually created and submitted as player created works for other players to download, improve upon and play[1].

Problem Discussion

Creating beatmaps for the game “osu!” entails placing and timing hit objects within the games play area using an in-game editor. The play area itself is a two-dimensional plane in which hit objects are placed on, that appear at set time points with the rhythm of the beatmap audio. The play area itself is scaled to the player’s screen. Gameplay involves clicking the hit objects, using a pointing device, as they appear in the play area. Points are awarded according to the accuracy of the clicks with the rhythm of the beatmap audio.

The responsibility of player creating the beatmap, lies in creating a sufficiently challenging, yet satisfying experience for the player while also maintaining quality in adhering to features of the game audio. Beatmap creation is currently a tedious manual process that requires the positioning and timing of the hit objects to be done by hand, with the detection of audio features done by ear and transcribed into beatmap form at the discretion of the player creating the beatmap, known as a “beatmapper”. The hit objects must be set such that they appear synchronised with audio features. The x and y positions of the placed hit objects must also be carefully considered, as the player must use a pointing device to move from the position of one hit object to another. Poor timing and placement of hit object will lead to an unsatisfying player experience.

The problem this project intends to tackle is to create software to generate playable beatmap files from a given audio source, for the “osu!” standard game mode. This is intended to greatly assist the job of a beatmapper, in which a large amount of the tedious timing and placement of hit objects in the play area, are completed automatically. For the purpose of this project, non-gameplay, aesthetic elements of a beatmap, such as background images and storyboard elements, are not considered relevant.

There are currently existing projects that tackle similar problems for other rhythm games. One such example is “Dancing Monkeys”[2], which generates “stepfiles” for the rhythm game “Stepmania”. In comparison to “osu!”, “Stepmania” uses a different approach to the rhythm game genre, itself being a clone of the arcade based game “Dance Dance Revolution”. “Stepmania” is designed with the limitation that the user input is typically completed via the player, using their feet to step on arrows which correspond to the arrows displayed on screen. As such, there is a limit of using only four directional based arrows as input to not overwhelm the player in gameplay, as input requires significant physical movement. Furthermore, the stepfile must not require inputs from the player that are too fast or may be impossible to complete using only their feet as input. While “Stepmania” can also be played using a keyboard, where the player is therefore able to complete faster and more complicated inputs using their fingers, this is not considered part of the problem domain of “Dancing Monkeys”[3].

The requirements of a stepfile in “Stepmania” lead to characteristics where player input is intended to resemble dancing to the beat of the audio. This is represented in the methods chosen for audio feature detection and stepfile generation in “Dancing Monkeys”. In summary, it uses Audio Self-Similarity to both determine the beat of a given audio stream and to find patterns throughout the audio that resemble song structure. Energy levels are then calculated at the given beat points in which the positive representation of the waveform is used. Low frequency (bass) instruments in the audio are found to “dominate the energy representation.” The Audio Self-Similarity data is then used to correlate the energy values at beat-points to repeatable patterns which can then be converted to stepfile arrow patterns in the generated stepfile.

In comparison, there is a greater fidelity in audio feature representation in an “osu!” beatmap. Input from a computer pointing device can, more quickly and accurately, be positioned over screen elements, such that hit objects can be freely placed in the two-dimensional plane of the play area. From observation, an “osu!” beatmap’s structure is more ideally suited to representing individual notes and melody rather that patterns that represent variations in beat over time. As such it would be more pertinent to use audio feature extraction methods that favour detection of both onsets and transients rather than beat, as found in “Dancing Monkeys”.

The placement of the hit objects according to audio features can be seen as entirely player subjective. As such, the use of any given algorithm to determine such hit object placements can be difficult to classify as “good” or “bad”. As a result an approach to solve this is to take an aggregate judgement on hit object placement, using data from existing beatmaps that have been determined satisfactory from a community standard. This would involve using a machine learning model, trained from the data set of the existing beatmaps. A suitable implementation of this is the use of an artificial neural network, specifically a convolution neural network, which have been found suitable for processing two-dimension data[4], in this case both time period energy values of audio data.

