Burghardt, M. & Weiß, C. (Hrsg.): 4th Workshop InfDH,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 1

Evaluating Real-Time Pitch Estimation Algorithms for Creative Music Game Interaction

Peter Meier[1,2] Simon Schwär[2] Gerhard Krump[1] Meinard Müller[2]

Abstract: Music-based games are an important genre in the gaming community and have become increasingly popular with games like SingStar and Guitar Hero. These types of games are usually based on reactive game mechanics, where the player must hit a certain note at a certain time in order to score points. In this contribution, we present a game prototype that goes beyond purely music-reactive game mechanics and focuses more on the creative aspect of making music in games. In particular, we developed a jump-and-run game that can be controlled with a gaming controller but also uses the player's singing voice to interact with the game world. To this end, we estimate the pitch of a microphone signal in real time and use it as a creative input to the game. This input can be used to control parts of the game world, for instance by singing and adding stair-like elements that allow the player to overcome obstacles and reach the end of a game level. With our game prototype, we demonstrate how game designers can incorporate musical challenges into a well-known game environment while motivating musicians to creatively explore and practice their musical skills. Furthermore, motivated by our game prototype, we evaluate different real-time pitch estimation algorithms using common MIR metrics on a publicly available dataset to analyse what works best for our gaming scenario.

Keywords: Evaluation, Real-Time, Pitch, Estimation, Algorithm, Creative, Music, Game, Interaction

1 Creative Music Game Interaction

This paper is part of a series on "Real-Time Signal Processing Algorithms for Interactive Music Analysis Applications." In previous work, we presented a music-reactive game with beat tracking, where the game world is generated in real time from ambient music (e.g., radio, live music) of the player [Me22]. In this article, we consider a different type of music game that uses pitch estimation as a control input, see Fig. 1 for an illustration. In developing this game, we want to encourage the creative use of melody and singing for players and game designers and concentrate on exploring and teaching real-time aspects of pitch estimation. In particular, we focus on two pitch estimation algorithms, `YIN` [CK02] and `SWIPE` [CH08], which we adapted from the Python library libf0 [RSM22] and modified to work in real time for our gaming context. In this paper, we closely follow [Me23], where we already described the conceptual idea of our game prototype, and now complement it with experiments to evaluate the developed real-time pitch estimation algorithms.

[1] Deggendorf Institute of Technology, 94469 Deggendorf, Germany, peter.meier@th-deg.de, gerhard.krump@th-deg.de
[2] International Audio Laboratories Erlangen, 91058 Erlangen, Germany, peter.meier@audiolabs-erlangen.de, simon.schwaer@audiolabs-erlangen.de, meinard.mueller@audiolabs-erlangen.de

CC BY SA

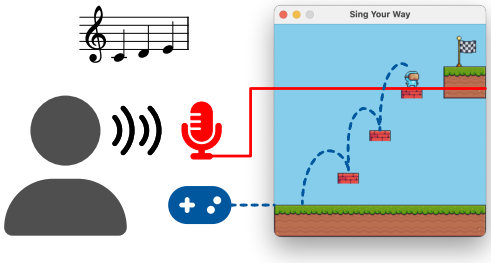

Fig. 1: Illustration of our “Sing Your Way” music game prototype. It uses a combination of traditional gaming controller and singing voice input to add note blocks to the game.

In our music game prototype called “Sing Your Way", the player can interact with the game world by using their singing voices. Fig. 2 shows what this might look like for a simple use case. In the first step, the player can sing a note. The pitch of this note is displayed as a red line in the game. The higher the note is sung, the higher the red pitch line is displayed. While singing a note, the player can interact with this red pitch line by jumping on it. By doing this, a fixed note block is created in the game for the player to stand on. The same step can be repeated several times at different pitches. As a result, players can add stair-like elements to the environment, using their voices to overcome obstacles, avoid enemies, or reach the end of a level.

