Pitch Estimation in Real Time: Revisiting SWIPE with Causal Windowing

Peter Meier[1 [0000−0002−3094−1931]], Sebastian Strahl[1 [0009−0007−9654−7762]], Simon Schwär[1 [0000−0001−5780−557X]], Meinard Müller[1 [0000−0001−6062−7524]], and Stefan Balke[1 [0000−0003−1306−3548]]

International Audio Laboratories Erlangen, Germany
peter.meier@audiolabs-erlangen.de

Abstract. Pitch estimation in real time is essential for a wide range of Music Information Retrieval (MIR) applications, including intonation monitoring, music education, and interactive systems. Many of these use cases, such as ensemble rehearsal settings, require low-latency, multi-channel processing on resource-constrained devices. While recent neural approaches offer high accuracy, they often fall short in real-time performance due to computational demands. In this paper, we revisit the well-established `SWIPE` algorithm and introduce `RT-SWIPE`, a real-time variant enabled by using causal windowing. We further propose a delay-tolerant evaluation metric that extends Raw Pitch Accuracy (RPA) to account for algorithmic delays. Experimental results on synthetic signals and multi-track ensemble recordings demonstrate that `RT-SWIPE` provides a practical balance of latency, accuracy, and efficiency. Although our study focuses on wind orchestra scenarios, the method is broadly applicable to similar real-time settings.

Keywords: Real-Time · Multi-Channel · Pitch Estimation.

1 Introduction

Real-time pitch estimation plays a crucial role in Music Information Retrieval (MIR), with applications ranging from intonation monitoring [7] and music education [17] to music scene analysis [8] and interactive gaming [13]. Many of these applications operate on resource-constrained platforms, such as smartphones, and demand low-latency, multi-channel processing [6]. In such contexts, lightweight and real-time capable algorithms such as `YIN` [5] are often favored over more computationally intensive methods like `PYIN` [10] and `CREPE` [9], despite potential trade-offs in estimation accuracy.

The `SWIPE` algorithm is a well-established method for pitch estimation, recognized for its robustness under real-world conditions [3]. Although not among the

Proc. of the 17th Int. Symposium on Computer Music Multidisciplinary Research, London, United Kingdom, 2025

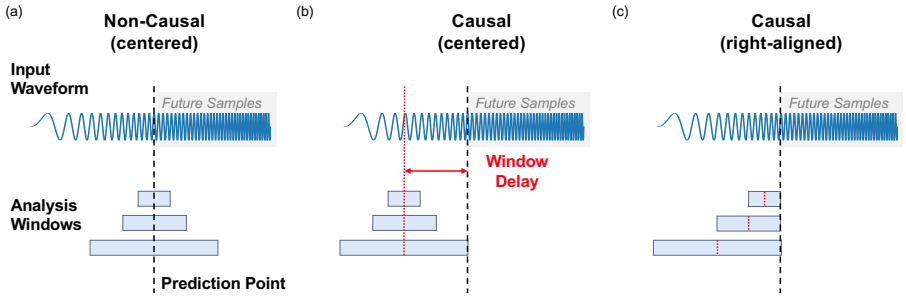

Fig. 1. Overview of window positioning strategies for real-time audio processing with `SWIPE`. **(a)** Non-causal, center-aligned windows used in the original (offline) `SWIPE` algorithm. **(b)** Causal, center-aligned windows with a constant delay for real-time processing. **(c)** Causal, right-aligned windows.

most recent developments, `SWIPE` remains relevant due to its resilience to noise and crosstalk—a critical factor in the context of wind music ensembles [12]. Its use of the Fast Fourier Transform (FFT) ensures computational efficiency, enabling real-time, multi-channel processing even on resource-constrained hardware. While neural network-based approaches may achieve superior pitch accuracy, their computational complexity and hardware demands often limit their applicability in low-latency, real-time scenarios. In addition, the structure of the spectral kernels in `SWIPE` offers a flexible foundation for further adaptations. Recent developments such as differentiable `SWIPE` (`dSWIPE`), which introduces trainable parameters, demonstrate the algorithm's ongoing relevance and adaptability [16]. `SWIPE` was originally developed for offline pitch estimation. Figure 1a illustrates this offline setting, where an input waveform is correlated with a set of analysis windows of varying lengths to estimate the pitch at a given prediction point. This configuration is inherently non-causal, as it requires access to future input samples.

