Master thesis on Sound and Music Computing

Universitat Pompeu Fabra

# Singing Voice Melody Estimation From Polyphonic Signals

Logan Stillings

**Supervisor:** Pritish Chandna

August 2021

Master thesis on Sound and Music Computing

Universitat Pompeu Fabra

# Singing Voice Melody Estimation From Polyphonic Signals

Logan Stillings

**Supervisor:** Pritish Chandna

August 2021

# Contents

## Dedication

I would like to dedicate this work to all the amazing people I met in Barcelona.

## Acknowledgement

I would like to express my sincere gratitude to:

- Pritish Chandna
- My family
- My friends
- My fellow students
- My teachers

# Abstract

Singing voice melody estimation is the task of calculating the fundamental frequency (f0) of the predominant voice in a piece of music containing multiple instruments. In this work, I evaluate the performance of several popular f0 estimation algorithms using an annotated dataset (MedleyDB) of raw monophonic tracks, polyphonic mixes, and source-separated vocals. Many of the models were created to estimate the predominant melody and not necessarily the sung vocal melody, for example they could be capable of estimating the melody in instrumental music. Of the models tested, CREPE performs highly as a monophonic model, and Deep Salience and Encoder/Decoder perform highly as polyphonic models. By implementing source-separation as a preprocessing step, monophonic models such as CREPE, SPICE, and become viable options for the task of vocal melody estimation. These monophonic algorithms each perform significantly better in pitch accuracy on the source-separated vocal tracks compared to the polyphonic mixes. Additionally, each of the polyphonic algorithms tested increased in overall accuracy when using the source-separated tracks instead of the polyphonic mixes. I suggest further research implementing source-separation as a preprocessing step to vocal melody estimation using other source-separation tools and different datasets. Potential datasets could include tracks with overlapping vocal harmonies as well as different musical styles such as metal, rap, or non-western music.



Keywords: Melody; Singing Voice; Frequency Estimation

# Chapter 1

# Introduction

Estimating the melody from a sound mixture is a heavily researched topic in the field of sound and music computing. To define the task of melody estimation from polyphonic signals, one must first define the term "melody". While the concept of melody can be quite subjective, one common definition [1] states that "the melody is the single (monophonic) pitch sequence that a listener might reproduce if asked to whistle or hum a piece of polyphonic music, and that a listener would recognize as being the essence of that music when heard in comparison." This definition fails to account for the fact that different listeners may reproduce different instruments when humming the same song, however, in practice, the melody is a single audio source that pertains to the main instrument or voice in the piece of music. Estimation involves retrieving the fundamental frequency (f0) of the melody. The f0 of a signal refers to the physical property most closely related to the perceptual property of pitch [2], however for this work, the terms pitch and f0 will be used interchangeably. Applications of melody estimation include transcription, removing effects from a vocal stem, synthesizing a single voice from unison, converting growling vocals to a clean voice, vocal effects, pitch correction, and remixing. Ideally, melody estimation could help audio engineers and singers alike, as well as improve the current research on singing voice extraction. In this work, I examine the accuracy of current state-of-the-art melodic estimation methods for the task of vocal f0 estimation from

polyphonic mixtures.

## 1.1 Melodic Estimation From Monophonic Signals

Melodic estimation methods are better understood when first examining methods of monophonic f0 estimation. Monophonic signals only contain a single musical note playing at any given time. One of the oldest and most intuitive methods of monophonic f0 estimation is auto-correlation. The auto-correlation of a discrete signal can be defined as:

$$r_t(\tau) = \sum_{j=t+1}^{t+W} x_j x_{j+\tau} \tag{1.1}$$

where $r_t(\tau)$ is the auto-correlation function of lag $\tau$ calculated at time index $t$, and $W$ is the integration window size. When given a periodic signal, the auto-correlation function contains peaks at multiples of the period. In practice, the auto-correlation method chooses the highest non-zero-lag peak by searching within a range of lags [3]. This method is somewhat rudimentary and is prone to several types of errors. If the lower limit is too close to zero, the algorithm may erroneously choose the zero-lag peak. Conversely, if the higher limit is large enough, it may erroneously choose a higher-order peak. One of the first methods proposed to combat these types of errors was by using auto-correlation in the spectral domain [4], as opposed to the time domain. In addition to this technique of using auto-correlation in the spectral domain, applying spectral conditioning operations such as clipping and flattening the spectrum help eliminate noise and emphasize lower amplitude harmonics. This method achieved far fewer octave errors (when an algorithm selects a fundamental frequency value that is an exact multiple of the ground truth value) than the original auto-correlation function and was viewed as a significant improvement [4]. The YIN algorithm [3] was proposed to address the errors commonly found with auto-correlation by implementing a cumulative mean normalized difference function. YIN was not as prone to amplitude changes and had the added benefit of not requiring an upper frequency bound for calculations. Although YIN provided a significantly higher accuracy of f0 estimation compared to auto-correlation, it outlines a smoothing procedure that does not use the frame-wise estimate, leaving some room

for improvement. Probabilistic Yin (pYIN) [5] takes YIN and modifies its frame-wise variant using a probabilistic distribution to output multiple pitch candidates with their associated probabilities, while also reducing the loss of useful information before smoothing. pYIN uses Hidden Markov Models to output a monophonic pitch track of the most likely pitch candidates over time for monophonic signals. For many years, pYIN was considered the state-of-the-art in monophonic f0 estimation. In recent years, many cutting-edge monophonic estimation systems have implemented deep learning algorithms. Algorithms such as CREPE [6] and SPICE [7] are data-driven approaches, meaning that they learn from large volumes of data and are different from knowledge-based approaches (like auto-correlation), which use domain-specific information to influence processes. Both CREPE and SPICE show significant improvements in f0 estimation accuracy for monophonic signals when compared to pYIN.

## 1.2 Melodic Estimation From Polyphonic Signals

Polyphony refers to a type of music that contains more than one note playing simultaneously. With this definition in mind, one can state that melody estimation is the task of calculating the fundamental frequency of the predominant instrument or voice within a piece of music that has multiple sources playing simultaneously. It is also important to note that the melody in consideration should be the predominant melody or the source that human listeners could easily agree on being the main melody, regardless of the musical genre of the piece. The main challenges of estimating the melody in a polyphonic mixture are detecting peak frequencies of the noisy audio signal, and determining which pitches belong to the melody versus accompaniment. In polyphonic music, the audio signal is a superimposed representation of multiple instruments playing simultaneously. In the frequency domain, this makes it very difficult to determine which peak frequencies belong to which instrument. Additionally, common audio post-processing techniques (such as reverb, delay, and compression) blur the note onsets and make it increasingly difficult to detect peak frequencies. Algorithms also need to be able to determine when the melody is

present and when it is not (otherwise known as voicing detection). Some algorithms can identify vocal melodies more accurately by exploiting major differences between the singing voice and other musical instruments, however wrong pitch estimates and voicing false alarms do still occur when multiple harmonic sources are present in the music. Another common error that occurs in vocal melody estimation is the octave error. This can happen for low f0s because there is an increased salience at double the true f0 (due to a singer's resonance) and because tracking algorithms commonly bias against low frequencies [8]. Even after obtaining a pitch-based representation of the signal, one must determine which pitches relate to the main melody. Melodia [9] introduces a knowledge-based system that characterizes pitch contours using signal processing steps and a salience function specifically designed for the task of melody extraction. Using the trajectory of calculated pitch contours, the system successfully introduces features that tend to indicate melodies, such as the standard deviation of the contour pitch and the presence of vibrato. These contour features enable voicing detection, octave error minimization, and melody selection. More recently, several data-driven approaches have been used for melodic estimation. A U-Net [10] based model was proposed in 2019 that uses a neural network originally designed for medical imaging segmentation to estimate the dominant melody in polyphonic music. Deep Salience [11] and ResNet [12] both use convolutional neural networks for melody extraction from polyphonic music.

For this research, I will focus on singing voice melodic estimation from polyphonic signals specific to contemporary popular music. Although the singing melody may not necessarily be the dominant melody in the mixture, it is the target to be estimated for this task. The presence of harmonic instruments that are commonly found in contemporary popular music, such as guitars and synths, may lead to the supersession of the voice as the dominant melody. Figure 1 contains an image of a spectrogram that exemplifies an overlap between the harmonics of the instruments and the harmonics of the singing voice in the song Phantom by Big Troubles. To address the challenge of overlapping harmonics, models such as HRNet [13] have implemented a two-stage vocal melody extraction system that successfully separates vocals from an audio mixture and uses an encoder-decoder network to predict the

vocal f0 values. In this work, I will be comparing the results of common melody estimation models using a dataset of monophonic, polyphonic, and source-separated tracks for the task of vocal melody estimation. Some of the algorithms that I use were originally written to perform f0 estimation of predominant melody and not necessarily the vocal melody, hence the need to evaluate their performance for this task.

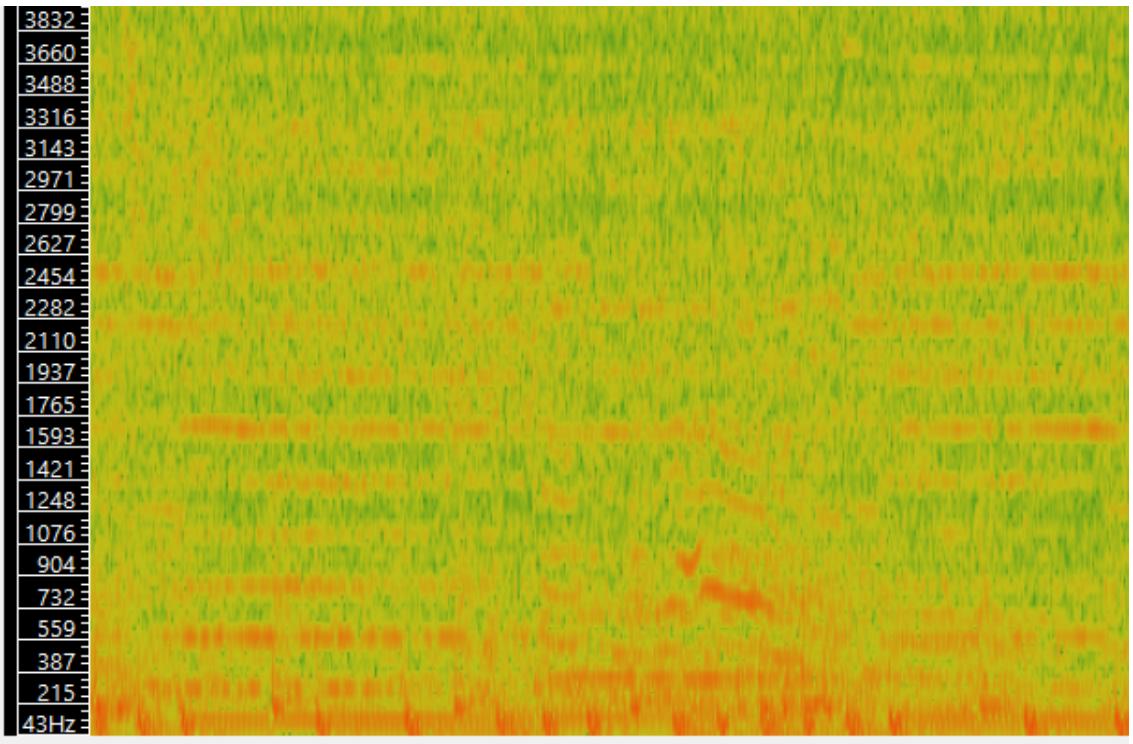


Figure 1: Big Troubles Phantom Mix Spectrogram. The spectrogram spans from the 1:10 mark to 1:15. This section of the song has many instruments including vocals, drums, bass, and guitar. Here the harmonics of the vocals overlap with the harmonics of the other instruments present in the mix.

### 1.2.1 Source Separation

The goal of source separation is to recover individual instruments from a mixed signal that has been mixed as part of a final song [14]. There are typically a wide variety of tones and timbres playing in a coordinated way, which adds to the difficulty of the task. Previous models attempted to predict soft masks over mixture spectrograms, however recent progress has been made using waveform-based inputs with recurrent neural networks. With spectrogram-based models, the mask applied

to the spectrogram has no reason to be real or obtained from a real signal, which could result in artifacts caused by a projection step in the inverse Short Time Fourier Transform process. Demucs [14] is a Convolutional Recurrent model which uses direct waveform training as opposed to spectrogram mixture. Open-Unmix [15] is a deep neural network reference implementation for music source separation based on a bi-directional Long Short Term Memory model. In this work, I will be using Open-Unmix to perform source separation on a dataset of polyphonic music to extract the vocal melody.

## 1.3 Background

### 1.3.1 Algorithms

#### CREPE

CREPE is a monophonic pitch tracker based on a deep convolutional neural network operating directly on the time-domain waveform input. It is state-of-the-art (as of 2018), outperforming popular pitch trackers such as pYIN and SWIPE.

#### Deep Salience

Deep Salience is a neural network architecture for learning pitch salience representations. Given a time-freq representation, the model learns the set of convolutional filters needed to create a salience representation with the same shape and frequency. The input of this model is the signal's harmonic constant-Q transform (HCQT). The HCQT is a 3-dimensional array measuring $h$-th harmonic of freq $f$ at time $t$. Combining bin ($k$) and harmonic ($h$) in the calculation allows an adaptable model which can find locality in time, freq, and harmonics.

#### Encoder-Decoder Model with SegNet

This algorithm is a streamlined encoder/decoder architecture that is designed for melody extraction [16]. It employs only 7 convolution or up-convolution layers.

### Melodia

Melodia is a polyphonic pitch tracker that extracts the fundamental frequency of the predominant melody from a signal. It characterizes pitch contours using signal processing steps and a salience function specifically designed for the task of melody extraction.

### pYIN

pYIN is a modification of the YIN algorithm for monophonic fundamental frequency estimation. In the first step of pYIN, f0 candidates and their probabilities are computed using the YIN algorithm. In the second step, Viterbi decoding is used to estimate the most likely f0 sequence and voicing flags.

### SPICE

SPICE is a self-supervised monophonic pitch estimator. When the audio signal is analyzed through the lens of the constant-Q transform (CQT), pitch shift maps to a simple translation. SPICE implements a convolutional encoder such that two shifted CQT inputs result in differing outputs proportional to the corresponding difference in pitch.