One of the main goals of our game prototype is to create a platform that connects scientists, level designers, and musicians, especially in an educational context. For this reason, we have integrated the ability to design levels using a map editor[3] and a tileset[4]. This allows for creative level design without any technical background in programming and gives researchers new and creative ways to test real-time algorithms in the context of a music game. Furthermore, this platform could also be interesting for musicians to explore and practice their musical skills in a creative and entertaining setting.

2 Real-Time Pitch Estimation

In order to use the player’s singing voice as an input controller for the game, we need to estimate the pitch of the singing voice, which is a common task in Music Information Retrieval (MIR). The pitch estimation algorithm used for this task has to meet the following requirements to create a smooth and enjoyable gameplay experience. First, the algorithm must be computed in real time with low latency. Second, the pitch estimation must be accurate. Third, one requires a robust voicing detection that can distinguish between singing and non-singing, as we only want to display a red pitch line when the user is actually

[3] https://www.mapeditor.org
[4] https://pixelfrog-assets.itch.io/pixel-adventure-1

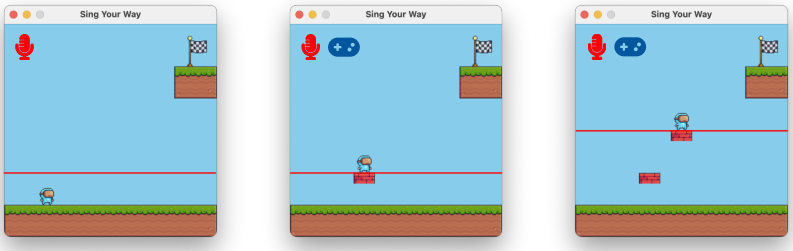

Fig. 2: Gameplay sequence showing how to add note blocks and stairs to the game: Sung notes appear as a red line that can be jumped on. This creates a fixed note block on which the player can stand.

singing[5]. In order to find suitable solutions for these requirements, we will first take a closer look at the general challenges of "Real-Time Music Information Retrieval" (Sect. 2.1) and then focus on two concrete examples of "Pitch Estimation Algorithms" (Sect. 2.2).

2.1 Real-Time Music Information Retrieval

Music Information Retrieval (MIR) is an area of research that focuses mainly on the analysis of large music datasets [ST22]. The algorithms used for these analyses are often not optimized for execution time and assume full availability of all data at all times. Compared to this *offline* processing, *online* algorithms used for real-time applications differ in three important aspects:

(a) **Causality of information**: For offline analysis, the entire audio signal is already available, and you can move forward or backward in the signal and select analysis windows of any size. For online analysis, only past and present information is available.

(b) **Trade-off between accuracy and latency**: In general, the more data is available, the more robust an analysis can be. For instance, a DFT with more audio samples results in a higher frequency resolution. However, for online analysis, this also has a direct impact on latency, as more audio samples also mean longer waiting times, especially for causal systems.

(c) **Signal processing with frames**: Online analysis is done in frames, processing a fixed number of samples as they are provided, for example, by a soundcard. There is usually a requirement to update the analysis result at the same frame rate, so the algorithm has limited time to complete before processing the next frame. As a result, online analysis methods may not be as complex as in the offline case and need to be optimized for execution time.

[5] General-purpose algorithms for voicing detection may not be able to differentiate between singing and speech since there is no clear boundary between the two. For our user interface, however, speech is not a suitable input as it typically does not allow for fine-grained control of pitch.

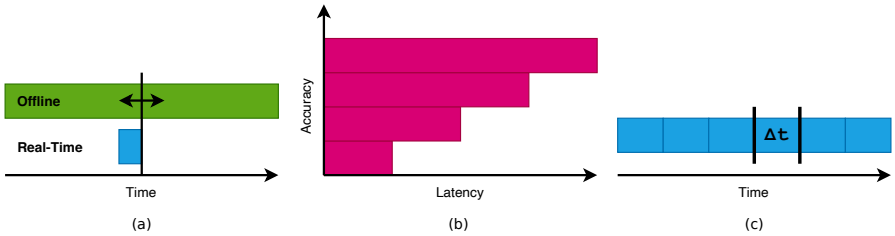

Fig. 3: Illustration of three main differences between offline and real-time Music Information Retrieval. (a) Causality of information. (b) Trade-off between accuracy and latency. (c) Signal processing with frames.