A causal adaptation is shown in Figure 1b: By introducing an artificial delay equal to half the length of the longest analysis window, the algorithm can be rendered causal, as it no longer depends on future data. In the illustrated example, the input is a sinusoidal signal with linearly increasing frequency. Due to the uniform shift of all windows, pitch estimates for higher frequencies—which rely on shorter windows—may suffer from reduced accuracy, as the analysis does not capture the most recent signal content. This limitation can introduce a systematic bias toward lower frequencies, particularly problematic in musical contexts involving glissandi or rapid note changes.

As shown in Figure 1c, this issue can be mitigated by right-aligning all analysis windows with respect to the prediction point. This strategy ensures that each window captures the most up-to-date input samples. Similar approaches were used before, as demonstrated in [2], which explored frequency dependent latency for the Constant-Q Transform (CQT). While the overall delay—determined by the longest window—remains unchanged, the responsiveness of shorter windows

improves substantially. Since these windows correspond to higher frequencies, which typically exhibit faster temporal variation, the alignment helps preserve estimation accuracy during fast transitions. In contrast, longer windows associated with lower frequencies are less affected, as low-frequency content tends to vary more slowly.

In our research efforts, we aim to apply pitch estimation in ensemble settings, e. g., wind ensembles or orchestras. In such contexts, low latency is essential, as many channels must be processed simultaneously in real time. Furthermore, the presence of variable acoustics and interference from other instruments poses additional challenges, establishing ensemble environments as a demanding yet valuable testbed for real-time pitch estimation. Our previous studies identify `SWIPE` as the most suitable approach for these scenarios. Compared to `CREPE`, it is significantly less computationally demanding, and it offers greater robustness to cross-talk than `YIN` [12].

This paper continues this line of research by adapting `SWIPE` to operate in a causal manner, making it suitable for real-time applications. We also conduct systematic experiments on ensemble recordings, closely aligned with our target use case. During our evaluation, we observed that Raw Pitch Accuracy (RPA) exhibits limitations, particularly when applied to real-time scenarios. To address this, we propose an extension to the RPA metric that accounts for timing tolerances. While our experiments focus specifically on ensemble music, the underlying concepts and findings are applicable to a broader range of real-time pitch estimation tasks.

The remainder of the paper is structured as follows. Section 2 introduces the `RT-SWIPE` algorithm and analyzes its computational efficiency in multi-channel scenarios. Section 3 presents systematic experiments on wind ensemble recordings and compares `RT-SWIPE` to state-of-the-art pitch estimation methods. In addition, we examine the influence of algorithmic delays on the standard Raw Pitch Accuracy (RPA) metric and propose an extension to address its limitations in real-time settings. Finally, Section 4 summarizes our findings and outlines directions for future work. Additional materials and resources are available on a supplemental website.[1]

2 Real-Time SWIPE

Our real-time approach builds upon the original `SWIPE` algorithm [3]. In this section, we outline the key components of `SWIPE` that are relevant to our real-time modifications. The `SWIPE` algorithm estimates pitch by examining the harmonic structure of a sound. It utilizes pitch candidate kernels derived from a sawtooth waveform and correlates these with the spectrum of an input signal to identify the best match. The candidate with the highest correlation score is selected as the estimated pitch.

The `SWIPE` algorithm employs different window sizes to accurately estimate the short-time spectrum of an input audio signal. For a given set of predefined

[1] https://www.audiolabs-erlangen.de/resources/MIR/2025-CMMR-RTSWIPE

pitch candidates f_i, each represented by a frequency-domain kernel, we determine the optimal window sizes N_i that maximize the overlap between these kernels and the spectral lobes. These window sizes are calculated as follows:

$$N_i = \frac{8 \cdot f_\mathrm{s}}{f_i}, \tag{1}$$

where f_s stands for the sampling rate of the input signal in Hz. However, instead of computing individual spectrograms for each pitch candidate, `SWIPE` approximates this process by calculating spectrograms for a limited set of window sizes that are powers of two. This approach requires interpolation between window sizes, but significantly reduces computational costs [3]. To generate the spectral representation, `SWIPE` utilizes window sizes tailored to the pitch range. Let f_min and f_max denote the minimum and maximum expected frequencies in Hz, respectively. We define:

$$a = \left\lfloor \log_2 \left(\frac{8 \cdot f_\mathrm{s}}{f_\mathrm{max}} \right) \right\rfloor, \quad b = \left\lceil \log_2 \left(\frac{8 \cdot f_\mathrm{s}}{f_\mathrm{min}} \right) \right\rceil. \tag{2}$$

