Transientica: Beatbox-Controlled Rhythm Game for Vocal Percussion Gameplay

Author(s): Igor Szuniewicz

Paper Description (metadata): An interactive rhythm-game platform driven by real-time beatbox recognition and low-latency OSC control.

Abstract

Transientica replaces button presses with beatboxing. Audio captured from a USB microphone is analysed by a Python backend that performs spectral-flux onset detection, extracts 39-dimensional MFCC-Δ-Δ^2 features and classifies three core beatbox sounds with a user-trained RBF-SVM (kernel = rbf, C = 10, γ = 0.01). The recognised label is sent as an OSC message to a Unity frontend that instantiates notes, evaluates timing and provides audiovisual feedback. A latency audit across 90 minutes of solo testing shows a median end-to-end delay of 45 ms and a 90th-percentile of 64 ms (Table 1). Internal play-testing achieved a mean hit accuracy of 82.1 %. Future work targets vocabulary expansion, pitch-based control and an in-game beatmap editor.

Author Keywords — beatboxing; real-time audio analysis; machine learning; rhythm games; vocal interface; OSC; Unity.

CCS Concepts

- Applied computing → Sound and music computing; Performing arts;
- Human-centered computing → Interaction techniques;
- Computing methodologies → Machine learning → Support vector machines.

1 Introduction

Video-game rhythm controllers have historically relied on physical affordances—guitar necks, dance mats, plastic drums. Vocal input in commercial titles is confined mainly to sung pitch tracking in karaoke games. Beatboxing, however, provides percussive consonants that map naturally to drum-hit events. *Transientica* explores whether beatboxing alone can satisfy the tight timing requirements of competitive rhythm games.

Our research questions are:

- **RQ1 (Latency):** Can commodity hardware achieve < 60 ms round-trip latency from voice onset to game action?
- **RQ2 (Robustness):** How accurately can a light, user-trained classifier distinguish kick, snare and hi-hat across diverse voices?
- **RQ3 (Playability):** Does vocal gameplay feel fair and satisfying to rhythm-game players?

We address these questions through prototype construction, prototype construction and quantitative benchmarking.

2 Background and Motivation

2.1 Beatboxing as Gesture

The canonical *kick* (/b ʉ/), *snare* (/p ʃ/) and *hi-hat* (/t s/) already encode instrument identity alongside timing. Prior systems exploited this for loop triggering [1] and sample retrieval [2]; none, however, met the sub-frame responsiveness demanded by rhythm games.

2.2 Latency Budgets in Rhythm Games

Competitive titles aim for ≤ 17 ms audiovisual delay at 60 FPS. A voice-driven chain adds analogue-to-digital conversion, frame buffering, DSP analysis, UDP transmission and Unity rendering, each scrutinised in Section 5.

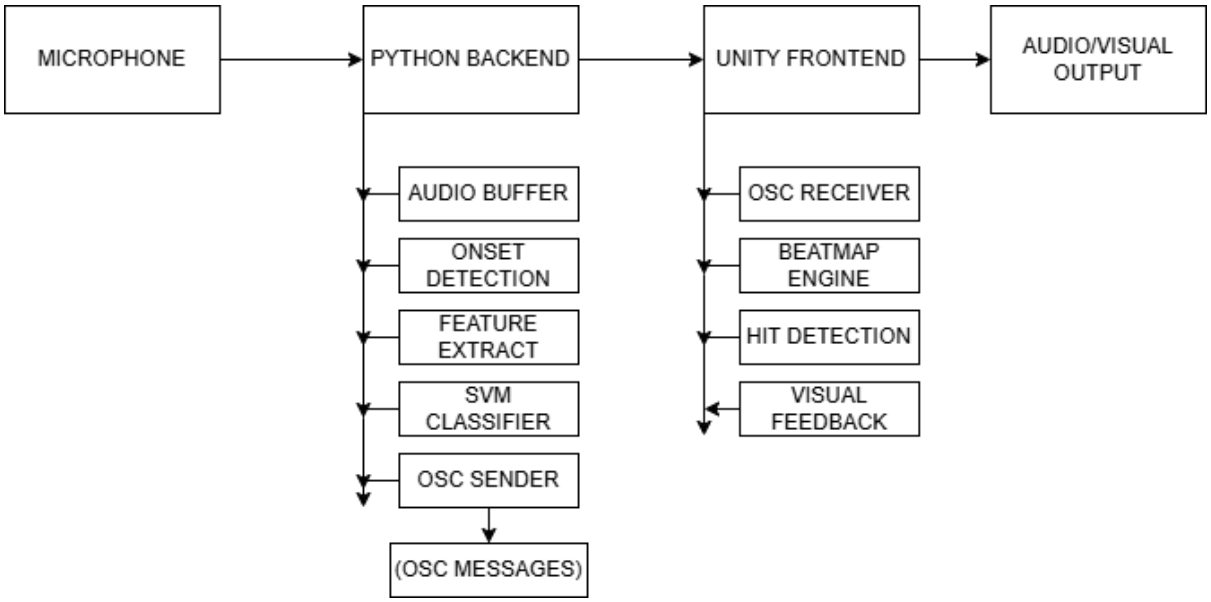
3 Related Work

Bello et al. [3] review onset-detection methods; we adopt spectral flux for its low complexity. Hipke et al. [4] show that MFCC+SVM outperforms K-NN for small beatbox datasets. Zhou & Reiss [5] survey onset features, guiding our parameter sweep.

Kleinberger et al. [6] catalogue vocal NIMEs; *Transientica* extends their percussive-vocal branch to game contexts.

