POLYPHONIC PITCH TRACKING BY EXAMPLE

Paris Smaragdis

University of Illinois at Urbana-Champaign
Advanced Technology Labs, Adobe Systems Inc.

ABSTRACT

We introduce a novel approach for pitch tracking of multiple sources in mixture signals. Unlike traditional approaches to pitch tracking, which explicitly attempt to detect periodicities, this approach is using a learning framework by making use of previously pitch-tagged recordings as training data to teach spectrum/pitch associations. We show how the mixture case of this task is a nearest subspace search problem which is efficiently solved by transforming it to an overcomplete sparse coding formulation. We demonstrate the use of this algorithm with real mixtures ranging from solo up to a quintet recordings.

Index Terms— Polyphonic pitch tracking

1. INTRODUCTION

Pitch tracking is a problem that has always been central in the audio sciences. Traditionally, this problem is approached as one of a physical measurement, the goal of which is to measure how many times a second a waveform repeats. Formulated this way, in both the single and multiple source case, one can imagine a wealth of options to perform this task ranging from simple time-domain cross-correlation to more complex systems that examine a relevant feature space, such as the Fourier transform, constant-Q transforms, as well as more sophisticated approaches [1].

A criticism of that approach is that humans do not seem to operate by making exact periodicity measurements, but rather we learn to recognize notes as we get increasingly experienced with music. In this paper we will try to replicate this model of pitch perception. We will show that we can efficiently learn to perform pitch tracking of polyphonic signals given training that is designed to teach a map of sounds to multiple pitches (or notes). We will keep this approach in the context of large training data, and in the process we will remove the need for a complex model and use a very simple learning core instead.

In the remaining sections of this chapter we will first define the basic framework and work our way from a simple toy example to a real woodwind quintet recording.

2. PITCH-TRACKING BY EXAMPLE

Let us consider a naive way of performing pitch tracking by example. Assume that we have two already pitch-tagged recordings $x_i(t), i = \{1, 2\}$, in this example recordings of a piano and a flute. We take the magnitude spectrograms of each of these recordings which we represent with time-stamped magnitude spectral vectors $\mathbf{f}_{x_i}(t)$. Because we are only concerned to estimate pitch, loudness is an irrelevant factor so we take the additional step of normalizing these vectors so that they sum to 1. For the recordings at hand, these vectors are shown at the top plots of figure 1. Additionally we obtain a pitch label $p_{x_i}(t)$ for each $\mathbf{f}_{x_i}(t)$ by using a regular pitch tracker. These are shown in the center plots of figure 1. Let us assume that having knowledge of the above data we are now presented with two more recordings $y_i(t)$ of the same instruments for which we compute $\mathbf{f}_{y_i}(t)$ as described above. Instead of using the pitch tracker again we can use an alternative approach by leveraging the available data. For each spectrum $\mathbf{f}_{y_i}(t)$ we find its most similar spectrum out of $\mathbf{f}_{x_i}(t)$ and use its pitch label to describe $\mathbf{f}_{y_i}(t)$, i.e.:

$$\tau = \underset{\tau}{\operatorname{argmin}}\, \mathrm{D}\left[\mathbf{f}_{y_i}(t)||\mathbf{f}_{x_i}(\tau)\right] \quad (1)$$

$$\hat{p}_{y_i}(t) \equiv p_{x_i}(\tau), \quad (2)$$

where $\mathrm{D}[\cdot||\cdot]$ is a user-defined similarity metric between two spectra, and $\hat{p}_{y_i}(t)$ is our estimate of the pitch of $\mathbf{f}_{y_i}(t)$. This is essentially a nearest-neighbor search from the training data after which we associate the tag of the nearest neighbor to the query data point. The only assumption that we make is that the correct pitch tags for $y_i(t)$ are a subset of the training data pitch tags $p_{x_i}(t)$. If that assumption does not hold then there isn't enough information to properly tag new pitch values.

Running this process on the data in figure 1 we obtain the results shown in figure 2. For the function $\mathrm{D}[\cdot||\cdot]$ we used the Kullback-Leibler divergence since it provides an appropriate distance-like measure for normalized non-negative data [2]. As a simple measure of performance we estimate the mean and standard deviation of the difference between the true pitch values $p_{y_i}(t)$, which we can