From these, the smallest and largest window sizes are calculated as:

$$N_\mathrm{min} = 2^a, \quad N_\mathrm{max} = 2^b. \tag{3}$$

Using these boundaries, we define a set K that includes all power-of-two window sizes used for spectrogram computation:

$$K = \{N_\mathrm{min}, 2 \cdot N_\mathrm{min}, ..., N_\mathrm{max}\} = \{2^a, 2^{a+1}, ..., 2^b\}. \tag{4}$$

For more details on the original method, we refer to [3].

2.1 Real-Time Audio Processing

In real-time audio processing, data is managed block-wise. These blocks, or frames, contain $H \in \mathbb{N}$ samples each and do not overlap. We refer to H as the hop size. When using `SWIPE` for real-time audio processing, three key considerations arise: (1) Real-time processing follows strict time constraints. The computation time, or *processing latency*, must not exceed a single frame period T_frame, given by:

$$T_\mathrm{frame} = \frac{H}{f_\mathrm{s}}. \tag{5}$$

This constraint is particularly relevant for multi-channel scenarios, where the processing latency may scale with the number of channels. (2) A single audio frame is typically smaller than the maximum window size $N_\mathrm{max} \in \mathbb{N}$ required for estimation. To address this, we need to implement audio buffering. (3) In real-time processing, only past data is accessible. Since analysis windows are usually centered, as illustrated in Figure 1b, estimates are delayed by half the maximum window size N_max, introducing *algorithmic delay.*

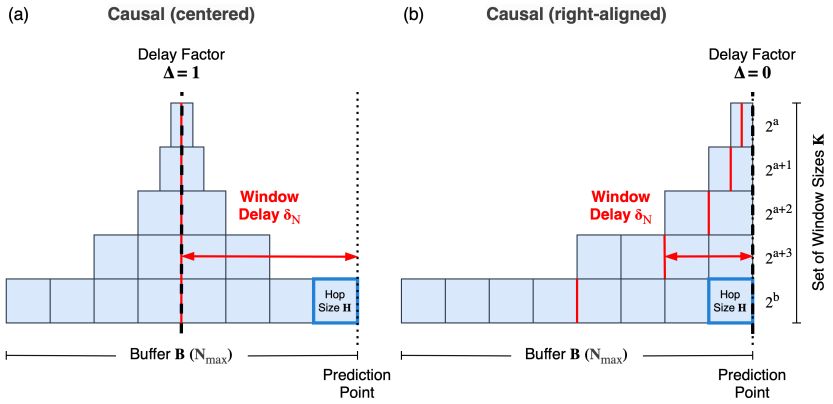

Fig. 2. Detailed view of adjustable window positioning with delay factor Δ for `RT-SWIPE`: **(a)** Centered windows share a common maximum delay. **(b)** Right-aligned windows use the latest audio data for estimation, each with individual delays.

To manage larger window sizes, we use a rolling buffer B that efficiently handles continuous audio input by cyclically adding new frames and discarding the oldest ones, defined as follows:

$$B \in \mathbb{R}^{C \times N_{\max}}. \tag{6}$$

$C \in \mathbb{N}$ is the number of audio channels, and $N_{\max}$ is the largest window size (see Equations 2 and 3).

For each new frame of audio input, an estimate is generated through a sequence of computations that must be completed within a single frame period T_{frame}. The following steps are performed:

(a) Update buffer B with the latest frame of audio input.
(b) For each window size $N_i \in K$, extract the corresponding data from the buffer according to the window positioning (see Section 2.2).
(c) Apply a Hann window to each window.
(d) Compute the Discrete Fourier Transform (DFT), vectorized over all C channels to ensure computational efficiency.
(e) Follow the original method to correlate the resulting spectra with pitch candidate kernels and select the candidate with the highest score as the estimated pitch.