### U-Net

U-Net is a downsampling-upsampling model which was used originally for image segmentation. The model has been applied for the task of melody estimation in polyphonic music. The applied model uses a sequential method to train the U-Net with ground truth data at increasing resolutions. It also contains skip-connections from the encoding levels to their counterpart decoding levels, which provide an informative context to the next reconstruction level, and have proven to help localize features of interest.

### 1.3.2 Evaluation Metrics

There are several core evaluation metrics for measuring the accuracy of f0 estimation. The Raw Pitch Accuracy (RPA) measures the percentage of melody frames in which the estimated pitch is within a half semitone of a reference [17]. The Raw Chroma Accuracy (RCA) also measures pitch accuracy but estimated and reference frequencies are mapped to a single octave, therefore this metric ignores common octave errors. Voicing Recall (VR) measures the percentage of frames labeled as voiced in the reference that are also labeled as voiced by the algorithm. Voicing False Alarm (VFA) measures the percentage of frames labeled as unvoiced in the reference that are mistakenly estimated as voiced by the algorithm. Overall Accuracy (OA) measures the percentage of frames that were labeled correctly in terms of pitch and voicing. These metrics are commonly used across most f0 estimation systems.

### 1.3.3 Dataset

To evaluate melodic estimation systems, a good dataset with ground truth annotations is needed. MedleyDB [18] is a dataset of annotated, royalty-free multitrack recordings, which was curated specifically for the task of melody extraction. The dataset is made up of 108 multi-tracks with individual WAV files for the mix, processed stems, raw audio. The tracks are from a wide mix of genres such as Singer/-Songwriter, Classical, Rock, World/Folk, Fusion, Jazz, Pop, Musical Theater, Rap. In addition to the audio files, each song is provided with metadata information as well as ground truth annotations for the fundamental frequency of the melody. The annotations are stored in a CSV containing timestamp and f0 pairs. This dataset was chosen because of its access to annotations and raw vocal stems. In this work, I use a subset of this dataset containing only the 56 tracks that have sung vocals. Further, I will evaluate each algorithm on the raw audio stems, the polyphonic mix track, and a source-separated version of the mix tracks.

# Chapter 2

# Methods

In this work, I evaluate the performance of several f0 estimation algorithms for the specific task of singing voice melody estimation from polyphonic music. Algorithms that were specifically designed for monophonic f0 estimation will be evaluated to achieve a baseline accuracy, which can be compared with the accuracies of polyphonic estimation tools. Of the algorithms previously mentioned in chapter 1, only those that are available for use as python packages or through open-source repositories are used. Predictions are made for each algorithm using the subset of tracks in the MedleyDB dataset as mentioned in section 1.3.3. Each algorithm will only use the raw tracks, polyphonic mix tracks, and source-separated tracks from the dataset that contain vocals.

## 2.1 Source Separation

### 2.1.1 Open-Unmix (umx)

Open-Unmix is available as a python package and can be implemented from the command line. The umx command outputs separated stems from the original mix of vocals, bass, drums, and other instruments. I perform source separation on each of the polyphonic mix tracks from the MedleyDB subset to have a variety of tracks to test as well as to test the validity of source separation as a pre-processing step to

vocal melody estimation.

## 2.2 Estimation

### 2.2.1 pYIN

Using librosa [19], which is a python package for music and audio analysis, I obtained predictions for each of the raw vocal tracks from the MedleyDB subset. The minimum and maximum frequency parameters used in the algorithm are 65.41 Hertz (Hz) and 2093 Hz respectively (corresponding to the range between musical notes C2 and C7). The librosa method of the pYIN algorithm returns unvoiced frames as NaN (Not a Number), therefore some post-processing was done to convert the unvoiced frames to zero to have consistency in the evaluations across all algorithms. The predictions are saved in a text file as a timestamp-frequency pair with time in the leftmost column and frequency in the rightmost column, separated by a space.

### 2.2.2 CREPE

Similar to pYIN, predictions were obtained on each of the tracks from the MedleyDB subset. The implementation of CREPE is available as a python package. The output of CREPE's "predict" function also contains NaN for non-voiced frames, so those values are converted to zeros. Additionally, the "predict" function returns a confidence estimation along with the frequency estimation. Any frames with a confidence value of less than 50% (0.5) are regarded as unvoiced. The predictions are also saved in a text file as a timestamp-frequency pair with time in the leftmost column and frequency in the rightmost column, separated by a space.

### 2.2.3 Melodia

Predictions are obtained on each of the polyphonic mix tracks in the MedleyDB subset. The implementation of Melodia is available through the Essentia library, which is an open-source python package. A hop size of 128 samples is used for the estimator. The predictions are also saved in the same format as mentioned in the

two previous algorithms.

### 2.2.4 Deep Salience

The predictions were obtained by cloning the original repository that accompanied the paper and by implementing a script that was made available to predict Deep Salience's output from audio. The script allows for several parameters to affect the output. The non-default parameters implemented were the vocal task to perform estimations on vocal melodies, and the "singlef0" output to save a CSV of single f0 values. The output is saved as a timestamp-frequency pair where the leftmost column is time and the rightmost column is frequency. The output values are separated by a tab, and unvoiced timestamps have negative frequency values.

### 2.2.5 SPICE

SPICE was implemented by loading the pre-trained TensorFlow model. The model expects songs with a sampling rate of 16kHz and to contain a single channel (mono), therefore each track was processed to meet these requirements before using the track as input to the model. The model outputs uncertainties as a list of values in the interval [0, 1], corresponding to the rate of certainty of the model in correctly estimating the pitch. The model also outputs the pitches as a list of values in the interval [0, 1], which requires conversion to frequency values in Hertz. Pitch estimates with confidence below 90% are ignored. The timestamps are calculated separately using a hop size of 512 samples as dictated by the pre-trained model.

### 2.2.6 U-Net

The estimations for the U-Net model were obtained by cloning the original repository and by implementing a provided script. For each audio file, the script computes its Harmonic Constant Q Transform (HCQT) and estimates its dominant melody. This representation is trimmed along the frequency axis to keep only 3 octaves around the mean pitch. The final resolution is one bin per semitone and each time frame is 58 ms. Note that this implementation is different from each of the other algorithms

and will require a separate step in the evaluation phase to convert the output of the model to frequency values.[1]

### 2.2.7 Encoder-Decoder Model with SegNet

The predictions were obtained by cloning the repository that accompanied the original paper and by implementing an available python script to calculate the estimated f0. The script was run on each of the polyphonic mix tracks in the MedleyDB subset. The vocal model was chosen for this task. The output is saved in a text file containing the timestamp-frequency pairs separated by a space.

## 2.3 Evaluation

After obtaining the predicted f0 for each of the algorithms on the monophonic, polyphonic, and source-separated tracks, I then perform an evaluation to test the accuracy of the predictions. Evaluations are done by comparing the ground truth f0 in the annotations from MedleyDB with the estimated f0 as output by each of the algorithms. For the monophonic tracks, additional parsing is done to ensure that the raw vocal track corresponds to the vocal stem that matches the predominant vocal melody according to a ranking system included in the MedleyDB annotations. Raw vocal tracks that are not part of the stem of the predominant vocal melody are disregarded for the sake of this evaluation. Stems are ranked and annotated according to how likely they are to be the predominant melody. The ground truth is taken as the annotation labeled as Melody1 in MedleyDB.[2] The correct ground truth annotation and the estimated f0 are then evaluated using mir_eval [20]. mir_eval is a Python library for evaluating Music Information Retrieval systems. More specifically, I use mir_eval's "melody.evaluate" method, which compares two melody transcriptions,

[1] The output of the U-Net model requires some additional handling compared to the other algorithms. U-Net outputs a NumPy array with a shape of (T, 360) where T corresponds to the Time of the signal and 360 corresponds to the frequency bins within 3 octaves of the mean pitch. I use the argmax function to map the results from frequency bins into cents, which are then converted to Hz. A confidence threshold of 0.3 is chosen to determine voiced and unvoiced frames.

[2] According to the documentation from MedleyDB, the annotation labeled as Melody1 corresponds to using the following interpretation of melody: "The f0 curve of the predominant melodic line drawn from a single source". Therefore, this implementation is well suited for tracks where there is a single predominant melodic source.

where the first is treated as the reference and the second as the estimate to be evaluated. The method returns scores for the estimation compared to the ground truth using the five evaluation metrics mentioned in section 1.3.2. I calculate the average evaluation of all the songs as well as the standard deviation in order to easily compare each algorithm's performance and variance.

# Chapter 3

# Results

## 3.1 Monophonic Test Set

This section contains the results of each model being tested on the raw monophonic vocal tracks from MedleyDB. [3]

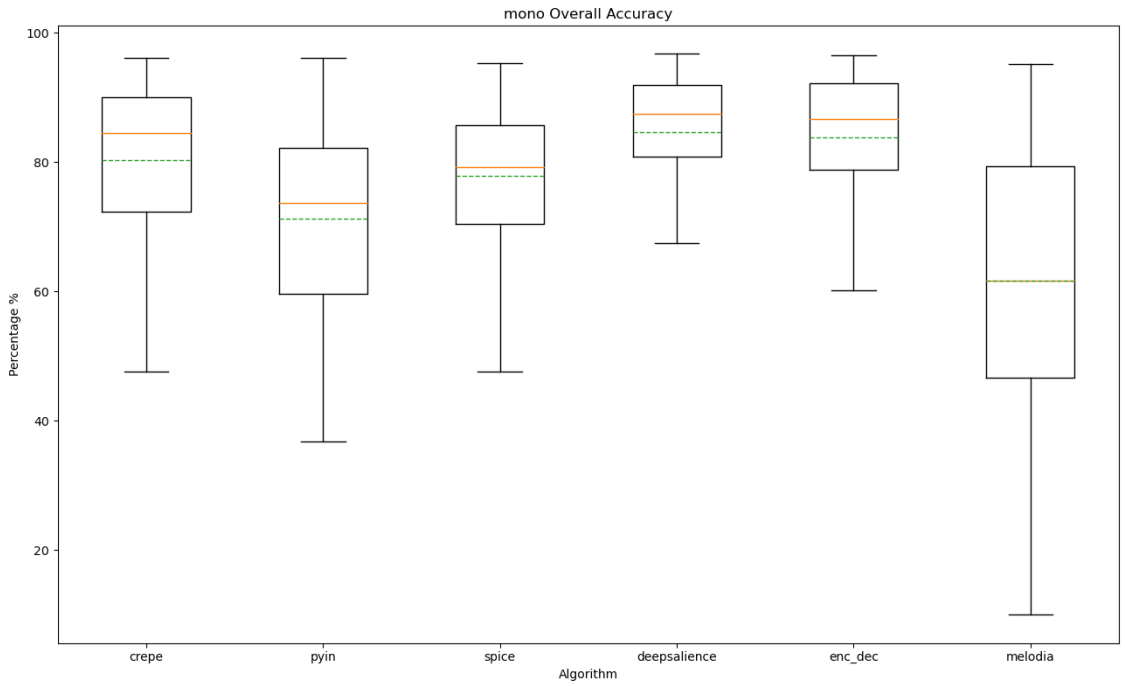


Figure 2: Mono Overall Accuracy[4]

[3]Due to unforeseen errors in converting the output of the U-Net model to frequency values, the evaluations of U-Net have been omitted from the results.

Table 1: Mono Overall Accuracy

| **Algorithm** | **Type**[5] | **Accuracy Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 80.31 | 12.04 |
| pYIN | Monophonic | 71.21 | 14.22 |
| SPICE | Monophonic | 77.88 | 10.96 |
| Deep Salience | Polyphonic | 84.67 | 10.11 |
| Encoder/Decoder | Polyphonic | 83.87 | 11.65 |
| Melodia | Polyphonic | 61.64 | 21.31 |

The Overall Accuracies can be seen in Table 1 and Figure 2. CREPE holds the strongest Overall Accuracy for the monophonic algorithms however, both the Encoder/Decoder model and Deep Salience have higher accuracy scores for the polyphonic algorithms.

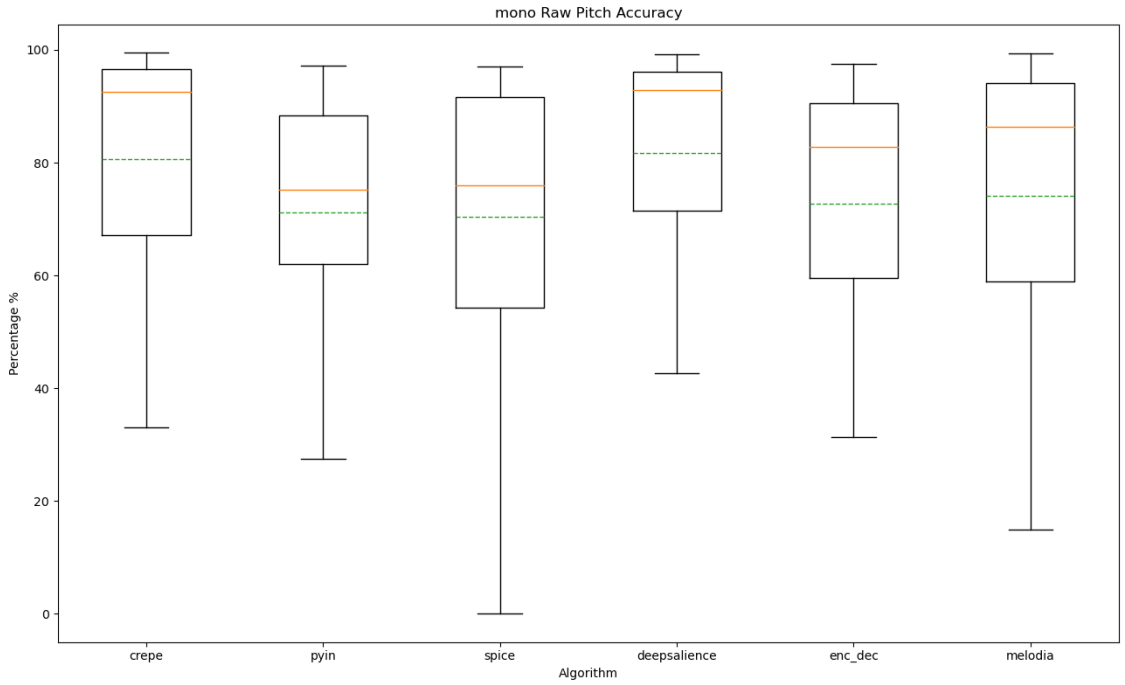


Figure 3: Mono Raw Pitch Accuracy

[4]Figure 2 and each of the following Figures pertaining to accuracy metrics use the same format of box plots. In each figure, the dashed green line represents the average, while the solid orange line represents the median. The top of each box represents Quartile 3, the median of the upper dataset when dividing the total dataset into two parts via the median. The bottom of each box represents Quartile 1, the median of the lower dataset when dividing the total dataset into two parts via the median. The boxed area represents the interquartile range or the range between Quartile 3 and Quartile 1. The whiskers mark the range of the non-outlier data. The implemented definition of non-outlier is [Q1 - 1.5xIQR, Q3 + 1.5xIQR].