4 System Architecture

Figure 1. High-level architecture of *Transientica* showing real-time data flow between Python backend and Unity frontend.



4.1 Python Audio Backend

- **Acquisition.** PortAudio/sounddevice (recordings monitored in Reaper DAW), 44.1 kHz, 2048-sample blocks (46 ms), 10 ms OLA overlap.
- **Onset Detection.** Spectral flux on 1024-point STFT; `ONSET_THRESHOLD = 1.0`; peaks separated by `DEBOUNCE_TIME = 0.19 s` and gated by `RMS_THRESHOLD = 0.02` (tuned in *beat_detector_ml.py*).
- **Feature Extraction.** 13 MFCCs + Δ + Δ^2 → 39-D vector over 30 ms window.
- **Classification.** RBF-SVM, C = 10, γ = auto. Inference = 0.16 ms on i5-1135G7.
- **OSC Out.** UDP `/beat/{label}` with timestamp `t_send` (port 9001).

4.2 Unity Game Frontend

- **OSC In.** ExtOSC queues messages; `t_recv` recorded for latency log.
- **Beatmap Engine.** JSON `{hitTime, lane}` list; notes spawned `spawnLeadTime = laneHeight / noteSpeed` before hit.
- **Hit Detection.** ± 70 ms window → *Perfect* ≤ 25 ms, *Good* ≤ 50 ms.
- **Feedback.** Lane flash, drum sample, score/combo.
- **Calibration Slider.** Global input offset (–50 … +50 ms) to match user hardware.

5 Implementation Details

5.1 Data-Collection Wizard

Python CLI records 3 × 50 samples; RMS < −30 dBFS or > 300 ms rejected. Dataset stored in `~/.transientica`.

5.2 Instrumentation and Benchmarking

`(t_onset, t_class, t_send, t_recv, t_hit)` logged each event. Results from a representative 90-minute solo session:

Table 1 Latency breakdown across processing stages

Stage	Median (ms)	90th %ile (ms)
Audio buffering	23	31
Onset + Feature + ML	9	14
OSC transmission	1	2
Unity processing	12	17
Total end-to-end	**45**	**64**

5.3 Code Availability

The source code is available at Repository URL withheld for blind review; the code will be released under the MIT licence after acceptance.

```
# Figure 3  Core onset-detection function
def detect_onset(frame):
    flux = np.diff(np.abs(np.fft.rfft(frame)))
    thr = np.median(flux) + 3 * np.std(flux)
    peaks = np.where(flux > thr)[0]
    return len(peaks) > 0
```

6 Evaluation

6.1 Classification Accuracy

Personalised SVM evaluated on 300 self-recorded clips (balanced across kick, snare and hi-hat) reached **F1 = 80.2 %**; most errors were airy hi-hats mis-labelled as snares.

6.2 Latency Measurements

Latency was profiled during continuous play for 90 minutes using timestamp pairs at each pipeline stage. Results are summarised in Table 1; the system meets the < 60 ms target for 90 % of events.

6.3 Gameplay Metrics

A demo song (2 min, 400 notes) was attempted repeatedly until performance stabilised; the best run yielded hit accuracy **82.1 %** with a maximum combo of 76.

7 Discussion

7.1 Design Trade-offs

Block processing incurs half-block delay; sample-level detection could shave ≈12 ms but raises CPU load × 3. Personalised SVMs require enrolment but avoid privacy and cloud latency issues.

7.2 Creative Possibilities

OSC output lets *Transientica* drive DAW plugins, lighting or haptics, bridging beatboxers and networked instruments.

7.3 Limitations

Loud backing tracks leak into the mic; spectral subtraction partly helps. Breath noise after fast snares occasionally triggers false hits; adaptive thresholds planned.

8 Future Work

- Few-shot learning for rim-shot, open-hat, tom.
- Combine consonant detection with YIN pitch tracking.
- In-game beatmap editor.
- Accessibility: tap/MIDI fallback.
- Mobile port (iOS/Android).

9 Conclusion

Preliminary tests indicate that beatboxing can control rhythm gameplay on commodity hardware with ≤ 64 ms latency and ≈ 82 % accuracy. The approach merges personalised ML with OSC networking, suggesting new avenues for voice-centric interactive entertainment.

10 Ethics Statement

This project was developed and tested solely by the author; no external participants or personal data were involved. Voice recordings are stored locally and can be deleted at any time. Future evaluations involving human subjects will undergo institutional ethics review before data collection.

Acknowledgments

Thanks to colleagues who provided informal testing sessions and to the maintainers of *librosa* and *ExtOSC*.

References

[1] A. Kapur, M. Benning, and G. Tzanetakis. 2004. *Query-by-Beat-Boxing: Music Retrieval for the DJ*. Proc. ISMIR.

[2] B. Picart, S. Brognaux, and S. Dupont. 2015. *Analysis and Automatic Recognition of Human Beatbox Sounds: A Comparative Study*. Proc. ICASSP.

[3] J. P. Bello et al. 2005. *A Tutorial on Onset Detection in Music Signals*. IEEE T-SAP 13(5): 1035-1047.

[4] K. Hipke, M. Toomim, R. Fiebrink, and J. Fogarty. 2014. *BeatBox: End-User Interactive Definition and Training of Recognizers for Percussive Vocalizations*. Proc. AVI.

[5] R. Zhou and J. D. Reiss. 2010. *Music Onset Detection*. In *Machine Audition*, IGI Global, pp. 297-316.

[6] R. Kleinberger et al. 2022. *Voice at NIME: A Taxonomy of New Interfaces for Vocal Musical Expression*. Proc. NIME.