2.2 Adjustable Window Positioning

In Figure 2a, we illustrate a more detailed view of causal centered windows as used in the `RT-SWIPE` method. All windows are not centered around the prediction point (compare to the non-causal case in Figure 1a); instead, each

window extracts data from the center of buffer B, containing only past samples. Consequently, estimates are delayed by half the maximum window size $N_{\max}$, even though more recent samples could be used for smaller windows (higher frequencies).

To address this, we propose a right-aligned window positioning (Figure 2b), where each window uses the most recent audio samples available in buffer B. This reduces the delay, especially for small window sizes, thereby affecting the expected algorithmic delay for high pitches. To interpolate between centered (offline) and right-aligned (real-time) window positioning, we introduce a normalized delay factor $\Delta \in [0, 1]$, where $\Delta = 1$ corresponds to centered and $\Delta = 0$ to right-aligned positioning.
For each delay factor Δ, the corresponding delay size D in samples is defined as

$$D(\Delta) = \left\lfloor \Delta \cdot \frac{N_{\max}}{2} \right\rfloor, \tag{7}$$

which describes the position at which the windows are intended to be centered, i.e., the position of the black dashed line in Figure 2. Depending on the individual window sizes N_i, we further introduce window delays (given in samples)

$$\delta_{N_i} = \max\left(\frac{N_i}{2}, D\right), \tag{8}$$

which represent the positions at which the windows are actually centered, considering that only past samples are available. These center positions are illustrated by red lines in Figure 2.

For right-aligned window positioning ($\Delta = 0$), the window delays δ_N vary, introducing a pitch-dependent delay. Low pitches corresponding to large windows have large delays, while high pitches with small windows have small delays. In contrast, centered window positioning ($\Delta = 1$) provides a frequency-independent delay determined by the maximum window size $N_{\max}$.

2.3 Multi-Channel Efficiency

For our research and field studies on wind music ensembles, we require not only real-time pitch estimation but also the ability to handle multiple input channels simultaneously. To this end, we investigate the multi-channel efficiency of the `RT-SWIPE` implementation and test how many input channels can be processed on consumer-level hardware.

As illustrated in Figure 3, we measure the real-time factor (RTF) as the number of input channels increases, identifying the maximum number of channels for which the RTF remains 1. An RTF smaller than 1 indicates computations take longer than a frame period, making it unsuitable for real-time processing. In contrast, an RTF greater than 1 means computations are faster than required, making it suitable for real-time applications. For our efficiency experiment, we conduct tests on a Mac Mini 2023 equipped with an Apple M2 Chip and 16 GB of RAM. We compare `RT-SWIPE` with an offline implementation of `SWIPE` [15],

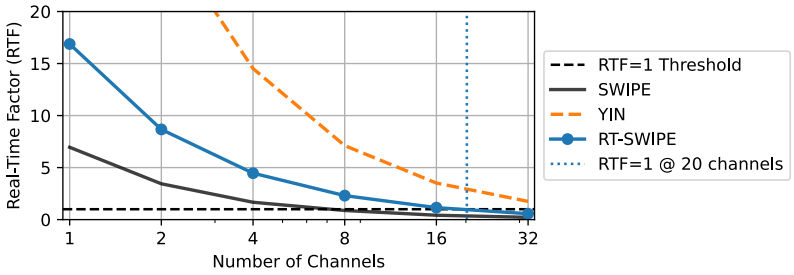

Fig. 3. Efficiency of `RT-SWIPE`: Real-Time Factor (RTF) over number of channels.

which is not optimized for frame-based processing and, therefore, serves as a lower performance bound. For an upper performance benchmark, we use the `YIN` implementation from librosa [11], a frame-based algorithm known for its speed and efficiency due to its autocorrelation method.

While the `YIN` implementation can process up to 38 channels on our test computer in real time, the offline `SWIPE` implementation can handle up to 7. In contrast, `RT-SWIPE` achieves a solid balance by supporting up to 20 channels in parallel. This highlights the trade-off between computational efficiency and robustness in pitch estimation, with `SWIPE`-based methods generally offering greater reliability in noisy, real-world scenarios compared to `YIN` [12].