[5]For Table 1 and each of the following tables, Type refers to the primary target audio type that the algorithm was originally trained to handle.

Table 2: Mono Raw Pitch Accuracy

| **Algorithm** | **Type** | **Accuracy Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 80.61 | 23.16 |
| pYIN | Monophonic | 71.13 | 21.64 |
| SPICE | Monophonic | 70.46 | 23.42 |
| Deep Salience | Polyphonic | 81.79 | 22.45 |
| Encoder/Decoder | Polyphonic | 72.66 | 24.10 |
| Melodia | Polyphonic | 74.16 | 25.77 |

The Raw Pitch Accuracies can be seen in Figure 3 and Table 2. The highest Pitch Accuracy of monophonic algorithms is CREPE and Deep Salience for polyphonic algorithms.

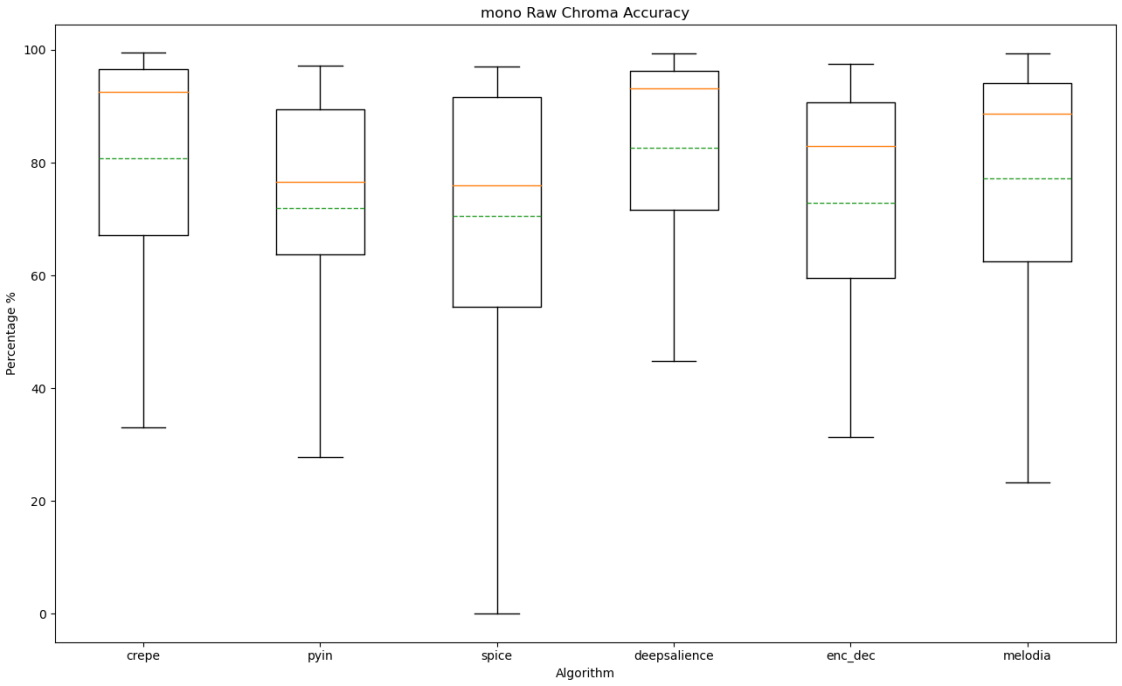


Figure 4: Mono Raw Chroma Accuracy

Table 3: Mono Raw Chroma Accuracy

| **Algorithm** | **Type** | **Accuracy Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 80.85 | 23.17 |
| pYIN | Monophonic | 71.99 | 21.62 |
| SPICE | Monophonic | 70.51 | 23.39 |
| Deep Salience | Polyphonic | 82.62 | 20.88 |
| Encoder/Decoder | Polyphonic | 72.90 | 24.16 |
| Melodia | Polyphonic | 77.29 | 23.17 |

The results for Raw Chroma Accuracy can be seen in Figure 4 and Table 3. The

results are quite similar to that of the Raw Pitch Accuracies, with no significant differences between the two metrics across the different algorithms. This is rather intuitive given that the Raw Chroma Accuracy is implemented as a version of pitch accuracy that ignores octave errors, therefore the Raw Chroma Accuracy for each algorithm will be slightly better than the Raw Pitch Accuracy.

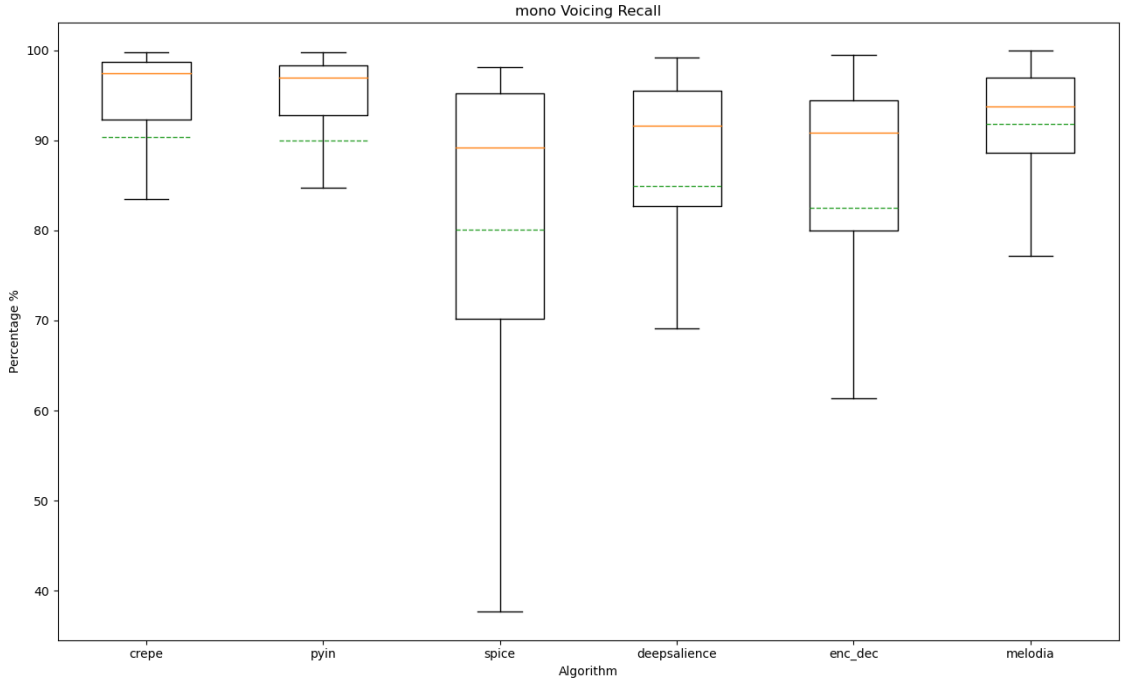


Figure 5: Mono Voicing Recall

Table 4: Mono Voicing Recall

| **Algorithm** | **Type** | **Rate Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 90.32 | 19.32 |
| pYIN | Monophonic | 89.95 | 19.56 |
| SPICE | Monophonic | 80.10 | 22.23 |
| Deep Salience | Polyphonic | 84.90 | 19.12 |
| Encoder/Decoder | Polyphonic | 82.47 | 21.02 |
| Melodia | Polyphonic | 91.85 | 7.16 |

The Voicing Recall rates can be seen in Table 4 and Figure 5. Both CREPE and pYIN achieve an accuracy of roughly 90% for the Monophonic models. Melodia is the strongest of the Polyphonic models.

The Voicing False Alarm rates can be seen in Figure 6 and Table 5. It should be noted that models scoring lower VFAs are seen as better-performing since they are

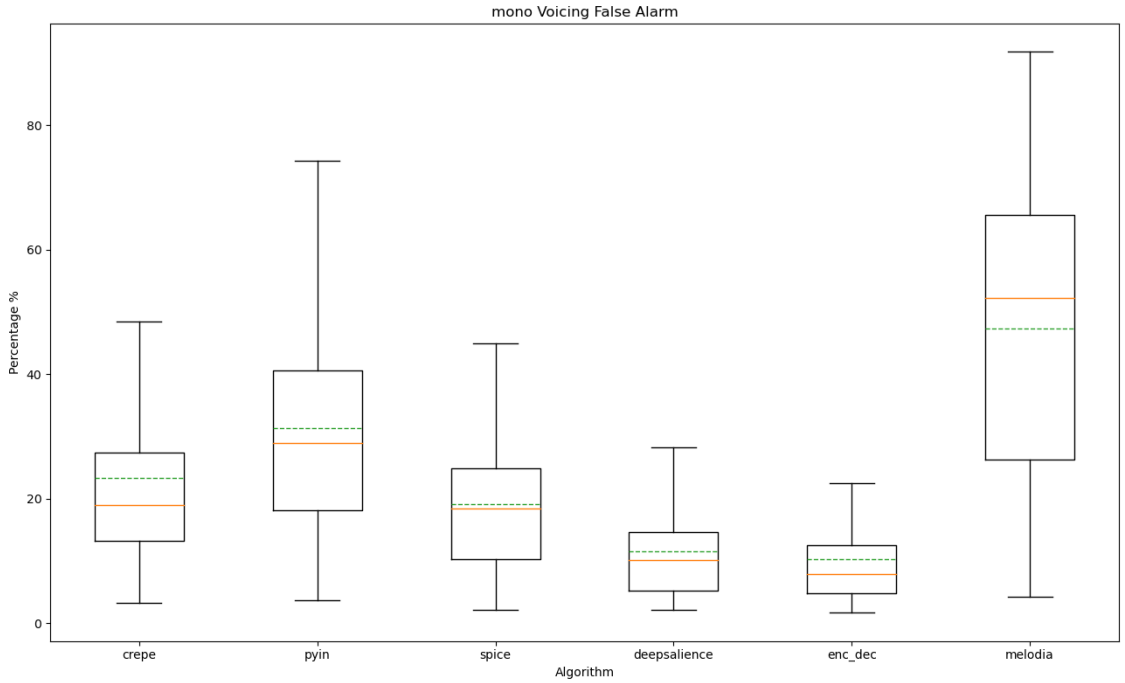


Figure 6: Mono Voicing False Alarm

Table 5: Mono Voicing False Alarm

| **Algorithm** | **Type** | **Rate Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 23.39 | 16.66 |
| pYIN | Monophonic | 31.30 | 18.81 |
| SPICE | Monophonic | 19.14 | 11.25 |
| Deep Salience | Polyphonic | 11.49 | 8.03 |
| Encoder/Decoder | Polyphonic | 10.22 | 9.50 |
| Melodia | Polyphonic | 47.33 | 23.21 |

less likely to incorrectly predict unvoiced frames as being voiced. The algorithms with the lowest VFA rate are SPICE for monophonic, and Encoder/Decoder for polyphonic. Both Melodia and pYIN have relatively high VFAs as well as high Voicing Recalls. This could be attributed to the algorithms predicting a voicing for a large portion of the total frames in a given track.

## 3.2 Polyphonic Test Set

This section contains the results of each model being tested on the mixed polyphonic tracks from MedleyDB.

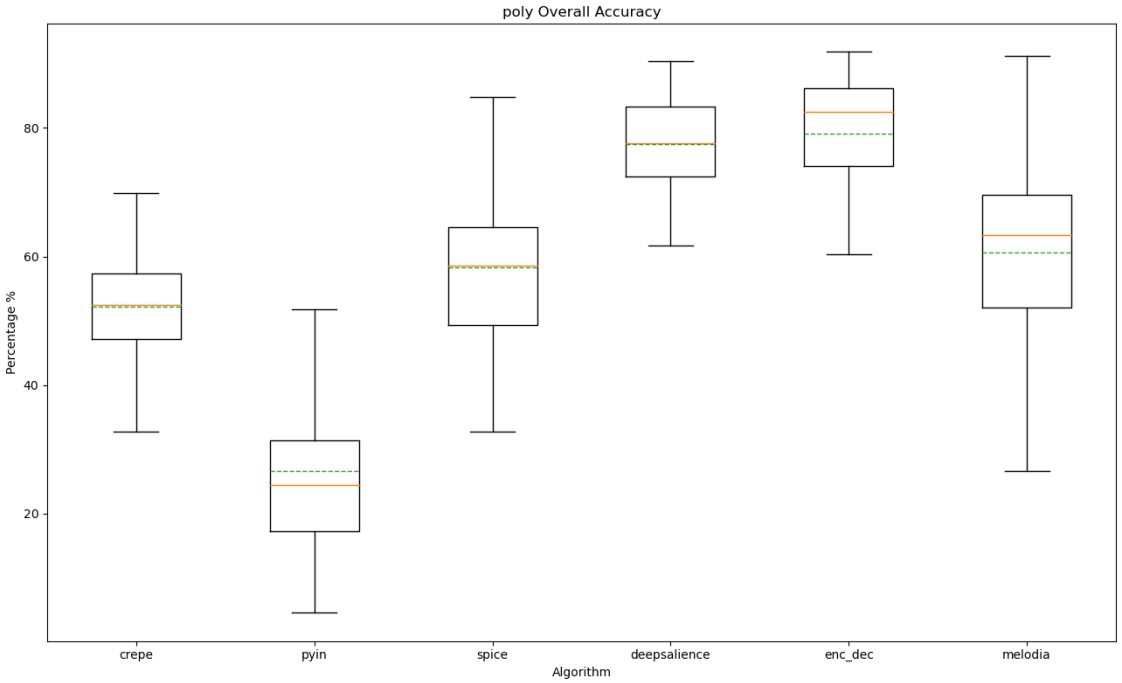


Figure 7: Poly Overall Accuracy

Table 6: Poly Overall Accuracy

| **Algorithm** | **Type** | **Accuracy Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 52.17 | 11.78 |
| pYIN | Monophonic | 26.60 | 13.45 |
| SPICE | Monophonic | 58.34 | 11.78 |
| Deep Salience | Polyphonic | 77.41 | 8.59 |
| Encoder/Decoder | Polyphonic | 79.08 | 9.47 |
| Melodia | Polyphonic | 60.63 | 14.59 |

The Overall Accuracies can be seen in Figure 7 and Table 6. The highest performing model is Encoder/Decoder with Deep Salience showing comparable results of around 79%. Each of the monophonic algorithms performs noticeably worse than the polyphonic algorithms. This can be easily attributed to the fact that these models were not trained or intended to be used to predict melody from polyphonic tracks. Therefore, these monophonic models likely have trouble discerning the correct harmonic information of the vocal melody from the other instruments present in the mix.