2.2 Pitch Estimation Algorithms

In addition to the general challenges of real-time analysis discussed in the previous section, we now want to take a closer look at the specific challenges of real-time analysis with respect to different pitch estimation algorithms. When we talk about pitch estimation in this article, we actually mean fundamental frequency estimation (F0 estimation), which is often used synonymously with pitch estimation. The basic idea of F0 estimation is to find the signal's fundamental frequency that correlates with the perceived pitch for each time position [CH08; CK02; SG12]. The input to the algorithm is the waveform, spectrogram, or other representation of the signal. The output of the algorithm is the frequency estimate of the input signal and an indication of how confident the algorithm is that the signal is tonal and periodic, called the "confidence value." This confidence value is often used in combination with a threshold to provide a form of voicing detection, for example, to distinguish singing from non-singing signals. Two methods that compute the values on a frame-by-frame basis, and are therefore in principle suitable for real-time applications, are shown in Fig. 4 and will be discussed in the following section.

One of the most well-known F0 estimation approaches is called `YIN` [CK02]. The `YIN` algorithm is based on an autocorrelation method and is calculated entirely in the time domain, using the signal's waveform as input, see also Fig. 4. The algorithm provides accurate pitch estimates for signals with singing, but also incorrectly detects pitches for non-singing parts of the signal. With respect to real-time applications, `YIN` is relatively simple and can be implemented in an efficient way with low latency. In fact, the offline algorithm can be directly transferred into an online variant, since only one input frame of audio data is required for each output frame of pitch estimation. In addition, `YIN` can be extended with post-processing steps [MD14], which we would like to investigate more closely for real-time applications in the future. For voicing detection, `YIN` calculates the so-called "aperiodicity" of the input signal. The lower the aperiodicity, the higher the probability that the signal is periodic and tonal. This is opposite to the confidence, where lower values indicate a lower probability that the signal is periodic and tonal. To allow a

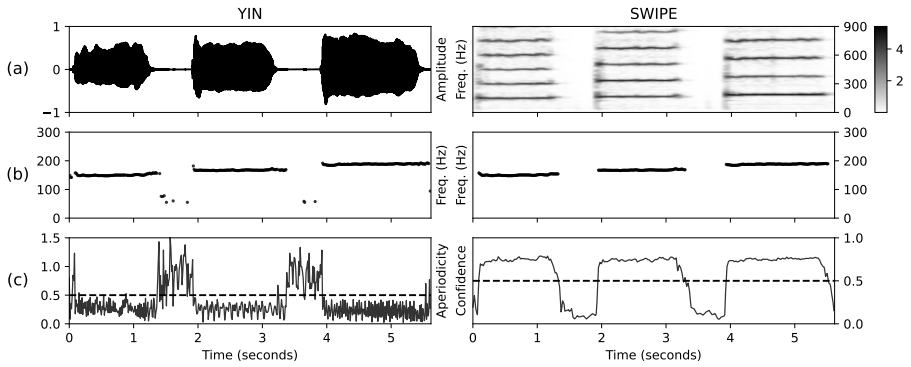

Fig. 4: An overview of two pitch estimation algorithms used for our music game prototype. `YIN`: (a) Waveform input. (b) Pitch estimation trajectory. (c) Aperiodicity for voicing detection. `SWIPE`: (a) Spectrogram input. (b) Pitch estimation trajectory. (c) Confidence for voicing detection.

more direct comparison with `SWIPE`, in this paper we define a confidence = 1 − aperiodicity for `YIN`, which will be used in the following.