3 Experiments

Dataset. Our experiments are performed on the `ChoraleBricks` dataset, a compilation of 193 multi-track recordings featuring wind instruments such as brass and woodwinds [1]. The dataset includes performances of 10 different chorale pieces, with soprano, alto, tenor, and bass parts played by different instruments. In addition to the multi-track recordings, `ChoraleBricks` includes F0 annotations, interactively generated using *Sonic Visualiser* (v5.0.1) [4] and the *pYIN VAMP plugin* (v3) [10]. These annotations were verified through sonification methods [14] and manually corrected as needed. The fundamental frequency (f_0) of reference annotations range from 38.19 Hz (played by a tuba) to 1250.97 Hz (played by a flute).

Experimental Setup. In our experiment, we utilize `RT-SWIPE` with right-aligned window positioning ($\Delta = 0$). We compare its performance against the original `SWIPE` method as implemented by the Python library `libf0` [15], and `YIN` [5], as implemented in `librosa`.[2] We also compare it with `CREPE` [9], using the official implementation available on GitHub.[3] The experiments are conducted with a

[2] We specifically used commit `ebd878f`, which includes recent updates and bug fixes for `YIN` and `PYIN`.
[3] https://github.com/marl/crepe.

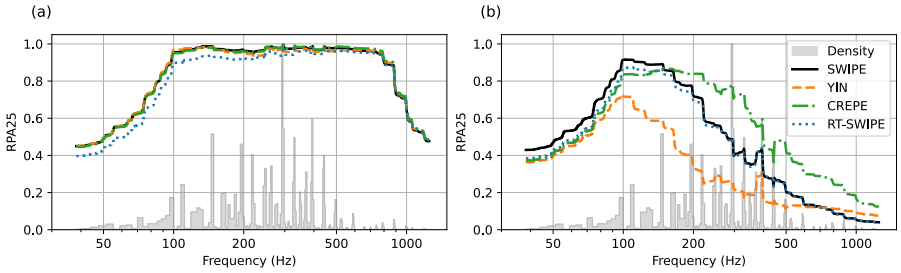

Fig. 4. RPA with a 25-cent tolerance over frequency on the `ChoraleBricks` dataset for `SWIPE`, `YIN`, `CREPE`, and `RT-SWIPE`. **(a)** Instruments without cross-talk. **(b)** Instruments with cross-talk: Target instrument signals are mixed with a randomly selected interfering instrument from a different voice, using a signal-to-noise ratio (SNR) of 5 dB. Background bars show the pitch distribution in the `ChoraleBricks` dataset.

sample rate $f_s = 44.1$ kHz and a hop size $H = 512$ samples. All estimators operate within a frequency range from note C1 (approximately 32.70 Hz) to note A6 (1760 Hz), effectively covering the entire range of the `ChoraleBricks` dataset. `SWIPE` and `RT-SWIPE`, the cent resolution is configured to 10 cents. For the pitch evaluation, we employ RPA with a tolerance of 25 cents.

3.1 Results

In Figure 4a, we present the frequency-dependent evaluation results of our experiment. For each estimated frequency, we check if it matches the reference within the cent tolerance. We sort this binary data (1 = correct; 0 = not correct) by annotation frequency and apply a Hanning window-based convolution to smooth the results and achieve RPA values over frequency. The RPA values for `SWIPE` are illustrated by the black solid line, those for `RT-SWIPE` are shown by the blue dotted line, `CREPE` is represented by the green dash-dotted line, and `YIN` is depicted with the orange dashed line. The step function, with a grey shaded area, indicates the density of f_0 reference annotations, helping to understand the frequency distribution in the `ChoraleBricks` dataset. With this, we identify that most notes are around 300 Hz, whereas notes at 50 Hz or 1 kHz are rarely played.

Across the dataset, `SWIPE` achieves an overall RPA of 0.960, while `YIN` records an RPA of 0.952, and `CREPE` achieves 0.961, indicating very similar performance among these algorithms. These results align with findings from previous studies [12]. However, `RT-SWIPE` shows a lower RPA of 0.931.

As illustrated in Figure 4a, the decrease in RPA for `RT-SWIPE` is frequency-dependent; all algorithms perform similarly above 500 Hz. For instance, at 200 Hz, the RPAs for `SWIPE`, `YIN`, and `CREPE` are 0.963, 0.948, and 0.959, respectively, compared to 0.922 for `RT-SWIPE`. At 50 Hz, the differences become more pronounced: `SWIPE`, `YIN`, and `CREPE` have RPAs of 0.490, 0.492, and 0.486 respectively, while `RT-SWIPE` falls to 0.434.