The Raw Pitch Accuracy can be seen in Figure 8 and Table 7. Deep Salience has the highest accuracy for polyphonic models with Encoder/Decoder and Melodia being

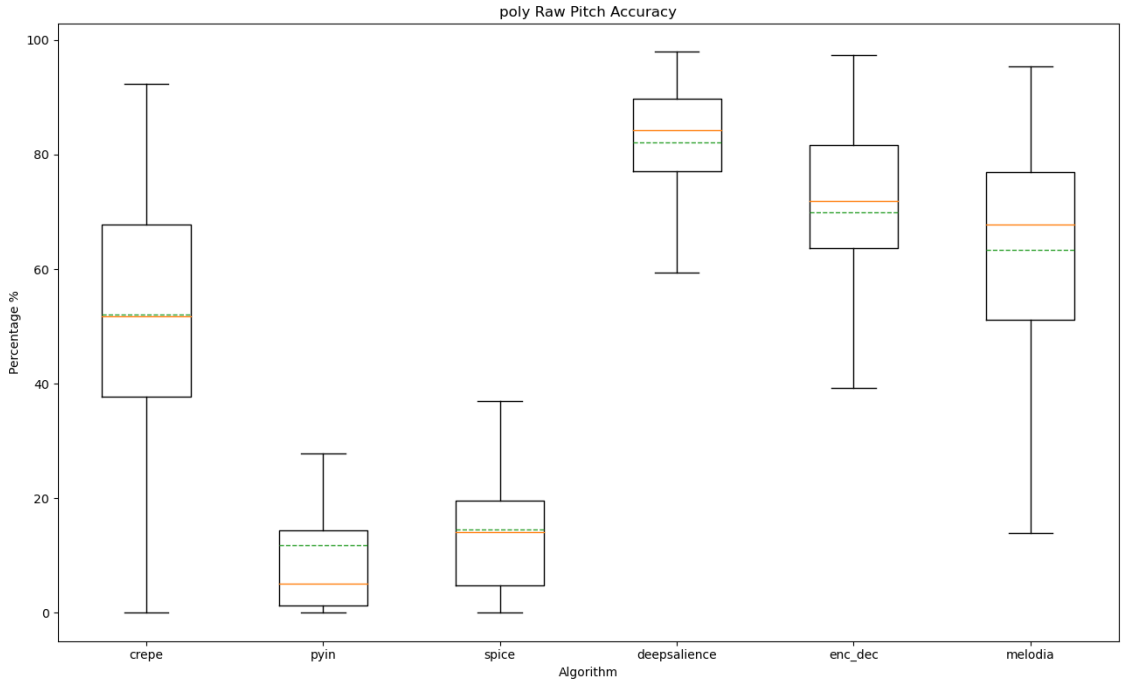


Figure 8: Poly Raw Pitch Accuracy

Table 7: Poly Raw Pitch Accuracy

| **Algorithm** | **Type** | **Accuracy Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 52.11 | 20.60 |
| pYIN | Monophonic | 11.79 | 15.15 |
| SPICE | Monophonic | 14.58 | 11.29 |
| Deep Salience | Polyphonic | 82.11 | 10.51 |
| Encoder/Decoder | Polyphonic | 69.86 | 18.09 |
| Melodia | Polyphonic | 63.30 | 21.01 |

slightly worse in performance as well as containing more variance.

Table 8: Poly Raw Chroma Accuracy

| **Algorithm** | **Type** | **Accuracy Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 50.23 | 19.46 |
| pYIN | Monophonic | 25.96 | 15.75 |
| SPICE | Monophonic | 14.74 | 11.31 |
| Deep Salience | Polyphonic | 84.78 | 9.51 |
| Encoder/Decoder | Polyphonic | 71.26 | 17.68 |
| Melodia | Polyphonic | 69.31 | 16.87 |

The Raw Chroma Accuracy can be seen in Figure 9 and Table 8. The results are very similar to the Raw Pitch Accuracy with Deep Salience having the best performance.

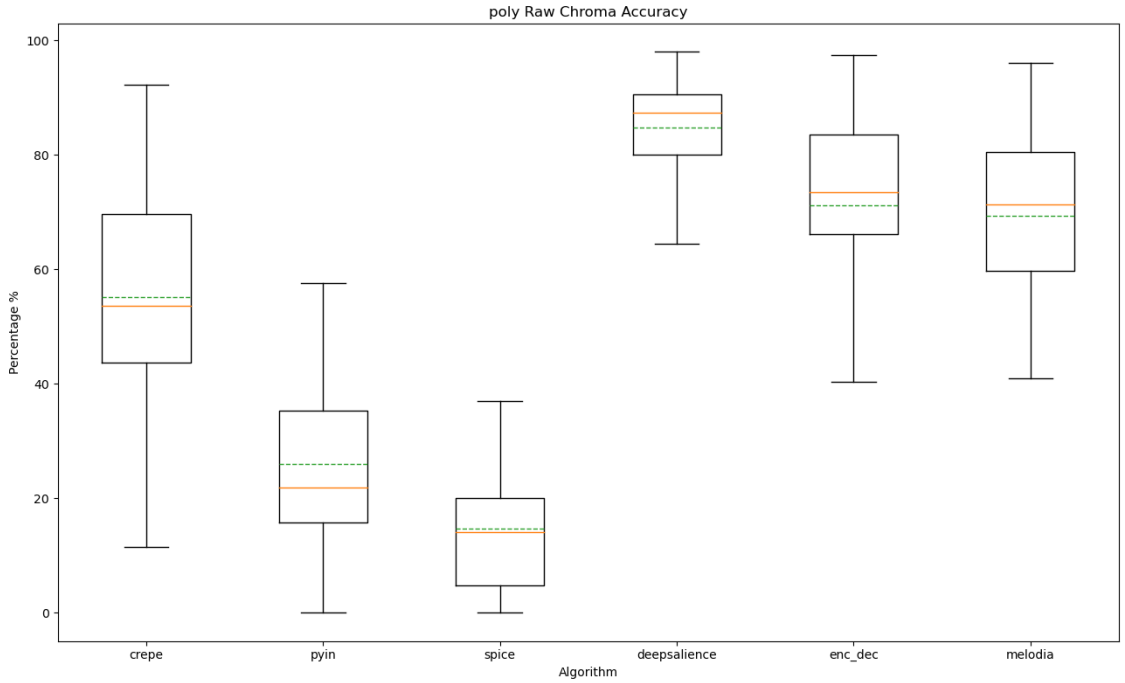


Figure 9: Poly Raw Chroma Accuracy

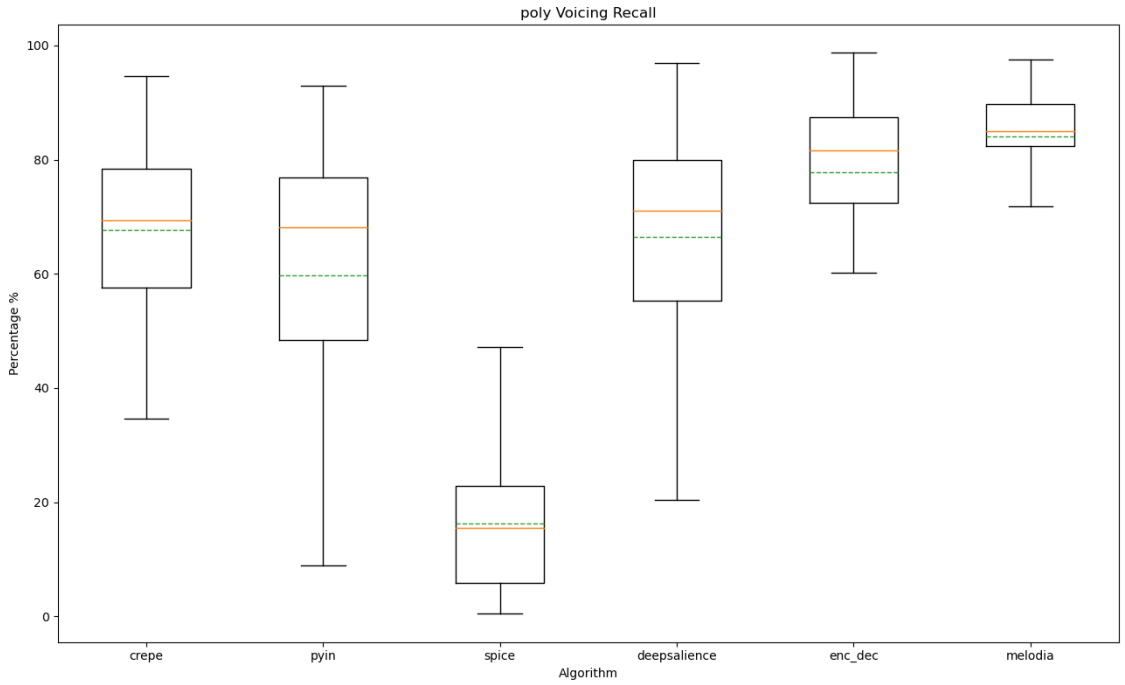


Figure 10: Poly Voicing Recall

The Voicing Recall rates can be seen in Figure 10 and Table 9. Melodia has the highest performance for this metric, however as previously mentioned in Section 3.1, Melodia was seen to predict numerous frames as being voiced, therefore leading to

Table 9: Poly Voicing Recall

| Algorithm | Type | Rate Value (%) | Standard Deviation (%) |
|---|---|---|---|
| CREPE | Monophonic | 67.61 | 15.16 |
| pYIN | Monophonic | 59.80 | 22.55 |
| SPICE | Monophonic | 16.30 | 12.01 |
| Deep Salience | Polyphonic | 66.50 | 20.18 |
| Encoder/Decoder | Polyphonic | 77.85 | 16.35 |
| Melodia | Polyphonic | 84.12 | 8.62 |

an inflated Voicing Recall rate.

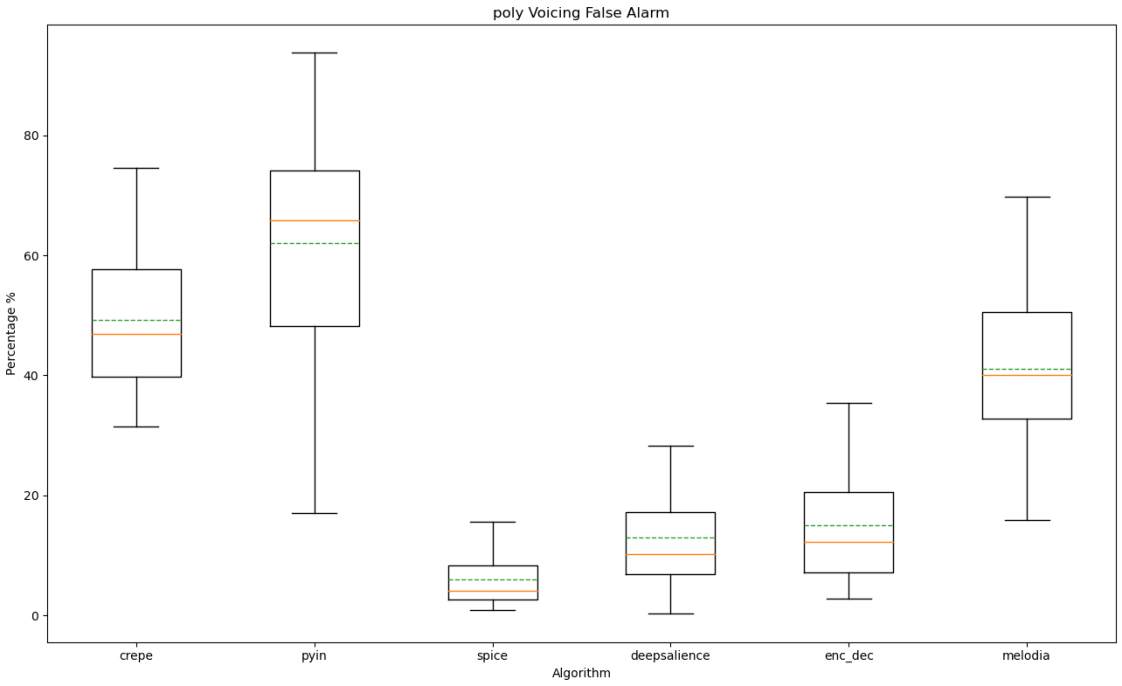


Figure 11: Poly Voicing False Alarm

Table 10: Poly Voicing False Alarm

| Algorithm | Type | Rate Value (%) | Standard Deviation (%) |
|---|---|---|---|
| CREPE | Monophonic | 49.30 | 11.50 |
| pYIN | Monophonic | 62.06 | 19.38 |
| SPICE | Monophonic | 5.96 | 4.90 |
| Deep Salience | Polyphonic | 12.96 | 9.62 |
| Encoder/Decoder | Polyphonic | 14.97 | 10.56 |
| Melodia | Polyphonic | 41.04 | 13.07 |

The Voicing False Alarm rates can be seen in Figure 11 and Table 10. Of the polyphonic models, Melodia has a very high VFA rate, whereas Deep Salience and

Encoder/Decoder perform much better. In terms of correctly estimating the frames in a (polyphonic) song as voiced or unvoiced Encoder/Decoder performs better than any of the models.

## 3.3 Source-Separation Test Set

This section contains the results of each model being tested on the vocal separated versions of the original mixed polyphonic tracks from MedleyDB. Source separation was done using the Open-Unmix tool.