Unlike the `YIN` time-domain algorithm, `SWIPE` is based on spectral correlations in the frequency domain [CH08]. To this end, `SWIPE` uses multiple pitch candidates with a unique fundamental frequency, each represented by a single pre-computed kernel inspired by a sawtooth waveform. The original offline version of the algorithm (`SWIPE-Off` in the following) transforms the input signal into a time–frequency representation using Short-Time Fourier Transforms (STFTs). Each pitch candidate requires a different optimal window size for the spectral input, which results in more accurate pitch estimates, but also in a higher computational complexity. Pitch estimates are calculated by correlating the spectrum of each input frame with the set of pitch kernels, finding the kernel with the highest correlation. The amount of correlation is also used as a confidence value for voicing detection. In the online version of `SWIPE`, we use a rolling buffer that stores the previous input frames needed to compute all the different window sizes for each frame. This concept allows for three different online variants of `SWIPE` (`SWIPE-On-A/B/C`), which are illustrated in Fig. 5 and described below:

`SWIPE-On-A`: The most obvious arrangement of windows follows the offline algorithm by keeping the spectra perfectly centered and aligned for each point in time. For a causal real-time system, this results in a delayed output signal because the rolling buffer must be filled completely before computation, see (A) of Fig. 5.

`SWIPE-On-B`: To compensate for this delay, the windows can be filled in such a way that the most recent frames are always in the center of each window, while keeping the alignment between the different window sizes as in `SWIPE-On-A`. Any samples in the future are set to zero. As illustrated in (B) of Fig. 5, this causes a reduction in the confidence values.

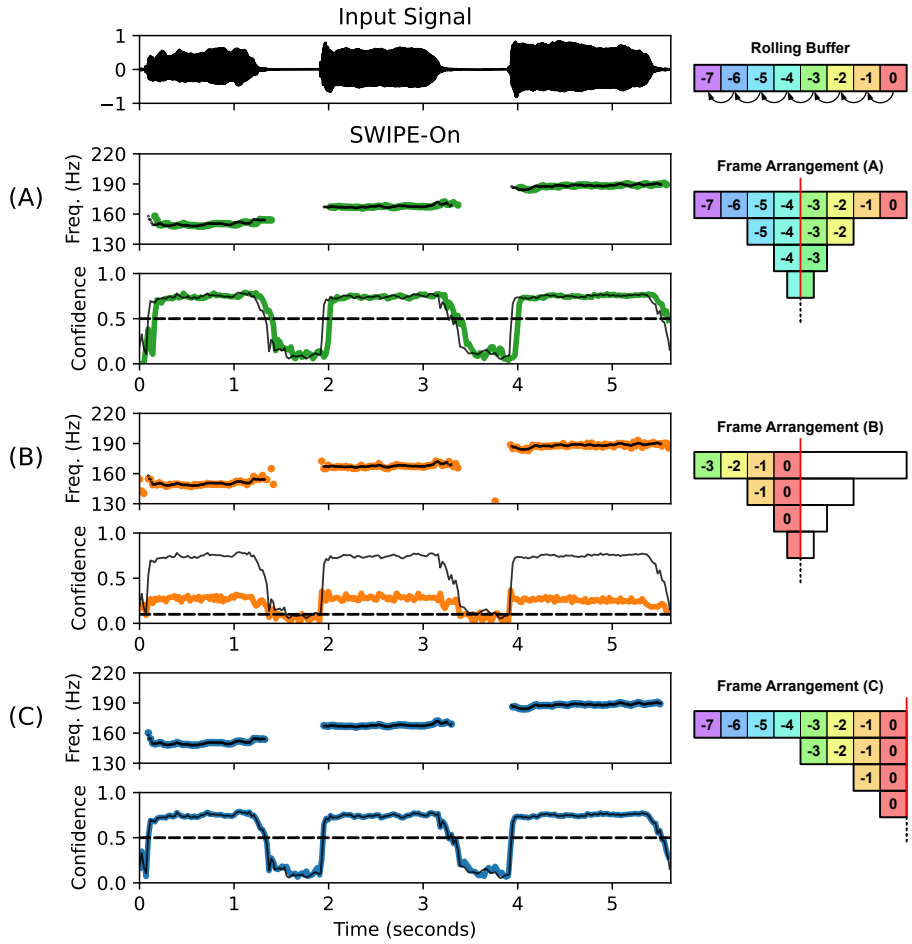

Fig. 5: **Left**: Frequency estimates and confidence values of the `SWIPE-On` variants (denoted by A, B, C, and colored in green, orange, blue). Values for `SWIPE-Off` are displayed in black. Confidence thresholds for each variant are indicated by black dashed lines. **Right**: The rolling buffer holds several input frames, symbolized by squares. The numbers indicate the frame index, where 0 is the current frame, -1 is the previous frame, and so on. The frame arrangements for each variant visualize the used signal segment for different window sizes. We only show four rows per frame arrangement, although there are usually much more windows sizes to calculate for `SWIPE`.