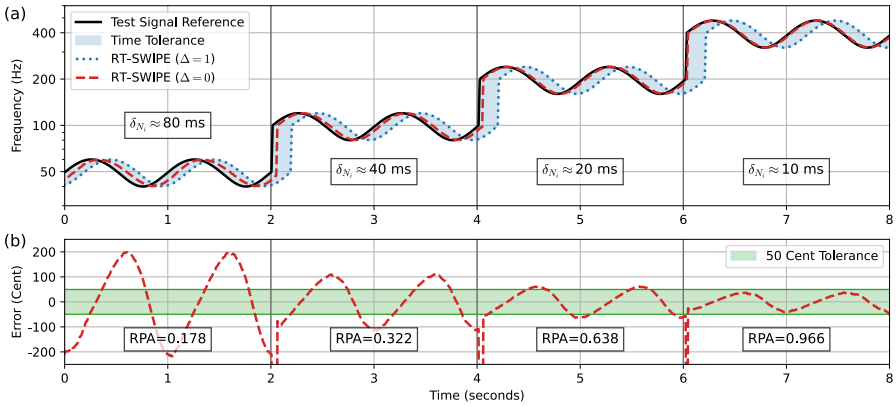

Fig. 5. (a) Reference annotation of a vibrato test signal (black line) with estimates for different delay factors Δ. **(b)** Error of `RT-SWIPE` ($\Delta = 0$) estimates compared to the reference in cents, with corresponding RPA values.

Figure 4b shows the results of the different approaches in a cross-talk scenario. Here, target instruments are mixed with randomly selected interfering instruments from a different voice, using a signal-to-noise ratio (SNR) of 5 dB. As observed in our previous study [12], `YIN` is most susceptible to interference from cross-talk. `CREPE` achieves the highest overall accuracy, while `SWIPE` offers a favorable balance between performance and computational efficiency. Notably, all approaches exhibit increased sensitivity to cross-talk in frequency regions above 200 Hz.

The observed deviations of `RT-SWIPE` arise from a mismatch between real-time estimation and the RPA metric, which assumes perfect temporal alignment between estimates and reference annotations. In real-time settings, however, causal windowing introduces algorithmic delays—causing accurate but slightly delayed estimates to be penalized as incorrect.

3.2 Results with Time Tolerance

In the following, we illustrate the algorithmic delays introduced through the causal windowing with a synthetic test signal. In Figure 5a, the reference f_0 trajectory of the test signal is shown as a black line. This trajectory features sinusoidal oscillations around a base frequency f_{base}, resembling a vibrato pattern. The modulation has a frequency of $f_{\text{mod}} = 1$ Hz and an amplitude of $A_{\text{mod}} = 0.2 \cdot f_{\text{base}}$. The test signal includes four segments, each two seconds long, where the base frequency f_{base} doubles sequentially from 50 Hz to 100 Hz, 200 Hz, and finally 400 Hz. We sonify this f_0 trajectory using the `libsoni` [14] toolbox with the function `sonify_f0`, utilizing 20 partials with individual amplitudes of 1/20, at a sampling rate f_s of 44.1 kHz to mimic a harmonic signal.

The test signal is analyzed with two delay factors $\Delta = 1$ (centered) and $\Delta = 0$ (right-aligned) ($f_{\min} = 30$ Hz, $f_{\max} = 700$ Hz, $f_s = 44.1$ kHz, $H = 512$ samples). The centered window positioning with a delay factor of $\Delta = 1$ is illustrated by the blue dotted line in Figure 5a. The estimated trajectory closely resembles the original reference trajectory, although there is a constant algorithmic delay of 186 ms. This delay stems from the window delay associated with the largest window ($N_{\max}$).

In contrast, right-aligned window positioning with $\Delta = 0$ is illustrated by the red dashed line in Figure 5a. Like the center-aligned configuration, this trajectory closely follows the reference but exhibits a frequency-dependent delay rather than a constant one. Specifically, the delay decreases with increasing pitch: at 50 Hz, the delay is approximately 80 ms; at 100 Hz, it drops to around 40 ms; at 200 Hz, to about 20 ms, and at 400 Hz, to roughly 10 ms.