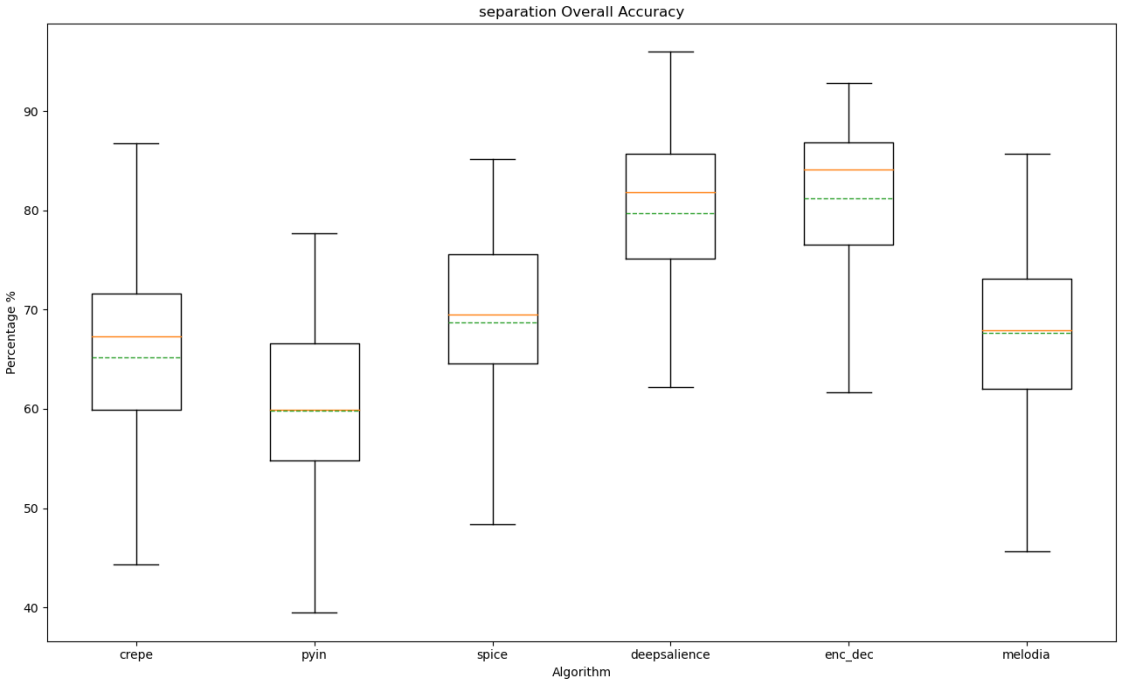


Figure 12: Separation Overall Accuracy

Table 11: Separation Overall Accuracy

| Algorithm | Type | Accuracy Value (%) | Standard Deviation (%) |
|---|---|---|---|
| CREPE | Monophonic | 65.21 | 10.23 |
| pYIN | Monophonic | 59.79 | 9.98 |
| SPICE | Monophonic | 68.74 | 9.08 |
| Deep Salience | Polyphonic | 79.70 | 8.87 |
| Encoder/Decoder | Polyphonic | 81.17 | 9.04 |
| Melodia | Polyphonic | 67.65 | 11.29 |

The Overall Accuracy can be seen in Figure 12 and Table 11. Encoder/Decoder and Deep Salience have similarly high accuracies of around 80%. Note that each

of the polyphonic models performs better on the source-separated data set than on the original polyphonic mixes.

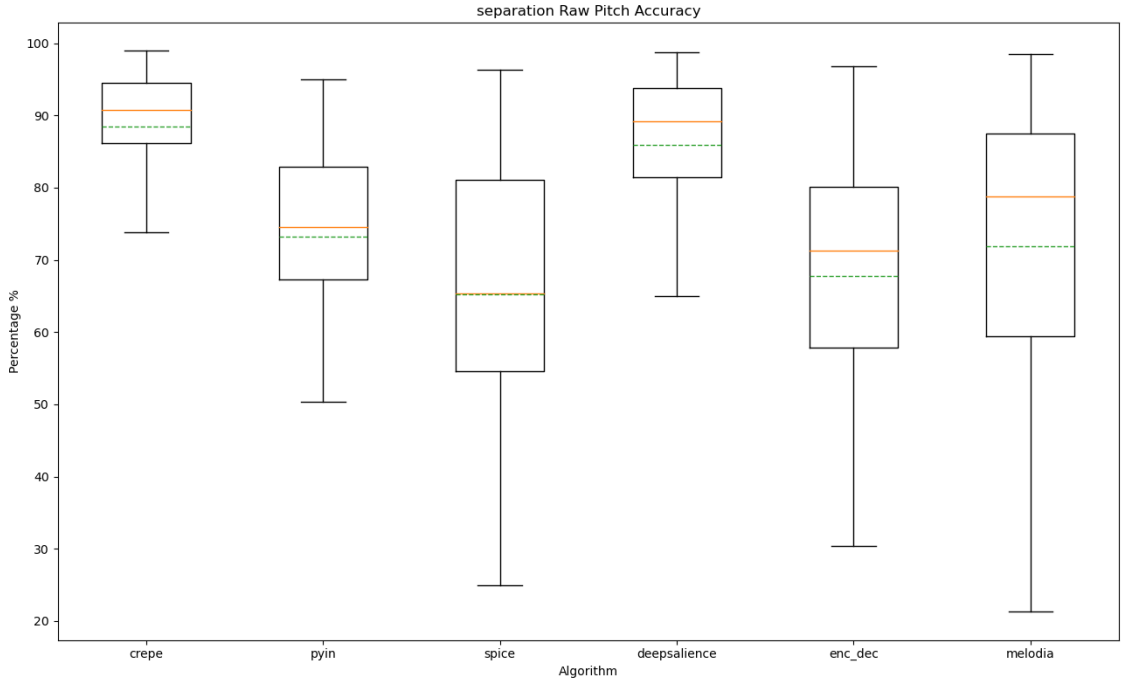


Figure 13: Separation Raw Pitch Accuracy

Table 12: Separation Raw Pitch Accuracy

| **Algorithm** | **Type** | **Accuracy Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 88.50 | 9.15 |
| pYIN | Monophonic | 73.17 | 13.36 |
| SPICE | Monophonic | 65.26 | 19.29 |
| Deep Salience | Polyphonic | 85.91 | 10.42 |
| Encoder/Decoder | Polyphonic | 67.76 | 17.05 |
| Melodia | Polyphonic | 71.89 | 20.24 |

The Raw Pitch Accuracy can be seen in Figure 13 and Table 12. CREPE is the highest performing algorithm with 88.5%, with Deep Salience having the highest performance of the polyphonic models.

The Raw Chroma Accuracy can be seen in Figure 14 and Table 13. The results are quite similar to the Raw Pitch Accuracy, with CREPE and Deep Salience having the best performance. There is a large difference in the RCA and RPA of Melodia, signifying that the estimation is being hindered by octave errors.

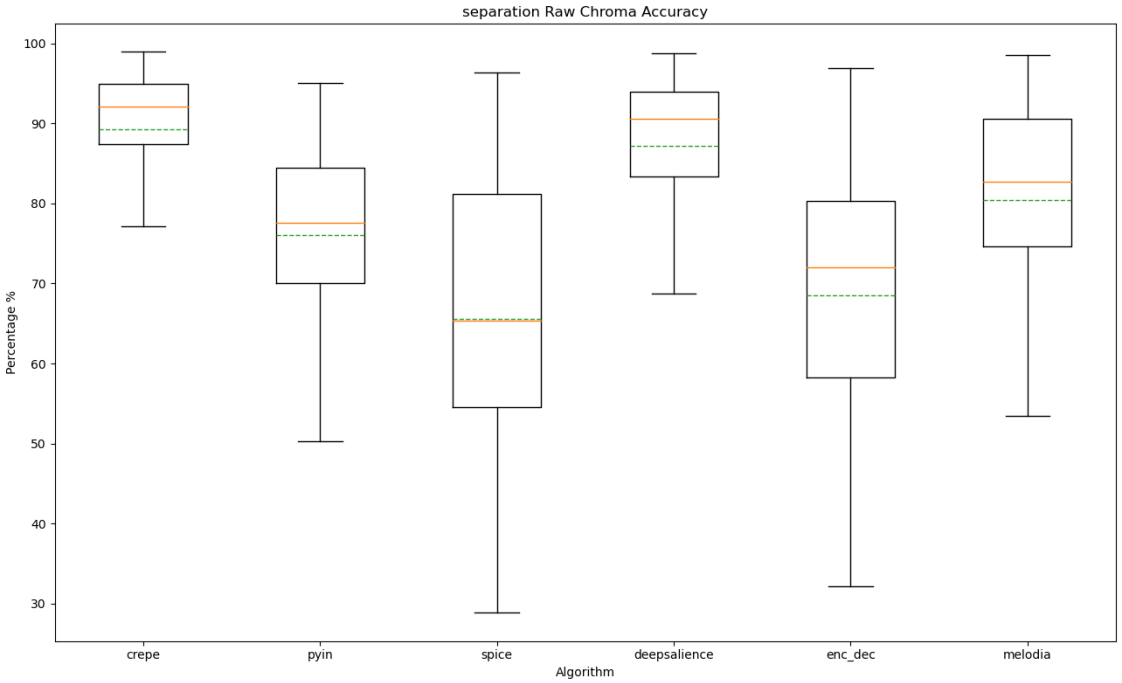


Figure 14: Separation Raw Chroma Accuracy

Table 13: Separation Raw Chroma Accuracy

| **Algorithm** | **Type** | **Accuracy Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 89.30 | 8.61 |
| pYIN | Monophonic | 76.07 | 11.44 |
| SPICE | Monophonic | 65.61 | 18.99 |
| Deep Salience | Polyphonic | 87.20 | 9.70 |
| Encoder/Decoder | Polyphonic | 68.55 | 16.76 |
| Melodia | Polyphonic | 80.38 | 12.58 |

Table 14: Separation Voicing Recall

| **Algorithm** | **Type** | **Rate Value (%)** | **Standard Deviation (%)** |
|---|---|---|---|
| CREPE | Monophonic | 94.82 | 4.20 |
| pYIN | Monophonic | 93.85 | 5.12 |
| SPICE | Monophonic | 70.56 | 17.80 |
| Deep Salience | Polyphonic | 81.27 | 13.46 |
| Encoder/Decoder | Polyphonic | 73.94 | 15.93 |
| Melodia | Polyphonic | 89.05 | 6.94 |

The Voicing Recall rates can be seen in Figure 12 and Table 11. CREPE, pYIN, and Melodia each have very high Voicing Recall.

The Voicing False Alarm rates can be seen in Figure 12 and Table 11. Compared

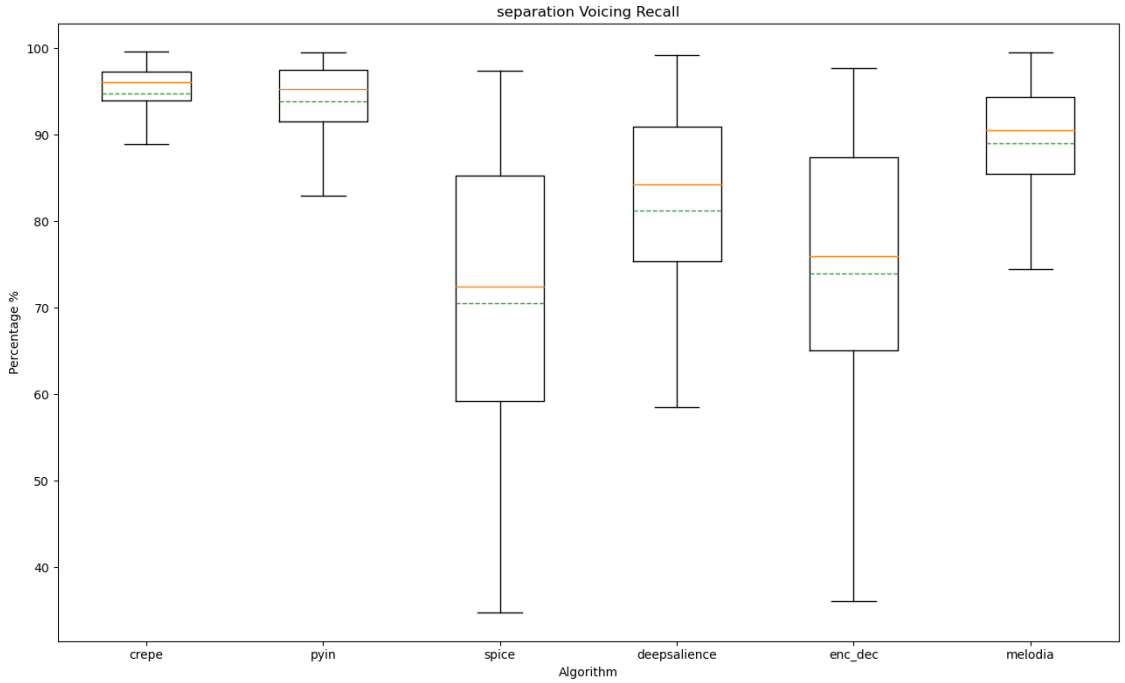


Figure 15: Separation Voicing Recall

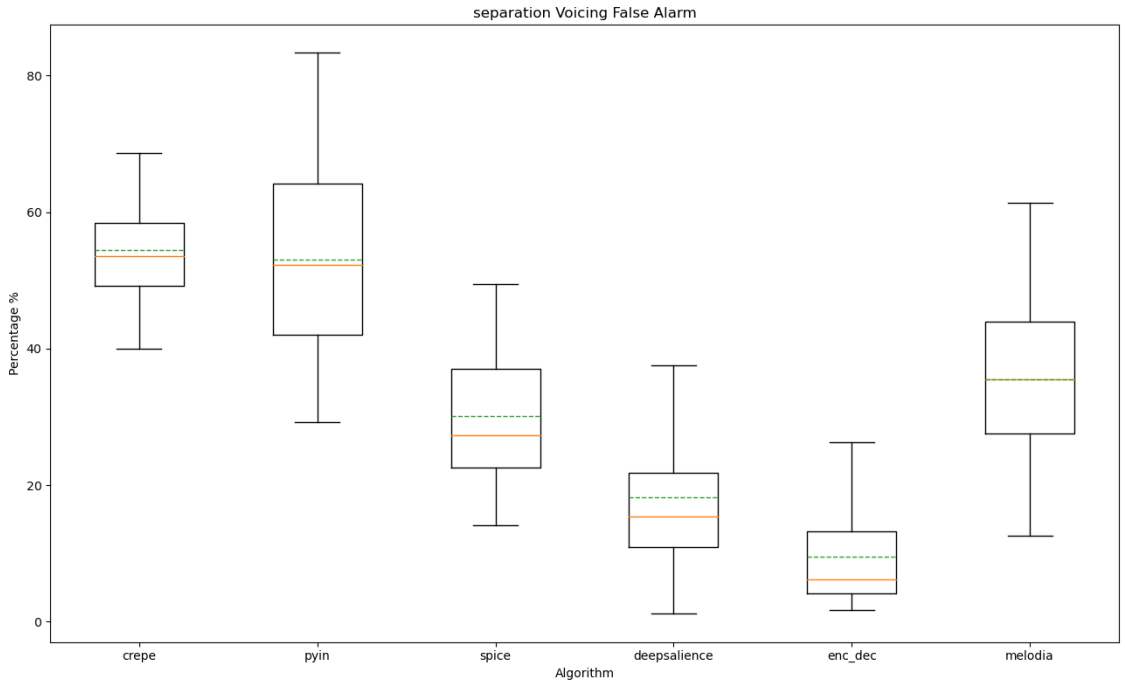


Figure 16: Separation Voicing False Alarm

to the monophonic and polyphonic test sets, the VFA rates for the source-separated set are much higher for the original mono models.

