

AN UNSUPERVISED SPECTRAL PITCH-CLASS BASELINE FOR AUDIO KEY DETECTION

Ziyi Xu

RESON (Independent Project)

2107383468@qq.com

ABSTRACT

This submission estimates a recording’s global musical key from its spectral pitch classes. A short-time Fourier transform supplies local peaks, which are mapped to twelve pitch classes and pooled over the recording. The resulting profile is compared with major and minor Krumhansl–Kessler templates at each tonic. The system has no training stage, learned weights, or dependence on an external service. It accepts one WAV file and writes a tonic and mode to a single-line text file. Local checks on synthetic chord mixtures exercise the program and its output format; they do not establish accuracy on recorded music. The submitted code is a separate public baseline for MIREX 2026, not the software used by the RESON website.

1. INTRODUCTION

Global key detection asks for a single tonic and mode for an entire recording. The task can be difficult when a piece modulates, emphasizes pitches outside the key, or obscures its tonal center with percussion and strong overtones. For this entry, a deliberately small approach was chosen: form a global pitch-class summary and compare it with fixed psychological key profiles reported by Krumhansl and Kessler [1]. This makes the submitted method straightforward to inspect and run. It is a baseline, rather than a claim of novel key-detection performance.

2. METHOD

2.1 Input and spectral analysis

The command takes an input path and a complete output path. It requires mono, 16-bit PCM WAV at 44.1 kHz. A Hann-window short-time Fourier transform uses 8192 samples per frame and a 2048-sample hop. Frames whose summed spectral magnitude is at or below the twentieth percentile are excluded. Within each remaining frame, the implementation finds local spectral peaks between 65 and 2093 Hz. Each peak is rounded to the nearest equal-tempered MIDI pitch using $A4 = 440$ Hz, then reduced modulo twelve to obtain its pitch class.

Peaks contribute the square root of their spectral magnitude. These weights are accumulated across frames in a twelve-bin vector, which is normalized to sum to one. This weighting moderates especially large peaks but does not remove harmonics or separate instruments. All settings

are fixed in the submitted program; no training corpus or development split is used.

2.2 Selecting the key

The observed vector is mean-centered and correlated with each of 24 candidates: the major and minor Krumhansl–Kessler profiles rotated to all twelve tonics. The candidate with the largest cosine-normalized correlation is selected. A tie is resolved deterministically. The program returns one of twelve tonic spellings followed by either `major` or `minor`; it does not return a confidence score or an “unknown” label.

3. SUBMISSION AND EXECUTION

The archive contains a Python 3.12 command-line program, a requirements file for NumPy and SciPy, and a README with installation and execution instructions. The evaluation command is `python key_detector.py %input %output`, with the environment’s Python and dependencies installed as stated in the README. The output is exactly one ASCII line, `tonic<TAB>mode<NEWLINE>`, in the task’s specified format [2]. It creates the output directory when needed and returns a nonzero exit status on unreadable or unsupported input. It does not make network calls at inference time.

4. CHECKS AND LIMITATIONS

The implementation was run on 24 synthetic chord mixtures, covering twelve major and twelve minor labels, and returned the intended label for each. These signals only check basic execution, transposition, and output behavior. They cannot stand in for an evaluation on recordings. No GiantSteps Key split was used for training, validation, parameter selection, or other development, in keeping with the task rule [2]. There is no claimed MIREX score in this abstract.

For a separate resource check, a two-minute 44.1 kHz synthetic C-major WAV was processed in 1.7 seconds including Python startup and reached about 365 MB peak resident memory on the local machine. This is an example of resource use, not a timing guarantee on the organizers’ hardware. The entire input and spectrogram are held in memory, so longer files require more RAM. The implementation uses one process and no scratch files. It can run without network access once NumPy and SciPy are installed.

MIREX reports both exact-key accuracy and a weighted score for related-key errors [2]. An exact match receives 1 point. A prediction a perfect fifth away receives 0.5, a relative major/minor 0.3, a parallel major/minor 0.2, and other errors 0. The submitted method optimizes none of these partial-credit categories directly: it always selects the largest template correlation. The weighted measure is useful for reading a result, but it would not turn a relative-key error into an exact detection.

Harmonics can shift the pitch-class profile away from the played fundamentals. Percussive spectra and chromatic passages can add unrelated peaks. A global summary also loses the order of musical events and cannot track a change of key. Relative major and minor keys may have similar pitch content, making their separation fragile for this method. These limitations are likely to be more pronounced in complex recordings than in the synthetic checks. There is also no segment-level output. A piece with different keys in its verse and chorus is reduced to one aggregate decision, which may reflect the louder or longer section. The classifier cannot explain which passage caused its choice.

5. RELATIONSHIP TO RESON

RESON is a separate music learning and creation website. This archive contains none of its website source, user records, private configuration, or deployed analysis pipeline. Any published evaluation result for this entry describes only the program submitted here and should not be presented as a measurement of the RESON product.

6. AI USAGE STATEMENT

AI assistance was used to draft the standalone implementation and edit this description. The reported local checks were run on the submitted program.

7. REFERENCES

- [1] C. L. Krumhansl and E. J. Kessler, “Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys,” *Psychological Review*, vol. 89, no. 4, pp. 334–368, 1982.
- [2] MIREX, “2026: Audio Key Detection,” https://music-ir.org/mirex/wiki/2026:Audio_Key_Detection, accessed September 28, 2026.