In Figure 5b, the red line represents the estimation error in cents between the reference and `RT-SWIPE` with right-aligned window positioning ($\Delta = 0$). The green shaded area indicates a 50-cent tolerance used for evaluating the RPA. For the 400 Hz segment (seconds 6 to 8), the error curve remains well within the tolerance band, resulting in a high RPA of 0.966. At 200 Hz (seconds 4 to 6), the RPA drops to 0.638, and further declines to 0.322 for the 100 Hz segment (seconds 2 to 4). The lowest RPA of 0.178 is observed in the 50 Hz segment (seconds 0 to 2), where large portions of the error curve fall outside the tolerance range.

This example indicates that strict frame-based evaluation, like RPA, may not be ideal for evaluating real-time signals with delayed estimates. Similar to how cent tolerance considers pitch estimates within a certain pitch range as correct (see the green area in Figure 5b), managing delayed estimates also requires a time tolerance (see the blue area in Figure 5a).

To address this limitation, we propose extending the RPA metric by incorporating a time tolerance in addition to the usual cent-based pitch tolerance. For each reference frame, we evaluate not only the pitch estimate at the same time frame but also those in subsequent frames within a specified time window $\tau \in \mathbb{R}$ (given in ms). If any of these estimates fall within the allowed cent tolerance (e. g., ± 25 ms), we count the reference frame as correctly estimated. Since we are dealing with real-time systems, the tolerance window includes only future frames, consistent with the causal nature of the application.

Figure 6a and Figure 6b illustrate the effect of varying time tolerances on our evaluation, shown for scenarios without and with cross-talk, respectively. At $\tau = 0$ ms, the result matches the original frame-based evaluation from Figure 4. As τ increases, RPA values improve, especially at lower frequencies where algorithmic delays are more pronounced. With $\tau = 23.2$ ms, the RPA aligns more closely with the offline baseline (`SWIPE`). When τ is increased to 46.4 ms, the RPA results are nearly identical to the `SWIPE` baseline at lower frequencies and slightly higher at higher frequencies. This indicates that estimation accuracy degradations are a result of delayed estimates—a problem that is not taken into account by standard RPA measures.

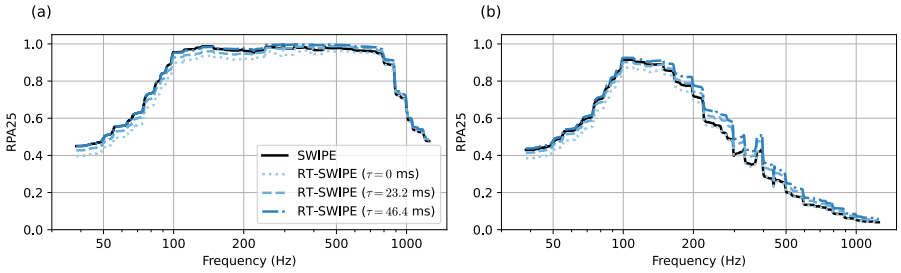

Fig. 6. RPA with a 25-cent tolerance over frequency on the `ChoraleBricks` dataset for `RT-SWIPE` at varying time tolerances τ, using `SWIPE` as a reference. **(a)** Instruments without cross-talk. **(b)** Instruments with cross-talk: target instrument signals are mixed with a randomly selected interfering instrument from a different voice, using a signal-to-noise ratio (SNR) of 5 dB.

4 Conclusions

In this paper, we present `RT-SWIPE`, a real-time adaptation of the `SWIPE` algorithm with adjustable window positioning to reduce pitch estimation delays. Our analysis shows that it supports real-time processing of up to 20 channels on consumer hardware. Experiments with synthetic signals highlight that while `RT-SWIPE` reduces algorithmic delay, frame-based metrics like RPA can misrepresent delayed yet accurate estimates. To address this, we propose a time-tolerant extension to RPA. Tests on real-world data confirm that `RT-SWIPE` preserves the accuracy of the original method while enabling real-time use with controlled delay characteristics.

In future work, we plan to use `RT-SWIPE` as the algorithmic backbone for a range of applications in ensemble music. One promising direction is the development of tools that provide musicians with objective intonation feedback to support and enhance their auditory skills. Beyond these application-oriented goals, we also see potential in further advancing the algorithm itself. In particular, `dSWIPE`, a differentiable extension of `SWIPE`, offers a compelling research avenue. By enabling the training of spectral templates specific to individual instruments, it may further improve robustness in acoustically challenging conditions, especially in scenarios with significant cross-talk between sources.