Table 15: Separation Voicing False Alarm

| Algorithm | Type | Rate Value (%) | Standard Deviation (%) |
|---|---|---|---|
| CREPE | Monophonic | 54.45 | 8.93 |
| pYIN | Monophonic | 53.05 | 14.23 |
| SPICE | Monophonic | 30.14 | 10.96 |
| Deep Salience | Polyphonic | 18.18 | 10.99 |
| Encoder/Decoder | Polyphonic | 9.47 | 9.17 |
| Melodia | Polyphonic | 35.49 | 11.64 |

## 3.4 Example Estimations

In this section, I provide several figures of estimations compared to their ground truth annotations. I chose songs where the estimations are highly accurate and highly inaccurate in terms of Overall Accuracy and Voicing False Alarm for each of the monophonic, polyphonic, and source-separated test sets.

Figures 17, 18, and 19 contain calculated estimations of the monophonic, polyphonic, and source-separated test sets respectively. These figures demonstrate poor Overall Accuracies, where unvoiced frames are incorrectly predicted as voiced and pitch estimations are not within the range of one semitone of the reference frequency.

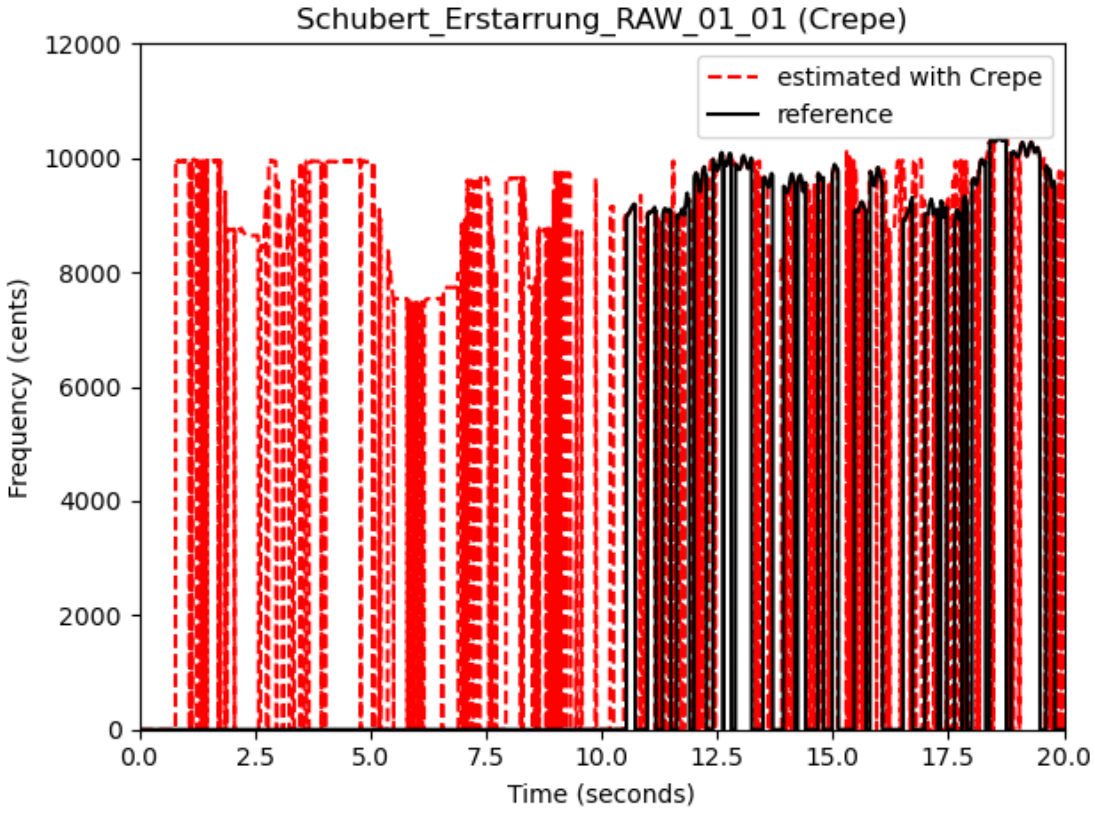


Figure 17: Minimum OA for Monophonic

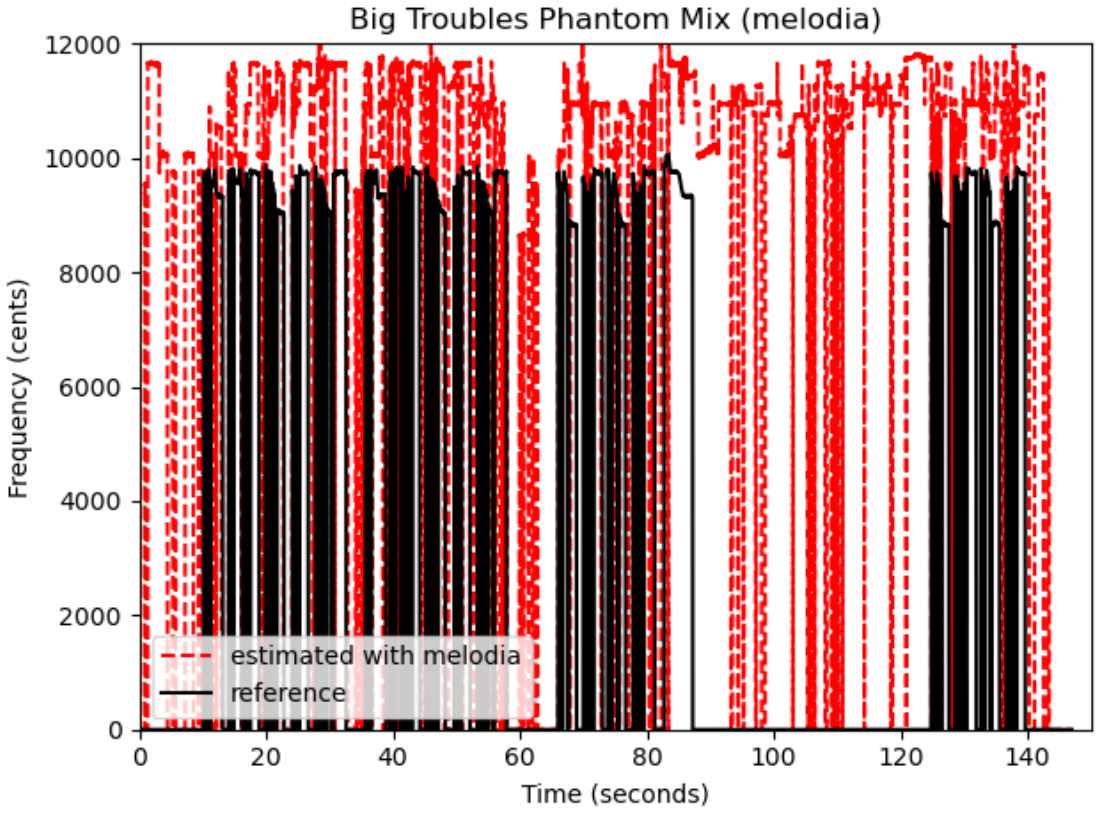


Figure 18: Minimum OA for Polyphonic

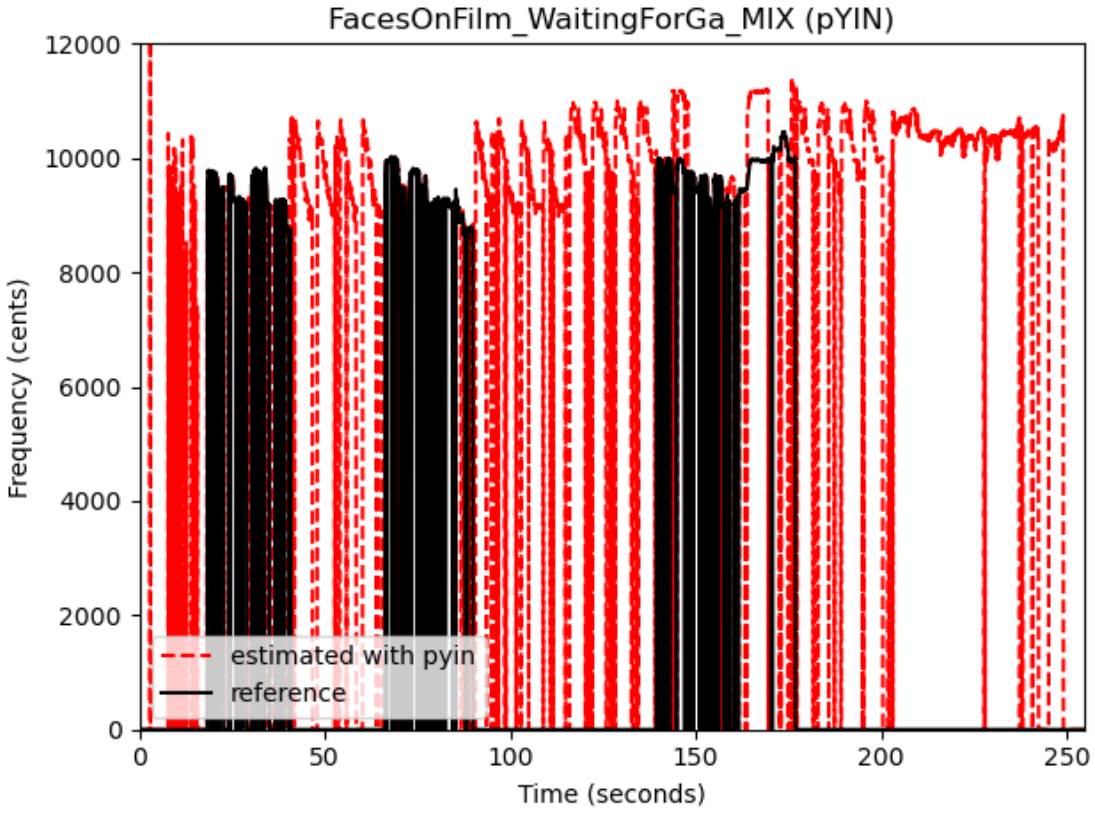


Figure 19: Minimum OA for Separation

Figures 20, 21, and 22 demonstrate high Overall Accuracies, where unvoiced frames are correctly predicted as unvoiced and pitch estimations are within the range of one semitone of the reference frequency.

