CPAC PROJECT

Meeting 1

Main idea: drum pattern generation using Markov Chains

Decisions made

Languages:

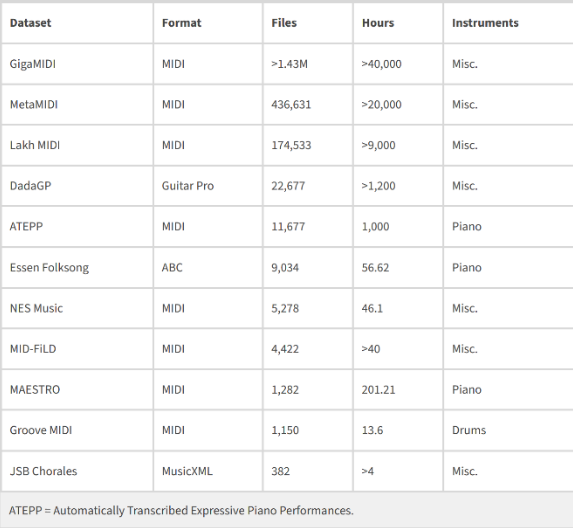
- Python

1st milestones:

- Data collection, annotation, augmentation (Ernest)
    - See if we can extract Spotify database
    - Take good data
    - Feature extraction
- Markov creation, start with order 5, update later if needed (Mattia, Giuliano)
- CHECK GITHUB AND INTERNET FOR ALREADY EXISTING SIMILAR PROJECTS

Software:

- Touch Designer



Dataset	Format	Files	Hours	Instruments
GigaMIDI	MIDI	>1.43M	>40,000	Misc.
MetaMIDI	MIDI	436,631	>20,000	Misc.
Lakh MIDI	MIDI	174,533	>9,000	Misc.
DadaGP	Guitar Pro	22,677	>1,200	Misc.
ATEPP	MIDI	11,677	1,000	Piano
Essen Folksong	ABC	9,034	56.62	Piano
NES Music	MIDI	5,278	46.1	Misc.
MID-FiLD	MIDI	4,422	>40	Misc.
MAESTRO	MIDI	1,282	201.21	Piano
Groove MIDI	MIDI	1,150	13.6	Drums
JSB Chorales	MusicXML	382	>4	Misc.

ATEPP = Automatically Transcribed Expressive Piano Performances.

Databases list

- **EMOPIA (Emotional Pop Piano)**
    - This dataset comprises 1,087 clips from 387 pop piano songs, each with clip-level emotion labels categorized into four groups. While the annotations are at the clip level, they reflect the mood within those segments, which may be useful for your needs.
    - frontiersin.org
- **VGMIDI**
    - VGMIDI is a dataset of piano arrangements of video game soundtracks. It contains 200 MIDI pieces labeled according to emotion and 3,850 unlabeled pieces. Each labeled piece was annotated by 30 human subjects according to the Circumplex (valence-arousal) model of emotion using a custom web tool.
    - paperswithcode.com
- **YM2413-MDB**
    - This dataset contains 669 audio and MIDI files from 1980s Sega and MSX PC games, featuring multi-label emotion annotations. The music is arranged with a subset of 15 monophonic instruments and one drum instrument.
    - paperswithcode.com

- **MidiCaps**
    - MidiCaps is a large-scale dataset of over 168,000 MIDI files accompanied by textual descriptions. Each caption describes the musical content, including tempo, chord progression, time signature, instruments present, genre, and mood. While the mood annotations are static, the dataset's size and diversity might still be beneficial.
    - arxiv.o
- **GigaMIDI**
    - The GigaMIDI dataset includes 56.8% single-track and 43.2% multi-track MIDI files. It contains 996,164 drum tracks and 4,338,224 non-drum tracks. The dataset is designed to facilitate research in expressive music generation and includes various features that might be useful for your project.
    - transactions.ismir.net

GPT recommendations

Extract Relevant Features

- **Tempo (BPM)**: Helps determine the drum pattern density.
- **Harmonic & Melodic Content**: Identifies the song's style and suggests suitable rhythmic elements.
- **Rhythm & Onset Detection**: Aligns the drum groove with existing rhythmic accents.
- **Emotion Detection**: Maps musical features to drum styles (e.g., fast & syncopated for energetic tracks, sparse for melancholic).

Build Your Markov Chain Model

- **State Representation**: Define drum pattern states (e.g., kick-snare-hihat combinations, bar-based subdivisions).
- **Transition Probabilities**: Train on existing drum patterns to establish probabilities for transitions between states.
- **Context Dependency**: Use higher-order Markov models (n-grams) to capture more realistic groove structures.