Acknowledgments. The International Audio Laboratories Erlangen, Germany, are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant numbers 500643750 (MU 2686/15-1) and 555525568 (MU 2686/18-1).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Balke, S., Berndt, A., Müller, M.: ChoraleBricks: A modular multitrack dataset for wind music research. Transaction of the International Society for Music Information Retrieval (TISMIR) **8**(1), 39–54 (2025). https://doi.org/10.5334/tismir.252
2. Bradford, R., Dobson, R., et al.: Sliding with a constant q. In: Proc. of the Int. Conf. on Digital Audio Effects (DAFx-08). pp. 363–369. DAFx (2008)
3. Camacho, A., Harris, J.G.: A sawtooth waveform inspired pitch estimator for speech and music. The Journal of the Acoustical Society of America **124**(3), 1638–1652 (2008)
4. Cannam, C., Landone, C., Sandler, M.B.: Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files. In: Proceedings of the International Conference on Multimedia. pp. 1467–1468. Florence, Italy (2010)
5. de Cheveigné, A., Kawahara, H.: YIN, a fundamental frequency estimator for speech and music. Journal of the Acoustical Society of America (JASA) **111**(4), 1917–1930 (2002)
6. Cont, A., Dubnov, S., Wessel, D.: Realtime multiple-pitch and multiple-instrument recognition for music signals using sparse non-negativity constraints. In: Proceedings of the International Conference on Digital Audio Effects. pp. 85–92. Bordeaux, France (2007)
7. Dai, J., Dixon, S.: Analysis of interactive intonation in unaccompanied SATB ensembles. In: Proceedings of the International Society for Music Information Retrieval Conference (ISMIR). pp. 599–605. Suzhou, China (2017)
8. Goto, M.: A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. Speech Communication (ISCA Journal) **43**(4), 311–329 (2004)
9. Kim, J.W., Salamon, J., Li, P., Bello, J.P.: CREPE: A convolutional representation for pitch estimation. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 161–165. Calgary, Canada (2018). https://doi.org/10.1109/ICASSP.2018.8461329
10. Mauch, M., Dixon, S.: pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 659–663. Florence, Italy (2014)
11. McFee, B., Raffel, C., Liang, D., Ellis, D.P., McVicar, M., Battenberg, E., Nieto, O.: Librosa: Audio and music signal analysis in Python. In: Proceedings the Python Science Conference. pp. 18–25. Austin, Texas, USA (2015). https://doi.org/10.25080/Majora-7b98e3ed-003
12. Meier, P., Müller, M., Balke, S.: Analyzing pitch estimation accuracy in cross-talk scenarios: A study with wind instruments. In: Proceedings of the Sound and Music Computing Conference (SMC). Graz, Austria (2025)
13. Meier, P., Schwär, S., Krump, G., Müller, M.: Evaluating real-time pitch estimation algorithms for creative music game interaction. In: INFORMATIK 2023 – Designing Futures: Zukünfte gestalten, pp. 873–882. Gesellschaft für Informatik e.V., Bonn, Germany (2023). https://doi.org/10.18420/inf2023_97
14. Özer, Y., Brütting, L., Schwär, S., Müller, M.: libsoni: A Python toolbox for sonifying music annotations and feature representations. Journal of Open Source Software (JOSS) **9**(96), 06524:1–6 (2024). https://doi.org/10.21105/joss.06524
15. Rosenzweig, S., Schwär, S., Müller, M.: libf0: A python library for fundamental frequency estimation. In: Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR). Bengaluru, India (2022)

16. Strahl, S., Müller, M.: dYIN and dSWIPE: Differentiable variants of classical fundamental frequency estimators. IEEE/ACM Transactions on Audio, Speech, and Language Processing (2025). https://doi.org/10.1109/TASLPRO.2025.3581119
17. Wang, J.H., Wang, S.A., Chen, W.C., Chang, K.N., Chen, H.Y.: Real-time pitch training system for violin learners. In: 2012 IEEE International Conference on Multimedia and Expo Workshops. pp. 163–168 (2012). https://doi.org/10.1109/ICMEW.2012.35