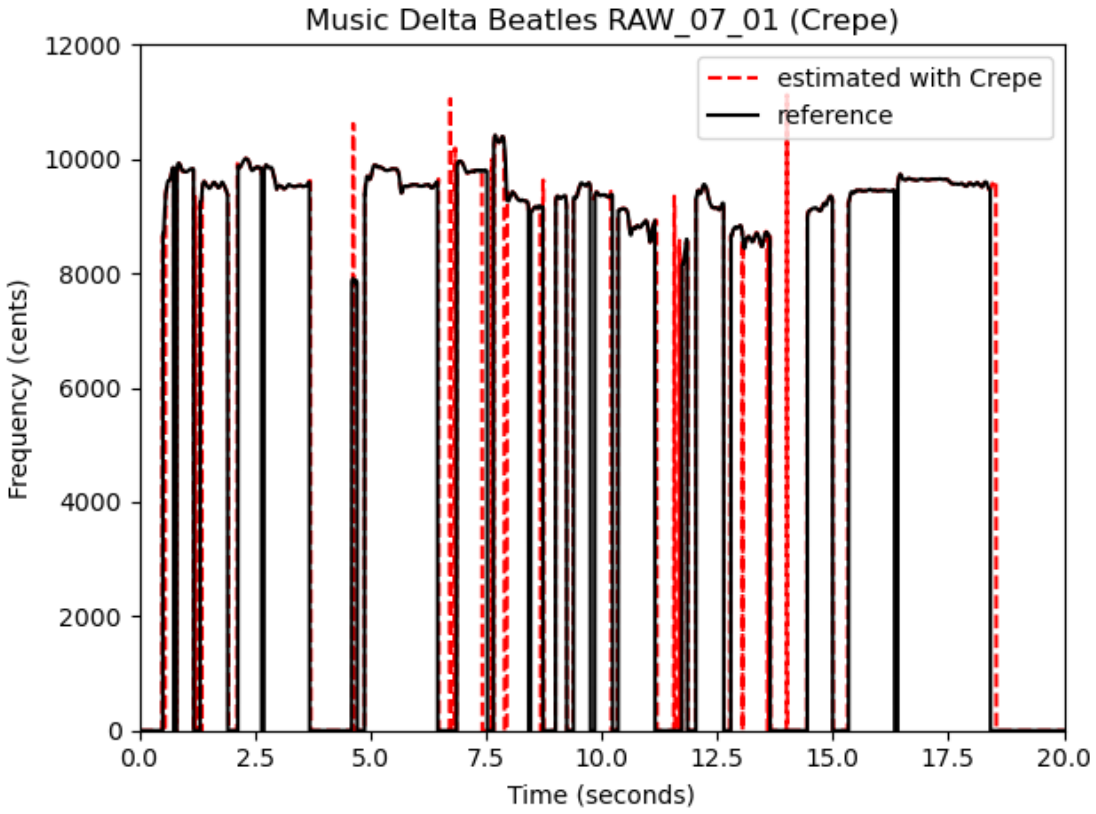


Figure 20: Maximum OA for Monophonic

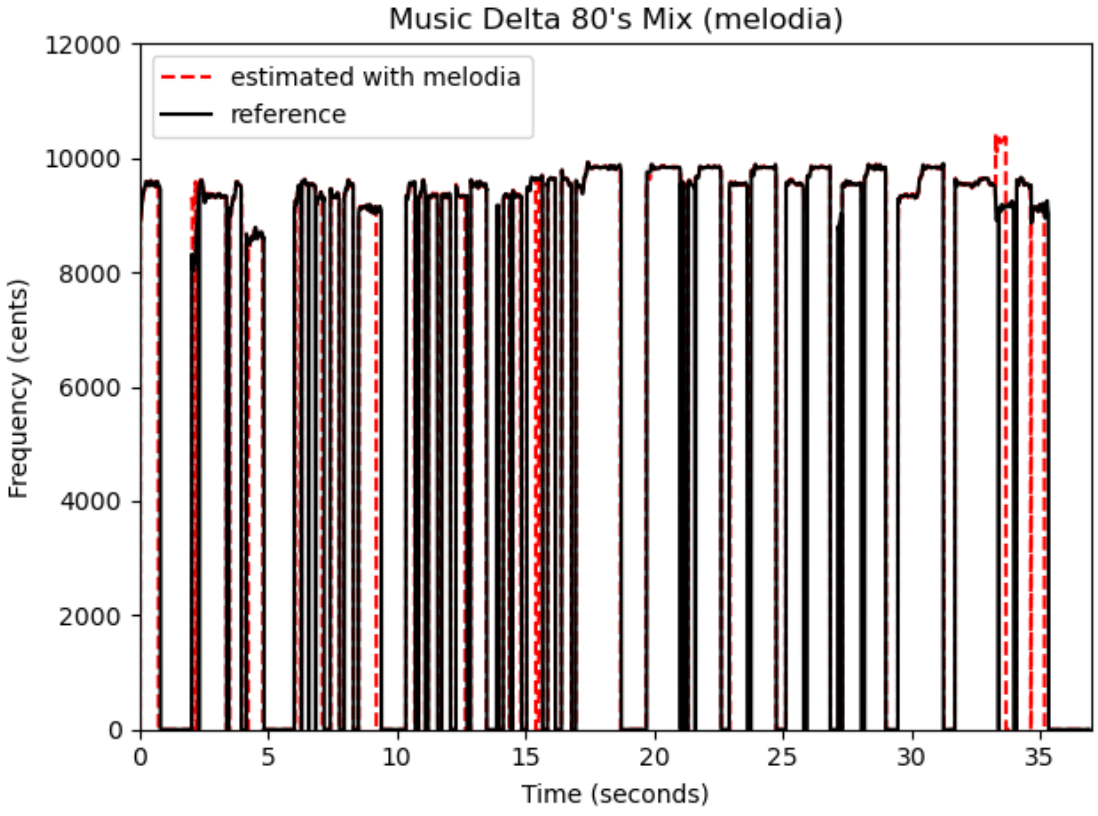


Figure 21: Maximum OA for Polyphonic

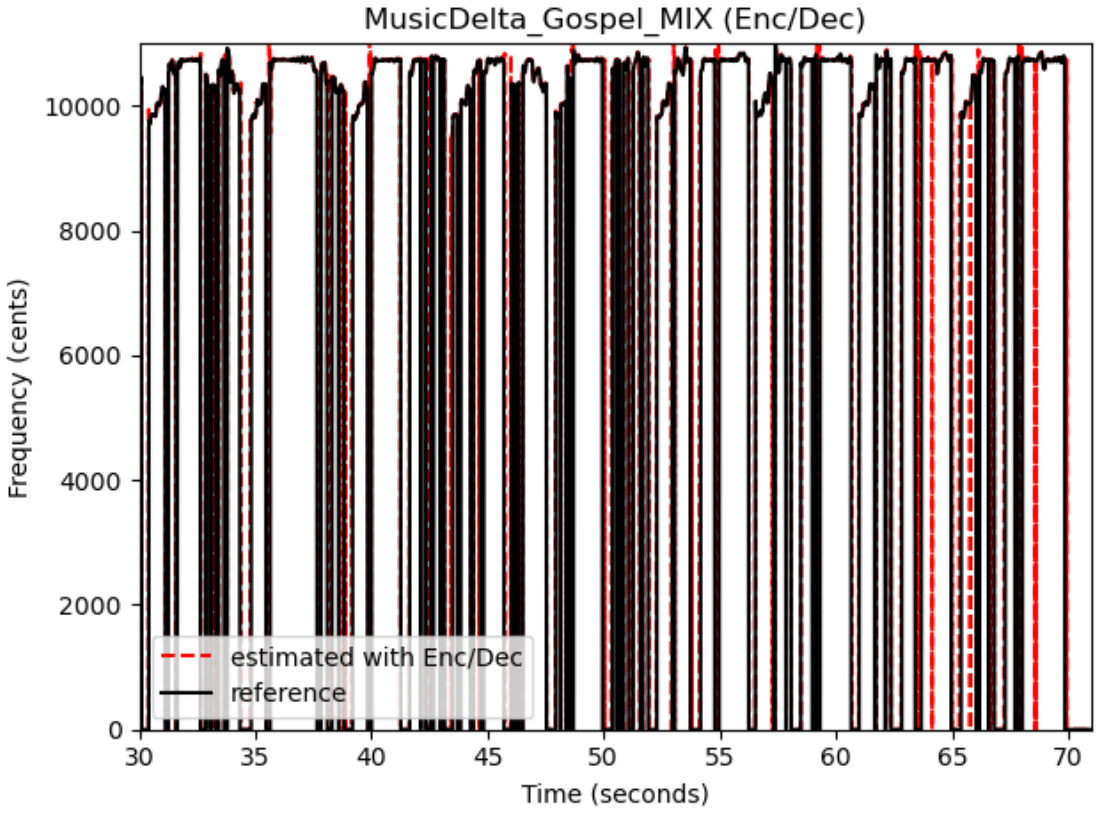


Figure 22: Maximum OA for Separation

Figures 23, 24, and 25 demonstrate low Voicing False Alarm rates, where unvoiced frames are correctly predicted as unvoiced.

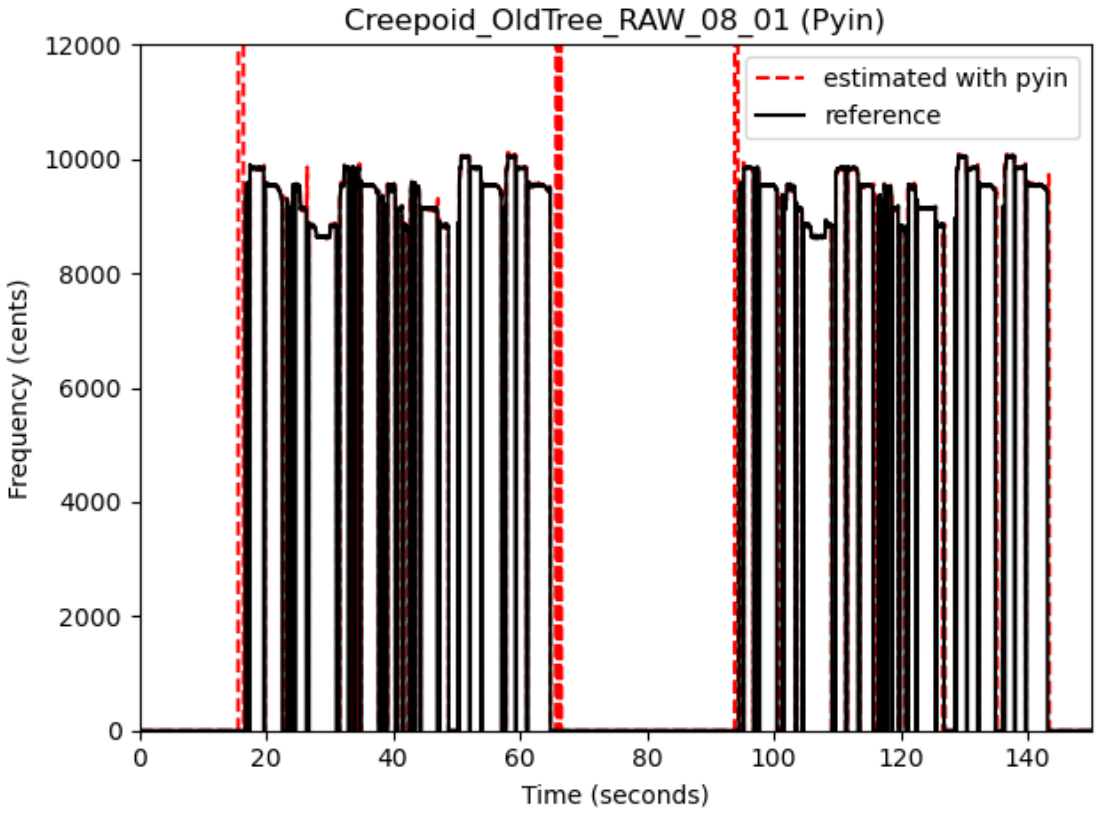


Figure 23: Minimum VFA for Monophonic

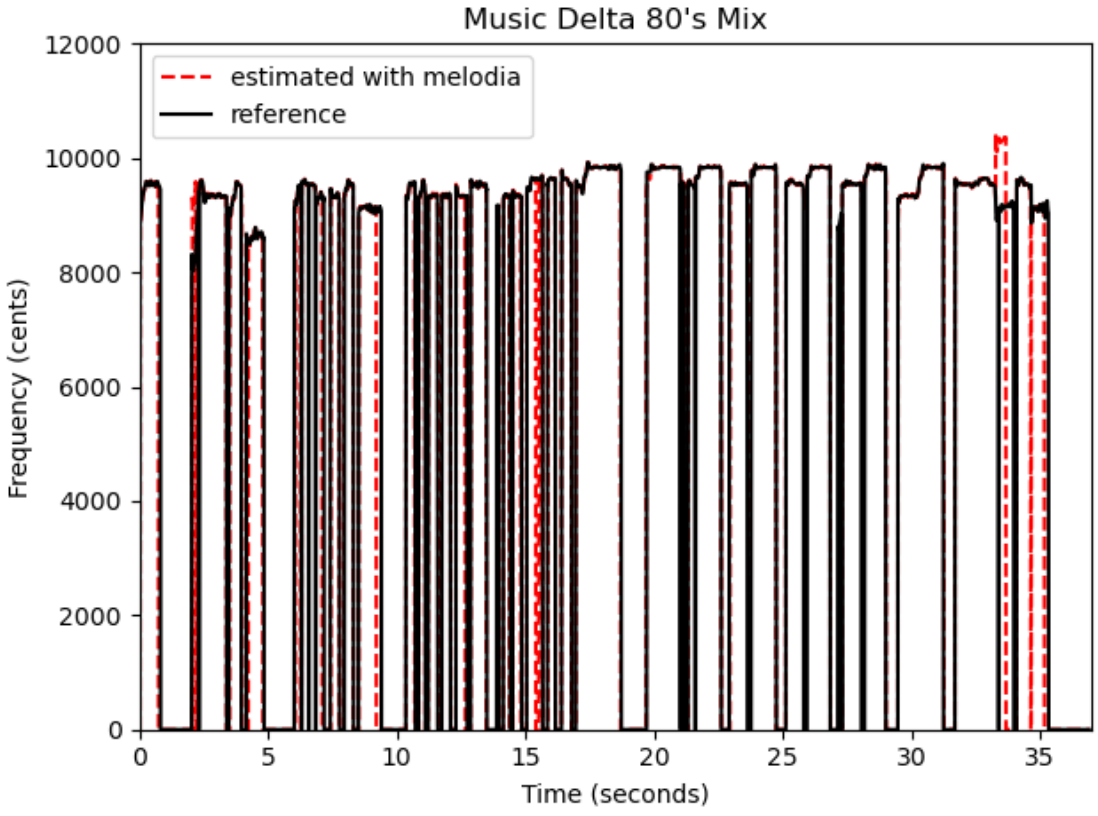


Figure 24: Minimum VFA for Polyphonic


Figure 25: Minimum VFA for Separation

Figures 26, 27, and 28 demonstrate high Voicing False Alarm rates, where unvoiced frames are incorrectly predicted as voiced.