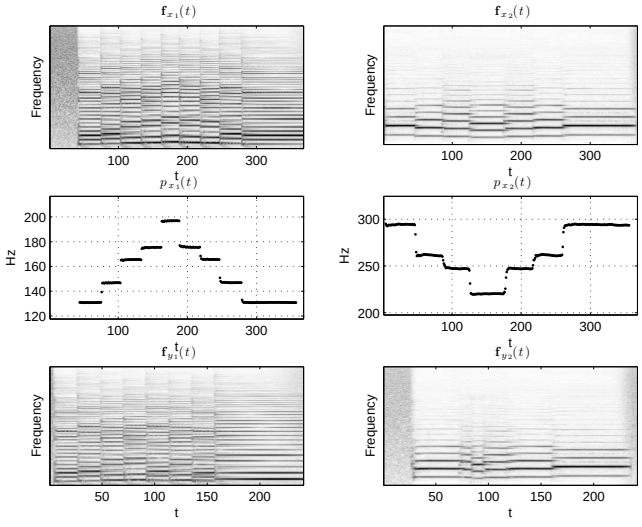

Figure 1: The pitched-tracked spectra of two sources are shown in the top figures. Their respective pitch values are shown in the middle two figures, and two recordings with unknown pitch are shown in the bottom two.

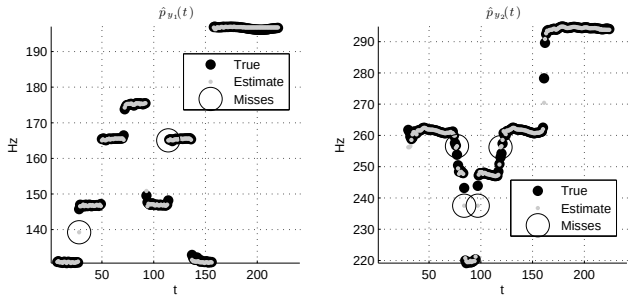

Figure 2: Results of nearest neighbor-based pitch tracking.

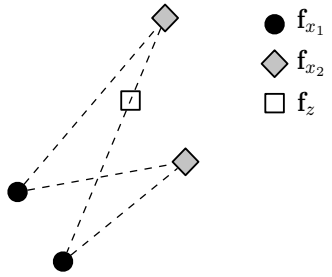

Figure 3: An example nearest subspace search.

obtain using the same pitch tracker we used to tag the training data, and the estimated pitch values $\hat{p}_{y_i}(t)$. In this case these values were $\mu = 0.02$ Hz and $\sigma = 1.1$ Hz respectively. The pitch estimates that were off by more than a semitone are highlighted by circles in figure 2. All such cases take place in transitionary sections where exact estimation of pitch is usually inaccurate. As a means of comparison, if we compare two well known pitch trackers, Praat [5] and Yin [4], operating on these segments we observe a mean and standard deviation of their estimates as $\mu = 0.1$ Hz and $\sigma = 1.2$ Hz, which leads to conjecture that this is a viable pitch tracking model.

3. MULTIPLE SOURCES

Let us know see how we can attack this problem if we observe a mixture of the two untagged recordings. Unlike the previous case, we will now have to deal with a more complex polyphonic pitch tracking problem. In the following two sections we will describe a straightforward search-based solution and a more efficient optimization-based approach that results in practical performance.

3.1. Nearest Subspace Search

Assume that we observe the mixture $z(t) = y_1(t) + y_2(t)$. We will also assume that $\mathbf{f}_z(t) = \alpha\mathbf{f}_{y_1}(t) + \beta\mathbf{f}_{y_2}(t)$, where the scalars α, β weigh the relative amplitude for each source. Although this equality is not strictly correct, it is a widely accepted approximation for various audio mixture problems. Given this setup, we will now generalize the previous process to:

$$\{\tau_1, \tau_2\} = \underset{\tau_1,\tau_2}{\operatorname{argmin}} \, \mathrm{D}\left[\mathbf{f}_z(t) || \alpha\mathbf{f}_{x_1}(\tau_1) + \beta\mathbf{f}_{x_2}(\tau_2)\right] \quad (3)$$

$$\hat{p}_{y_i}(t) \equiv p_{x_i}(\tau_i) \quad (4)$$

This kind of problem is known as the nearest subspace search and is substantially more complex than a nearest neighbor search. The objective in this problem is to explain each input as a weighted sum of two training data points, one from each source. A brute force approach to this problem is to construct all the possible lines that connect any pair between the spectra in $\mathbf{f}_{x_1}$ and in $\mathbf{f}_{x_2}$ and find which line passes the closest to the query point. A simple illustration of that is shown in figure 3, where the input $\mathbf{f}_z$ is best explained by only one specific pair of the training data $\mathbf{f}_{x_1}$ and $\mathbf{f}_{x_2}$. For N sources this search would involve constructing a $(N-1)$-dimensional subspace and measuring how close it is to the query point. As is clearly evident, the computational complexity of this approach can quickly get very demanding once we have a few thousand spectra or more than a couple of sources in our training data. There are efficient algorithms to resolve this problem [3], however the data sizes we intent to use for our purposes render even these approaches intractable, making the nearest subspace search an impractical approach for anything but trivial pitch tracking problems.