Objectives

The aim of this project is to create a set of inter-operable software tools that may generate an “osu!” compatible beatmap file. The generated beatmap file should be considered a basis for a complete beatmap and may be further edited by hand to bring the quality of the beatmap to a playable standard. The input of these software tools should be an audio file in a commonly used file format and should output a beatmap file.

This project aims to produce three distinct software tools, each of which aims to fulfil a single goal, as outlined below:

Audio processing software

This software should be able to process audio from a file source and be able to perform audio feature extraction operations on the input file to produce meaningful data. This software should be able to output its extracted audio data into a parsable format. This software should be able to output reproducible audio feature data results for the same input audio files. Additionally this software should be able to generate and output it’s data in a reasonable runtime, ideally not exceeding the run length of the source audio input file.

Beatmap generation software

This software should be created to generate an “osu!” compatible beatmap file from given audio feature data provided as output from the audio processing software above. This software should be able to generate a beatmap containing placed hit objects from a function of the input audio data. This software should be able to generate hit object placements from a naive algorithm, created from simple fundamentals of hit object placement in relation to audio data.

This software should prominently feature a machine learning model that is able to generate beatmap files from input audio data. This model should be trainable using a data set of existing complete beatmaps of a known standard. All beatmap files generated from this software should be fully compatible with the original “osu!” rhythm game.

Beatmap visualisation software

This software should be able to visualise a complete or partially complete “osu!” compatible beatmap. This software should be fully compatible with the “osu!” beatmap format. This software should be able to display a simple representation of the “osu!” play-field, showing simplified version of hit objects, timed with beatmap audio. This software should be have functionality to pause, play, and seek within a beatmap.

Methodology

This project will require periods of research to gather supporting materials to explore existing techniques of both audio feature extraction and machine learning principles. Research materials gathered will support a formalised feature specification to apply to the development of the project software components.

Development of the software component of this project will utilise an Agile methodology. A Kanban framework will be used to visualise workflow of completion of software features. Software features will mostly be added as a result of the prior research, but additional features may be added mid-development if necessary. A focus on output of usable software will be taken where a new version of the software component should be output in an functional state, in one to two week intervals.

Timetable

Project timetable

Resources

The following is a non-exhaustive list of the resources that the project will use. The contents of this list may change over time as due to changes in software implementation as result of additional research. All of the resources in this list are free and open source projects, so there is very little risk of any becoming unavailable for use.

- Python and python libraries
  - Programming language used to implement the audio feature detection and beatmap generation software. Libraries include NumPy, SciPy and Tensorflow
- C++ and the SFML graphics library
  - Programming language and library to implement the beatmap visualisation software
- osu!
  - The free and open source rhythm game, used to verify compatibility with generated beatmaps
- Git
  - Version control to manage the project repository
- GitHub
  - Remote git repository service

Legal, social, ethical, and professional issues

This project aims to create software, intended to released as open-source. As a result, the use of third party libraries in the creation of this collection of software must be restricted to those which are licensed under an open-source compatible license.

References

[0] osu! wiki "Beatmaps" `https://osu.ppy.sh/help/wiki/Beatmaps/` [Accessed: 12 October 2017]

[1] osu! wiki "Glossary - Beatmap Submission System" `https://osu.ppy.sh/help/wiki/Glossary/#bss` [Accessed: 12 October 2017]

[2] Monket "Dancing Monkeys - Karl O'Keeffe" `https://monket.net/dancing-monkeys/` [Accessed: 12 October 2017]

[3] Dancing Monkeys: Automated creation of step files for Dance Dance Revolution `https://monket.net/dancing-monkeys/DancingMonkeys.pdf` [Accessed: 12 October 2017]

[4] Yann LeCun - Deep Learning and the Future of AI `https://indico.cern.ch/event/510372/` [Accessed: 15 October 2017]