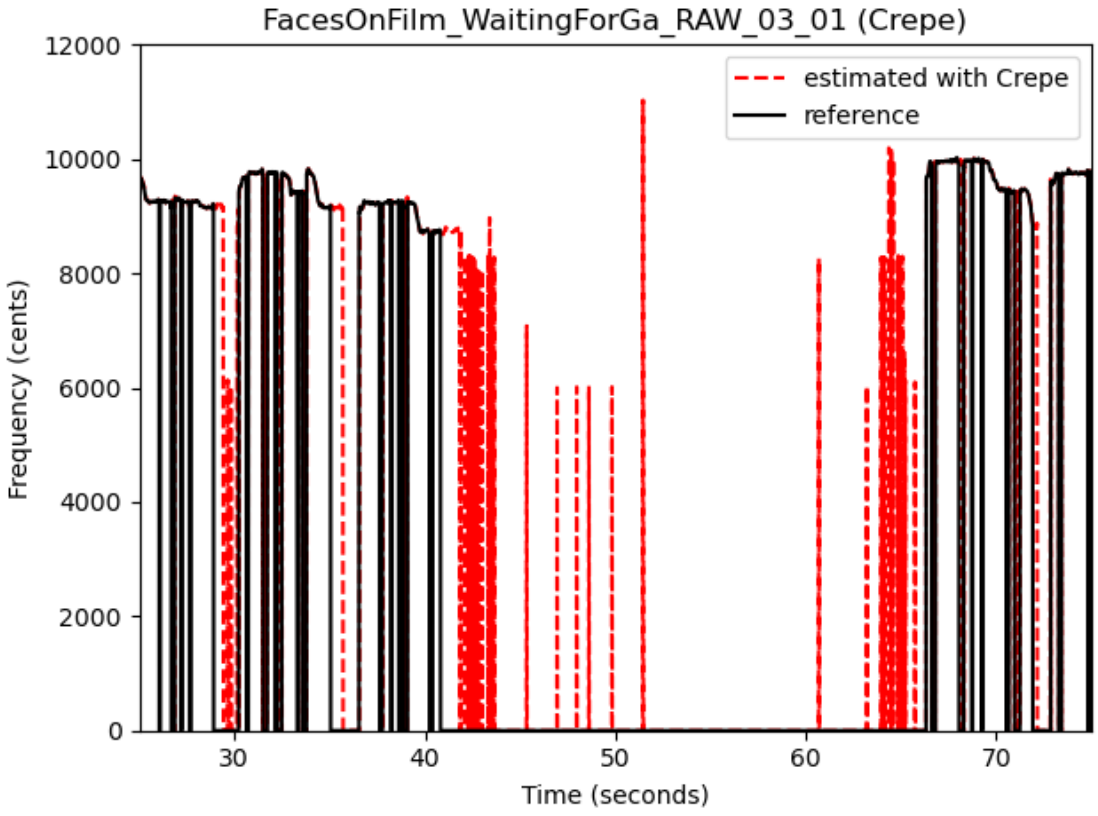


Figure 26: Maximum VFA for Monophonic

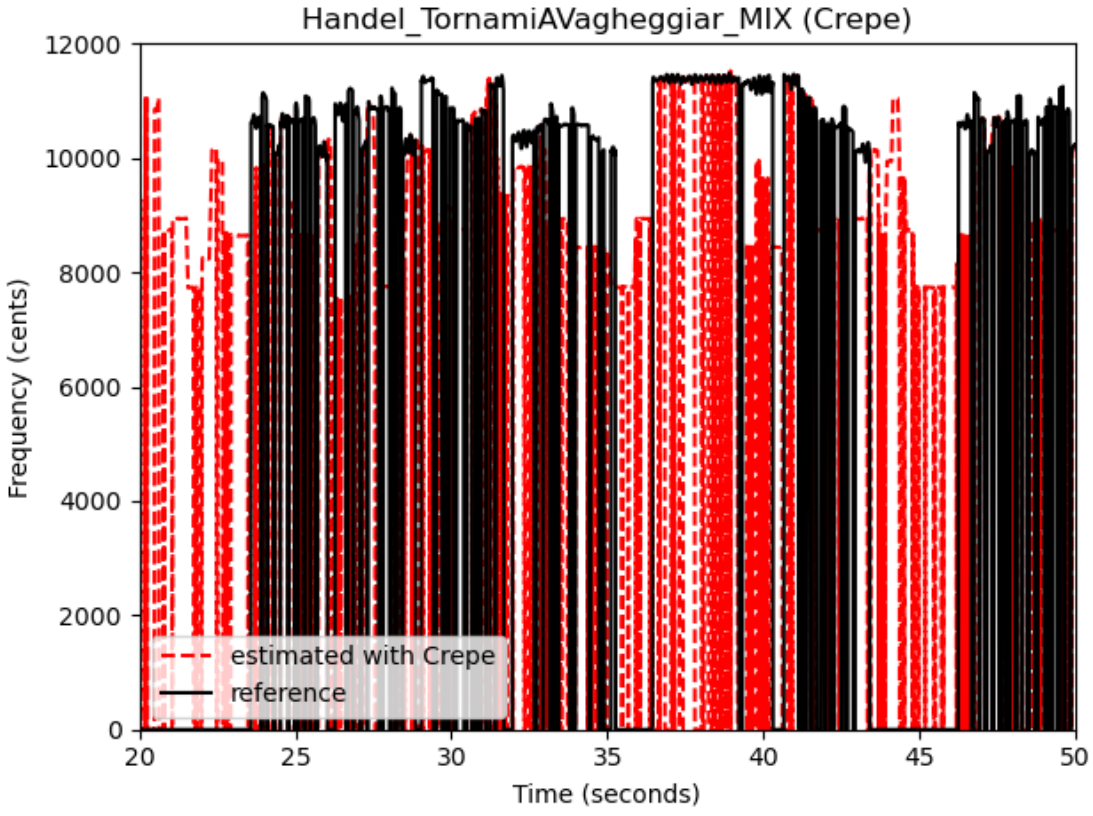


Figure 27: Maximum VFA for Polyphonic

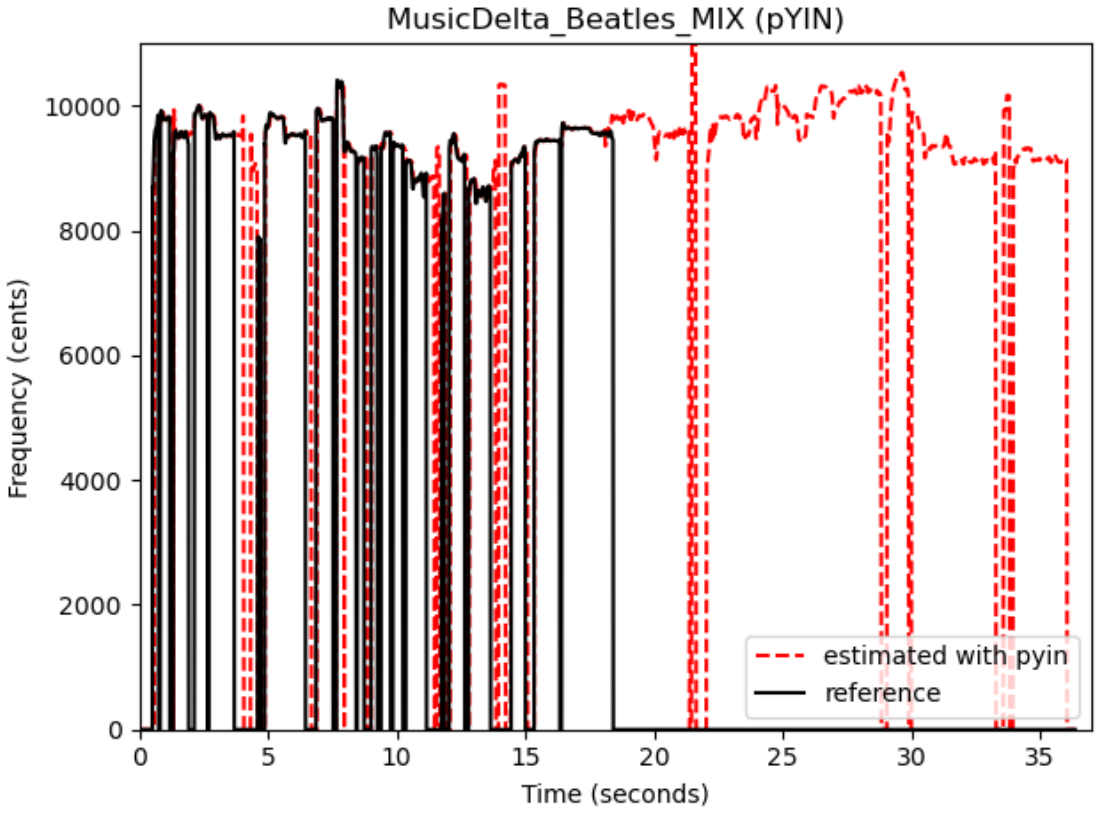


Figure 28: Maximum VFA for Separation

# Chapter 4

# Discussion

## 4.1 Discussion

The overall accuracy for CREPE (monophonic) allows a benchmark measure for the polyphonic algorithms whose accuracies should not exceed CREPE due to the inherent errors that occur when estimating f0 from polyphonic music. While the results do support previous findings that CREPE outperforms pYIN [6], several considerations must also be taken into account for this analysis. The first consideration is that the CREPE algorithm was trained on several datasets, including MedleyDB. Using the same data for training and testing the algorithm creates a bias in the performance of CREPE and could explain why the accuracy of CREPE is significantly higher than that of pYIN and SPICE. Secondly, the annotation process for MedleyDB includes manual corrections which do not guarantee a 100% perfect match between the annotation and the audio. Therefore, the results can also be affected by some degree of human subjectivity. Thirdly, algorithm parameters such as the voicing confidence threshold (50%) chosen for CREPE and the minimum (65.41 Hz) and maximum (2093 Hz) frequencies chosen for pYIN may also affect the results. For future work, I recommend finding an additional dataset, other than the one used in the training processes of these algorithms, to be used as a test set for evaluation. Ideally, this dataset would not contain manual annotations, which are subjective and do not necessarily guarantee a 100% match between the annotation and audio. And the al-

gorithmic parameters such as voicing confidence threshold and minimum/maximum frequencies should be further analyzed to view differences in terms of accuracy. I also propose evaluating the performance of these algorithms on datasets of different kinds of music including non-western or songs with multiple singers in harmony. It should also be noted that the Deep Salience model outperforms CREPE in Overall Accuracy when tested on the monophonic subset. Even though the model was originally trained on polyphonic signals, the Deep Salience model can be seen as a versatile model able to predict pitch with high overall accuracy for both monophonic and polyphonic signals.

Unsurprisingly, each of the models intended for use on monophonic signals performs poorly on the polyphonic test set, and their results should be more or less ignored. The models with the highest overall accuracy on the polyphonic test set are Deep Salience and Encoder/Decoder. Deep Salience outperforms Encoder/Decoder in terms of RPA and RCA, however, the Encoder/Decoder model has a higher Voicing Recall rate. According to this work, Deep Salience is the highest performing model for both the monophonic and polyphonic test sets and should be considered state-of-the-art.

Testing each of the algorithms on the vocal source-separated test set shows promising results. Aside from an increasing VFA rate for SPICE, each of the algorithms originally intended for monophonic signals (CREPE, pYIN, and SPICE) improve in every metric when comparing the polyphonic and source-separated estimations. Thus, these monophonic models (and especially the high-performing CREPE) are viable candidates for vocal melody estimation when source separation is included in preprocessing. The pitch-based metrics are comparable between the monophonic and source separated test sets for CREPE and pYIN, however, there is quite a large discrepancy between the voicing-based metrics. The VFA for CREPE goes from 23.39% to 54.45% between the monophonic and source-separated test sets. Similar results can be seen in the voicing-based metrics of pYIN and SPICE. This could potentially be attributed to artifacts or noise left in the separated tracks that the algorithms are not able to correctly label as unvoiced frames. I suggest fur-

thering this research by evaluating other test sets using the same processes with other source separation tools. The algorithms originally intended for polyphonic signals (Deep Salience, Encoder/Decoder, and Melodia) also each show improvements in Overall Accuracy when comparing the results of the polyphonic test set to the source-separated test set. Melodia improved in every metric. Deep Salience improved in RCA, RPA, and VR, however, the VFA rate also increased. Although the pitch accuracy is higher for the source separation test set, it should be noted that the process for generating the estimation takes longer than many of the other applied algorithms. This suggests that source separation may not be suitable for tasks in which melodic estimation is a time-sensitive matter or in real-time applications. However, the relative improvements that source separation provides should not be ignored.

## 4.2 Conclusions

In this work, I evaluate the performance of several popular f0 estimation algorithms for the task of singing voice melodic estimation. Estimation of monophonic algorithms provides a benchmark to compare polyphonic models and CREPE is seen to outperform pYIN and SPICE. Deep Salience performs the best of any of the polyphonic models and also performs quite well on monophonic signals. The most accurate models in terms of Raw Pitch Accuracy are CREPE and Deep Salience when evaluated on a source-separated test set. While the additional process of source separation can be more intensive than most of the out-of-the-box polyphonic methods, the pitch accuracy suggests potential room for improvement in the state-of-the-art when using such models and I suggest further research in this domain.

# Bibliography


[1] Poliner, G. E. *et al.* Melody transcription from music audio: Approaches and evaluation. *IEEE Trans. Audio, Speech, Lang. Processing* **15**, 1247–1256 (2007).

[2] Moore, B. C. J. An introduction to the psychology of hearing. *Academic Press* (2003).

[3] de Cheveigne, A. & Kawahara, H. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America* 1917–1930 (2002).

[4] Rahman, M. A. & Shimamura, T. Pitch determination using autocorrelation function in spectral domain. In *INTERSPEECH* (2010).

[5] Mauch, M. & Dixon, S. Pyin: A fundamental frequency estimator using probabilistic threshold distributions. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 659–663 (2014).

[6] Kim, J. W., Salamon, J., Li, P. & Bello, J. P. Crepe: A convolutional representation for pitch estimation. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 161–165 (2018).

[7] Gfeller, B. *et al.* Spice: Self-supervised pitch estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **28**, 1118–1128 (2019).

[8] Salamon, J., Gómez, E., Ellis, D. P. & Richard, G. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine* **118** (2014).

[9] Salamon, J. & Gomez, E. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing* 1759–1770 (2012).

[10] Doras, G., Esling, P. & Peeters, G. On the use of u-net for dominant melody estimation in polyphonic music. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, 66–70 (IEEE, 2019).

[11] Bittner, R., McFee, B., Salamon, J., Li, P. & Bello, J. Deep salience representations for $f_0$ estimation in polyphonic music. In *18th Int. Soc. for Music Info. Retrieval Conf.* (Suzhou, China, 2017).

[12] Dong, M., Wu, J. & Luan, J. Vocal pitch extraction in polyphonic music using convolutional residual network. *Proc. Interspeech 2019* (2019).

[13] Gao, Y., Zhang, X. & Li, W. Vocal melody extraction via hrnet-based singing voice separation and encoder-decoder-based f0 estimation. *Electronics* **10**, 298 (2021).

[14] Défossez, A., Usunier, N., Bottou, L. & Bach, F. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254* (2019).

[15] Stoter, F.-R., Uhlich, S., Liutkus, A. & Mitsufuji, Y. Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software* (2019). URL `https://doi.org/10.21105/joss.01667`.

[16] Hsieh, T.-H., Su, L. & Yang, Y.-H. A streamlined encoder/decoder architecture for melody extraction (2019). `1810.12947`.

[17] Bittner, R. & Bosch, J. J. Generalized metrics for single-f0 estimation evaluation. *20th International Society for Music Information Retrieval Conference* (2019).

[18] Bittner, R. *et al.* Medleydb: A multitrack dataset for annotation-intensive mir research. *International Society for Music Information Retrieval Conference* (2014).

[19] McFee, B. *et al.* librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, vol. 8 (2015).

[20] Raffel, C. *et al.* mir_eval: A transparent implementation of common mir metrics. *Proceedings of the 15th International Conference on Music Information Retrieval* (2014).