For the sake of demonstration we analyzed the mixture of $y_1(t)+y_2(t)$ from the data in figure 1. Even though this time we analyzed a mixture, the results were almost identical to the search on the isolated source recordings therefore we have a correct model, albeit one with a slow computational solution. As compared to the previous results the mean and standard deviation of the difference between the true and estimated values were $\mu = 0.1$ Hz and $\sigma = 2.2$ Hz. Just as before there are very few spectra that have been wrongly pitch tagged, and they all lie in transitional sections where pitch can be ambiguous.

3.2. An Efficient Algorithm

At this point we know that we have a potentially powerful approach to perform pitch tracking with mixtures, we do not however have an efficient method to complete this task. In this section we will use an efficient algorithm that can approximate the correct solution to the nearest subspace problem. We will use the approach shown in [2].

We will attempt to explain each input spectrum as a superposition of one spectrum from each of the two training sets. Additionally we want to consider the case where one of the instruments is not active, so explaining an input using only one spectrum from one of the sources is also desirable. Since we ruled out search techniques because of a high computational complexity, we will reinterpret this process as a sparse coding problem. Both of the preceding statements can be interpreted in the context of sparsity. We want to approximate the input spectrum as a sparse combination from each source training data, and in addition we want to use these sources sparsely. These two statements combine naturally and collapse to a simpler problem where we try to explain each input point as a sparse combination of all the training data combined. If only one source is active then a sparse solution will use only one spectrum, presumably one from the correct source, if two sources are active then an appropriate sparse solution is to use one spectrum from each source.

More concretely, the training data of all the sources are concatenated to form a single set of training spectral vectors $\mathbf{F} = [\mathbf{f}_{x_1}(1, \dots), \mathbf{f}_{x_2}(1, \dots), \cdots]$. We then assume that the input will be approximated as a linear superposition of that data, i.e.:

$$\mathbf{f}_z(t) \approx \mathbf{F} \cdot \mathbf{w}(t), \quad (5)$$

where the weight vector $\mathbf{w}(t)$ represents how much each training data point is used to approximate the current input. In order to ensure sparsity we impose a regularizing term that maximizes the ℓ_2-norm of $\mathbf{w}(t)$. Because both the training data and the input spectra

are normalized to sum to 1, this implies that the weight vector $\mathbf{w}(t)$ will also be normalized to 1. Due to the fact that all of its elements will be between 0 and 1, maximizing the ℓ_2-norm of that vector will result in growing as few as possible weights to be close to 1 and keeping most near 0, thus achieving sparsity. The overall problem is therefore to minimize:

$$\mathrm{D}\left[\mathbf{f}_z(t)||\mathbf{F}\cdot\mathbf{w}(t)\right]-\lambda\sum_i w_i(t)^2 \quad (6)$$

where λ is a scalar that indicates how strong we want the sparsity prior to be. For the case where $\mathrm{D}[\cdot||\cdot]$ is set to be the KL-divergence, this problem can be solved by initializing $\mathbf{w}(t)$ with random values and iterating through the following steps until convergence.

1. Get temporary estimate of weights:

$$\mathbf{w}(t)_{temp}=\mathbf{w}(t)\odot\left(\mathbf{F}\cdot\frac{\mathbf{f}_z(t)}{\mathbf{F}\cdot\mathbf{w}(t)}\right)$$

2. Impose sparsity:

$$\mathbf{w}(t)_{sparse}=\mathbf{w}(t)_{temp}+\lambda\frac{\mathbf{w}(t)_{temp}^2}{\sum\mathbf{w}(t)_{temp}^2}$$

3. Normalize to ensure proper scaling:

$$\mathbf{w}(t)_{new}=\frac{\mathbf{w}(t)_{sparse}}{\sum_i\mathbf{w}_i(t)_{sparse}}$$

where $\odot$ denotes element-wise multiplication, and the divisions and exponentiations are also element-wise. Throughout our simulations λ was set to values between 0.1 and 0.3 we trained for 20 iterations.

Once we obtain all the $\mathbf{w}(t)$ we find the training vectors with the largest weight values (one from each source), and use their associated pitch labels to explain the mixture. In the case where one source is inactive we will notice that the weights of its training spectra are uniformly close to zero in which case we assume that there is no pitched contribution from that source. For polyphonic sources (e.g. the piano) we will notice that multiple weights will be significantly larger than the rest, and these weights will correspond to the multiple pitches that are present in the input.