Training & Data Collection

- Use MIDI drum loops from different genres.
- **Public MIDI Datasets**:
    - **Groove MIDI Dataset (Google Magenta)** – Real drum performances with humanized timing.
    - **Lakh MIDI Dataset** – Large collection of MIDI files, including drum tracks.
    - **FreeDrumLoops.net, MIDIWorld, Cymatics MIDI Packs** – Various groove styles.
    - **Extract MIDI from DAW Libraries**:
- Extract probability matrices from existing patterns.
- Consider weighting transitions based on the extracted musical features.

Generation Strategy

- Start with an initial state based on the song's features.
- Generate a sequence following the trained Markov transition probabilities.
- Introduce variations using randomness or additional constraints (e.g., genre-specific rules).

Post-Processing & Refinement

- Introduce **humanization** (slight timing/velocity variations).
- Ensure drum hits align well with rhythmic accents in the input.
- Allow user control over complexity and intensity.

About the Markov Model:

- **Time Resolution**: Decide the subdivision level (e.g., 16th notes, triplets).
- **State Encoding**: Represent patterns as a sequence of drum hits:
  - Example (Kick, Snare, Hi-hat at 16th note resolution):
    K---S---K-H-S---H--- → (K=Kick, S=Snare, H=Hi-hat, -=Silence)
- **Extract Transitions**:
  - Count occurrences of each transition between beat subdivisions.
  - Normalize probabilities to create a Markov transition matrix.

About the Training

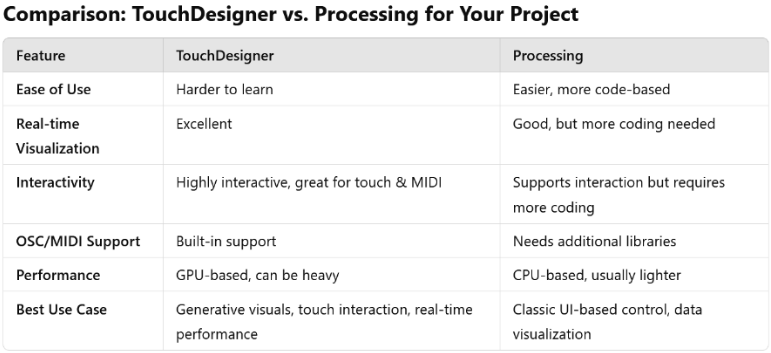
- **Genre-Specific Training**
  - Collect MIDI loops for different genres (jazz, rock, hip-hop, EDM, etc.).
  - Create separate Markov models for each genre or adjust probabilities dynamically based on extracted features.
  - Consider hybrid approaches: Allow cross-genre influences by mixing transition probabilities.
  - KNN for classification of styles? Good since no training required. Start with very simple tags (jazz, rock, ..., sad, melancholic,...)
- To expand training data:
  - **Swing & Groove Variations**: Shift timing slightly for more human feel.
  - **Velocity-Based Expressiveness**: Adjust hit intensities.
  - **Layered Sound Variants**: Modify hi-hat openings, ghost notes, and accents.
- If working with **audio input** rather than MIDI:
  - Use **Librosa** or **Essentia** to extract drum onset times.
  - Apply **beat tracking** to segment bars and identify repeating drum patterns.
  - Convert rhythmic transients into symbolic representations for Markov modeling

Interaction and UI ideas:

- Drum shown on the screen with highlights on the parts of the drum that are hit throughout the audio generation
  - Elements of the drum: kick, hi-hat, snare, crash, 2 toms, cymbals
  - Several samples for each part of the drum that the user chooses before running the code, or CHOSEN BY THE SOFTWARE
- Knobs to play with the agent's parameters, e.g. sensitivity to parameters (onsets, beats, mood, style, skill level).
- Dynamic display of the stave throughout the playout
- Graphic display depending on the features (maybe the drum display changes, we can add nice visual info)

NB: 1st prototype would not be real-time as it is easier to implement

NB: for later, implement style blending with Markov chains mix (or maybe another way).



Comparison: TouchDesigner vs. Processing for Your Project

Feature	TouchDesigner	Processing
Ease of Use	Harder to learn	Easier, more code-based
Real-time Visualization	Excellent	Good, but more coding needed
Interactivity	Highly interactive, great for touch & MIDI	Supports interaction but requires more coding
OSC/MIDI Support	Built-in support	Needs additional libraries
Performance	GPU-based, can be heavy	CPU-based, usually lighter
Best Use Case	Generative visuals, touch interaction, real-time performance	Classic UI-based control, data visualization