This is because only half-filled windows are available for the spectral correlation with the frequency kernels, lowering the score.

`SWIPE-On-C`: In the third and final online variant, all the different windows are filled with the most recent audio data. This discards a perfectly centered time alignment of frames between

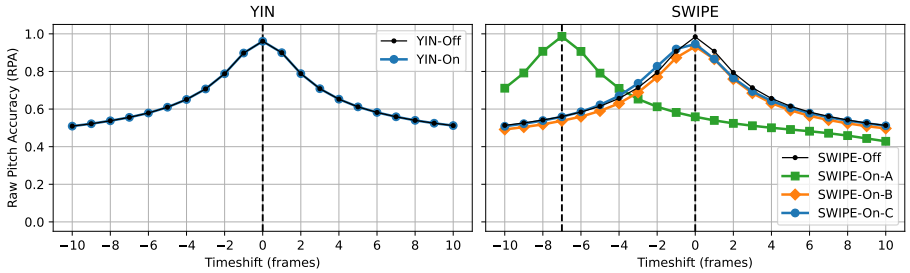

Fig. 6: Average raw pitch accuracy versus the timeshift (frames) of the estimated frequency values, tested with a pitch tolerance of 50 cents.

Metric	RPA		RCA	
Timeshift	-7	0	-7	0
`YIN-Off`	0.535±0.172	0.926±0.093	0.556±0.166	0.961±0.056
`YIN-On`	0.535±0.172	0.926±0.093	0.556±0.166	0.961±0.056
`SWIPE-Off`	0.554±0.171	0.976±0.092	0.560±0.166	0.984±0.070
`SWIPE-On-A`	0.978±0.091	0.553±0.171	0.986±0.069	0.559±0.166
`SWIPE-On-B`	0.521±0.168	0.909±0.116	0.538±0.161	0.933±0.089
`SWIPE-On-C`	0.556±0.170	0.938±0.101	0.559±0.169	0.946±0.087

Tab. 1: Average raw pitch/chroma accuracies and their standard deviations between different tracks of the dataset for different timeshifts (frames) of the estimated frequency values, tested with a pitch tolerance of 50 cents.

different window sizes and thus between F0 estimates at different frequencies. However, it avoids the problems of (`SWIPE-On-A`) delayed output and (`SWIPE-On-B`) reduced confidence, see (C) of Fig. 5.

3 Experiments

For an objective evaluation of the different algorithms, we use the MDB-melody-synth dataset [Sa18]. It consists of 65 different tracks with mainly singing voice melodies derived from the MedleyDB dataset [Bi14]. The MDB tracks were re-synthesized by the method described in [Sa17] to provide perfect ground truth annotations. Thus, MDB-melody-synth is well suited for evaluating real-time algorithms, as the annotations can be easily converted to a frame size of 512 samples and a sampling rate of 44100 Hz, which are also typical real-time audio settings that we use for our game.

In our experiments, we compare a total of six different pitch estimation algorithms: `YIN-Off/On`, `SWIPE-Off`, and `SWIPE-On-A/B/C`. To find a setting that is compatible with all the pitch estimation algorithms as well as the dataset, we run each algorithm with a

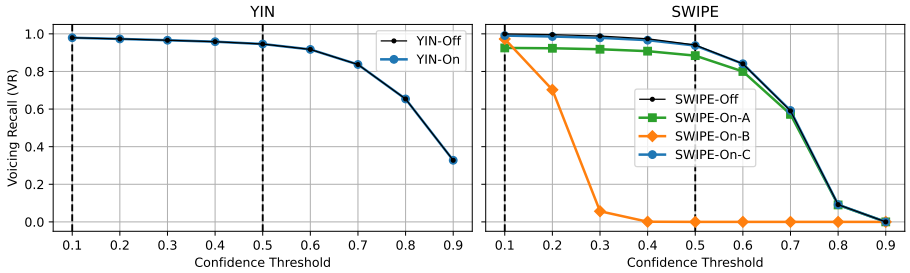

Fig. 7: Average voicing recall rates versus the confidence threshold.