The outcome of running this algorithm on the mixture $y_1(t)+y_2(t)$ from the data in figure 1 is shown in figure 4. We can see that the results are qualitatively very similar to what we have obtained thus far. The mean and standard deviation of the difference between the true and estimated pitch values are $\mu=0.003$ Hz and $\sigma=2.05$ Hz. The pitch estimates that are more than a semitone off are once again taking place in transition points. For all practical purposes we estimated the pitch values of this segment with satisfactory accuracy. In terms of performance it took 0.45 sec to pitch track $y_i(t)$ in order to obtain the true values, 0.12 sec to obtain their pitch with the nearest neighbor search on the separated recordings, 64.5 sec to do the exhaustive nearest subset search on the mixture, and 0.68 sec to estimate the pitch labels from the mixture with the proposed method [1].

3.3. The Details

In this section we will supply some extra information that was withheld until now for the sake of clarity. For the spectral vectors we use the warped DFT transform [6] and stretch the spacing of the low frequencies while compressing the high ones, akin to a constant-Q transform, in order to assist accurate comparisons between the input and the training data. The result of this is slightly better performance, although not dramatically so as compared to operating on regular magnitude spectrograms. For all cases the sample rate was 22,050Hz and we used a 1024pt DFT with a hop size of 256pt, and a Hann window for the STFT analysis.

As one might imagine when presented with training data not all of its spectral vectors will correspond to a pitched section. In order to reduce the size of the problem, training spectra that are not pitched are not considered in our search. An alternative approach is to use some unpitched spectra and carry their non-pitch labels.

[1] All times were measured using unoptimized MATLAB code on a 1.86GHz Core 2 Duo CPU.

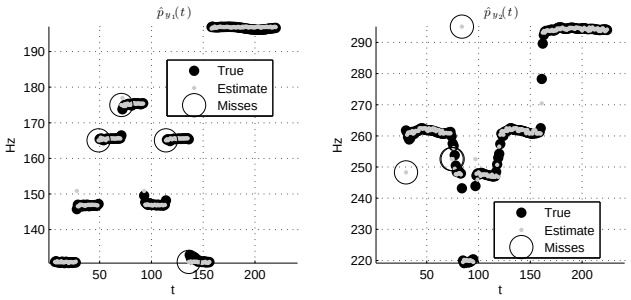

Figure 4: Estimated and true pitch values for when applying the proposed approach on a mixture of two sounds.

4. EXPERIMENTS

In this section we will examine the performance of our approach on some more challenging examples. More specifically we will consider the data from a multichannel woodwind quintet recording [7]. Since all tracks are available separately we can easily obtain pitch values using traditional pitch tracking and use that in order to evaluate our approach. For training data we used 7m41s of pitch tagged content, and we attempted to pitch track 1m10s of polyphonic mixtures of a duet, trio, quartet and quintet. After discarding unpitched parts this amounted to about 10,000 training spectral vectors per source, which were used on 6,000 input spectra to perform pitch tracking. Using the proposed approach we obtained the results shown in figure 5. For the case of the duet the most common errors were octave errors (peaks at a ratio of 0.5 and 2). As we introduced additional sources we see that the performance of the pitch tracking gradually degrades.

A short section of the pitch tracks in a flute part of the duet recording is shown in top plot of figure 6. In it one can see that the most common error is the flipping octaves, as well as mistakes in transient sections. The octave problem can be easily minimized by performing median filtering in the pitch tracking. Doing so we obtain the results in the middle plot of the same figure, where the octave errors are minimized but the transient errors are now more prominent. Additionally, if we are interested in note transcription and not exact pitch tracking, we can consolidate the weights of training spectra that correspond to the same note, obtain a per-note weight and use these weights to decide which note is being active at any time. The result of that is shown in the bottom plot of figure 6, where we see that the fine frequency detail is gone and the pitches are quantized to semitones. Obviously these are simple approaches, more sophisticated tools such as Kalman filters can be easily employed to minimize many of the temporal inconsistencies, but that

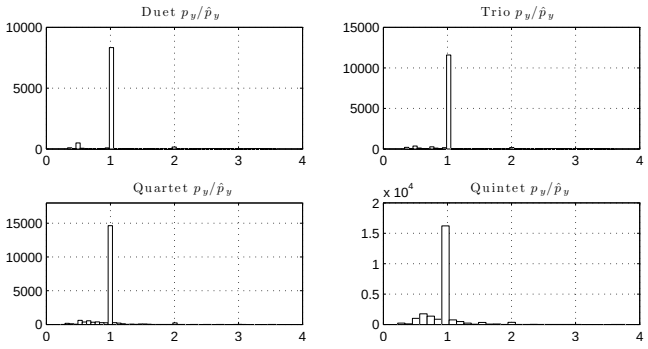

Figure 5: Histogram of the ratio between the true and estimated pitches for increasingly denser mixtures.