Metric	VR		VFA	
Confidence	0.1	0.5	0.1	0.5
`YIN-Off`	0.979±0.020	0.946±0.070	0.017±0.021	0.013±0.018
`YIN-On`	0.979±0.020	0.946±0.070	0.017±0.021	0.013±0.018
`SWIPE-Off`	1.000±0.003	0.941±0.088	0.012±0.010	0.000±0.000
`SWIPE-On-A`	0.925±0.054	0.884±0.096	0.131±0.106	0.105±0.089
`SWIPE-On-B`	0.973±0.075	0.000±0.000	0.009±0.010	0.000±0.000
`SWIPE-On-C`	0.991±0.008	0.938±0.087	0.009±0.010	0.003±0.007

Tab. 2: Average voicing recall/false alarm rates and their standard deviations between different tracks of the dataset for different confidence thresholds.

sampling rate of 44100 Hz, a frame size of 512 samples, a minimum F0 estimate of 45 Hz, and a maximum F0 estimate of 1760 Hz. For the evaluation, we measure the quality of the pitch estimation in Raw Pitch Accuracy (RPA) and Raw Chroma Accuracy (RCA), both with 50 cents pitch tolerance. Furthermore, we measure the quality of the voicing detection in Voicing Recall (VR) and Voicing False Alarm (VFA), as described in [SG12]. For all evaluation metrics, we use the reference implementation of `mir_eval` [Ra14]. In addition, we run all evaluations for different timeshifts (from -10 to 10 frames) and confidence thresholds (from 0.1 to 0.9).

In Fig. 6 we show the Raw Pitch Accuracy (RPA) for estimated frequency values shifted in time between -10 and 10 frames. With Tab. 1 we report both RPA and RCA for timeshifts of both -7 and 0 frames. We see that `YIN-Off` and `YIN-On` produce exactly the same accuracy values. `SWIPE-Off` outperforms `YIN-Off` by five percentage points with regard to RPA. When shifted -7 frames, `SWIPE-On-A` achieves similar accuracy as `SWIPE-Off`. For a zero frames timeshift, `SWIPE-On-C` comes closest to `SWIPE-Off` with a drop of 0.038, followed by `SWIPE-On-B` with a drop of 0.067.

In Fig. 7 we show the Voicing Recall (VR) for different confidence thresholds, and with Tab. 2 we report the values for Voicing Recall (VR) and Voicing False Alarm (VFA). `YIN-Off` and `YIN-On` provide exactly the same Voicing Recall (VR). Furthermore, `SWIPE-Off`

has a perfect Voicing Recall (VR) of 1.0 compared to `YIN-Off` with a drop of 0.021, both with similar Voicing False Alarm (VFA) rates close to zero. `SWIPE-On-B` works only with lower confidence thresholds, where it achieves a similar Voicing Recall (VR) as the other algorithms. The best overall online variant is `SWIPE-On-C` with a Voicing Recall (VR) similar to `SWIPE-Off` and a Voicing False Alarm (VFA) rate close to zero.

4 Conclusion and Discussion

With our music game prototype, we provide a platform that uses real-time pitch estimation as an input to an interactive game. Our general aim is to bring together researchers, level designers, and musicians to explore algorithms, design interesting levels and practice musical skills in a creative context. In this paper, we compared and evaluated several real-time pitch estimation algorithms to be used in our interactive gaming environment, where both the accuracy of pitch estimation and voicing detection are equally important. We have shown that `YIN` transfers well from offline to online and runs very efficiently on the waveform directly, but does not reach the pitch estimation accuracy of `SWIPE`. Compared to `YIN`, `SWIPE` is computationally more complex and requires the additional step of performing multiple STFT operations per frame. Nevertheless, these spectrogram computations are efficient enough to not add noticeable latency to our Python-based test implementation. We presented three different variants for `SWIPE-On`, where `(SWIPE-On-A)` problems with delayed outputs can be solved with timeshifts, and `(SWIPE-On-B)` problems with reduced confidence values can be handled with lower thresholds. We also showed that `SWIPE-On-C` produces the highest overall accuracy for both pitch estimation quality and voicing detection, and is therefore the best option for our gaming scenario. For future research, we would like to look at other pitch estimation algorithms based on both classical engineering and more recent machine learning techniques and extend our case study to other datasets.

5 Acknowledgements

This work was supported by the BayWISS Joint Academic Partnership “Digitalisation.” The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

References

[Bi14] Bittner, R. M.; Salamon, J.; Tierney, M.; Mauch, M.; Cannam, C.; Bello, J. P.: MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In: Proceedings of the International Society for Music Information Retrieval Conference (ISMIR). Taipei, Taiwan, pp. 155–160, 2014.

[CH08] 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, pp. 1638–1652, 2008.

[CK02] de Cheveigné, A.; Kawahara, H.: YIN, a fundamental frequency estimator for speech and music. Journal of the Acoustical Society of America (JASA) 111/4, pp. 1917–1930, 2002.

[MD14] Mauch, M.; Dixon, S.: pYIN: A Fundamental Frequency Estimator Using Probabilistic Threshold Distributions. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Florence, Italy, pp. 659–663, 2014.

[Me22] Meier, P.; Schwär, S.; Rosenzweig, S.; Müller, M.: Real-Time MIR Algorithms for Music-Reactive Game World Generation. In: Mensch und Computer 2022 - Workshopband. Bonn, 2022.

[Me23] Meier, P.; Schwär, S.; Krump, G.; Müller, M.: Real-Time Pitch Estimation for Creative Music Game Interaction. In: Fortschritte der Akustik - DAGA 2023 Hamburg. Deutsche Gesellschaft für Akustik e.V. (DEGA), Berlin, pp. 1346–1349, 2023, ISBN: 978-3-939296-21-8.

[Ra14] Raffel, C.; McFee, B.; Humphrey, E. J.; Salamon, J.; Nieto, O.; Liang, D.; Ellis, D. P. W.: MIR_EVAL: A Transparent Implementation of Common MIR Metrics. In: Proceedings of the International Society for Music Information Retrieval Conference (ISMIR). Taipei, Taiwan, pp. 367–372, 2014.

[RSM22] Rosenzweig, S.; Schwär, S.; Müller, M.: libf0: A Python Library for Fundamental Frequency Estimation. In: Late Breaking Demos of the International Society for Music Information Retrieval Conference (ISMIR). Bengaluru, India, 2022.

[Sa17] Salamon, J.; Bittner, R. M.; Bonada, J.; Vicente, J. J. B.; Gómez, E.; Bello, J. P.: An Analysis/Synthesis Framework for Automatic F0 Annotation of Multitrack Datasets. In: Proceedings of International Society for Music Information Retrieval Conference (ISMIR). Suzhou, China, pp. 71–78, 2017.

[Sa18] Salamon, J.; Bittner, R.; Bonada, J.; Bosch, J. J.; Gómez, E.; Bello, J. P.: MDB-melody-synth, version 1.0.0, Zenodo, Nov. 2018, URL: https://doi.org/10.5281/zenodo.1481168.

[SG12] Salamon, J.; Gómez, E.: Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics. IEEE Transactions on Audio, Speech, and Language Processing 20/6, pp. 1759–1770, 2012.

[ST22] Stefani, D.; Turchet, L.: On the challenges of embedded real-time music information retrieval. en, Proceedings of the 25-th Int. Conf. on Digital Audio Effects (DAFx20in22) 3/, pp. 177–184, Sept. 2022.