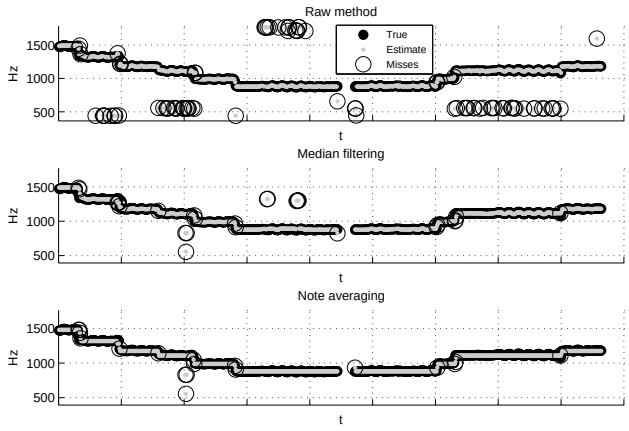

Figure 6: A small section of the flute part in the duet mixture. The top plot shows the raw output, the middle plot show the results after applying a median filter to the output pitch track, and the bottom plot finds the note which includes the most active training spectra.

study is outside the scope of this paper.

The mean and standard deviation of the estimation error as well as the run time[2] for mixtures ranging from a solo to a quintet are shown in the following table. The error statistics are shown for both pitch estimation in terms of Hz values, and for note estimation performed as described above.

Mix	μ	σ	Time
Solo	-4.41 Hz 0.13 semitones	41.57 Hz 1.46 semitones	20 sec
Duet	-18.51 Hz -0.08 semitones	89.51 Hz 1.56 semitones	37 sec
Trio	-32.65 Hz -0.15 semitones	109.96 Hz 1.57 semitones	49 sec
Quartet	-28.69 Hz -0.61 semitones	91.65 Hz 2.58 semitones	60 sec
Quintet	-42.79 Hz -0.87 semitones	112.69 Hz 3.32 semitones	114 sec

[2]MATLAB code measured on a 2.98GHz 6-core Xeon

5. CONCLUSIONS

In this paper we presented a learning-based approach to pitch tracking of mixtures of signals. We showed that it is plausible to efficiently use large pitch-tagged training recordings in order to learn how to recognize pitch in mixtures. The primary advantages of this approach is a very compact computational core which is not constrained by model assumptions. As long as the training data is appropriately tagged this approach will work regardless of the peculiarities in the present sources (e.g. quasi-pitched and inharmonic sources). This allows us to perform detection of perceived pitch as opposed to the more traditional measurement of absolute measured pitch. If the training data is properly tagged this approach can provide additional information besides pitch, such as tone quality, vowel character, playing style, etc.

What we present here is a simple beginning of a pitch tracking system. The efficiency of this model and its graceful behavior with mixtures leads us to believe that it holds promise for further extension towards a polyphonic transcription system. Especially with proper use of higher-level priors that one can either impose as a user, or preferably learn from recordings, a lot of the errors that we observe (octave and transient issues) can be minimized to a good degree. Open questions such as the impact of the quality of the training data are still relatively unexplored, and we reserve exploring these effects in future work.

6. ACKNOWLEDGMENT

The author would like to thank Mert Bay of the University of Illinois at Urbana-Champaign for providing and segmenting the quintet recordings used in the experiments section.

7. REFERENCES

[1] Klapuri, A. Signal Processing Methods for the Automatic Transcription of Music, Doctoral Dissertation, Tampere University of Technology, Finland 2004.

[2] Smaragdis, P. Approximate nearest-subspace representations for sound mixtures, in proc. ICASSP 2011.

[3] Basri, R., T. Hassner and L. Zelnik-Manor. 2007. Approximate Nearest Subspace Search with Applications to Pattern Recognition, CVPR'07.

[4] de Cheveigné, A. and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. The Journal of the Acoustical Society of America, 111:1917, 2002

[5] Boersma, P. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound, in Proc. of the Institute of Phonetic Sciences, University of Amsterdam, 1993, vol. 17, pp. 97 110.

[6] Makur, A. and Mitra, S.K. Warped discrete-Fourier transform: Theory and applications, in IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 48:9, Sep 2001.

[7] Bay, M., A.F. Ehmann, and J.S. Downie. Evaluation of multiple-F0 estimation and tracking systems, in Proc. of ISMIR, 2009.