A MULTI-PITCH TRACKING SYSTEM (MIREX 2009)

Zhiyao Duan
Northwestern University
zhiyaoduan00@gmail.com

Jinyu Han
Northwestern University
jinyuhan@gmail.com

Bryan Pardo
Northwestern University
pardo@northwestern.edu

ABSTRACT

This extended abstract describes the system we submit for MIREX 2009 task "Multiple Fundamental Frequency Estimation and Tracking". The task includes three subtasks: 1) Multi-pitch Estimation (MPE) at frame-level, 2) note events detection and 3) pitch tracking for each source. Our system consists of two parts: 1) MPE in each time frame and 2) pitch trajectory formation from these pitch estimates. We submit two versions of the system. The first one implements the whole system and does all the three subtasks. The second one only implements the MPE part and does the first subtask. For the first subtask the two versions output different results, since the first version refines MPE results with tracking results.

1. INTRODUCTION

Our system addresses the Multi-pitch Estimation and Tracking problem in two stages, as shown in Figure . We submit two versions of the system. The first one implements the whole system and does all the three subtasks. The second one only implements the MPE part and does the first subtask. For the first subtask the two versions output different results, since the first version refines MPE results with tracking results.

Now we briefly describe each part of the whole system. For a detailed description, please refer to [1].

2. MULTI-PITCH ESTIMATION

2.1 Multi-pitch Estimation In A Single Frame

We view the Multi-pitch Estimation (MPE) problem (given polyphony N) as a Maximum Likelihood parameter estimation problem in the frequency domain. The parameters to be estimated are the pitches $\boldsymbol{\theta} = \{F_0^1, \cdots, F_0^N\}$, and the observation is the spectrum, which is represented as spectral *peaks* and *non-peak regions*. Peaks are detected using the peak detection algorithm in [2]. The non-peak region is defined as the set of frequencies falling more than a quarter-tone from any observed peak. We try to find the set of F0s with harmonics that maximize the probability of the occurrence of observed peaks, and minimize the probability that they have harmonics in non-peak regions.

Thus, the likelihood function can be defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{peak}}(\boldsymbol{\theta}) \cdot \mathcal{L}_{\text{non-peak region}}(\boldsymbol{\theta}) \qquad (1)$$

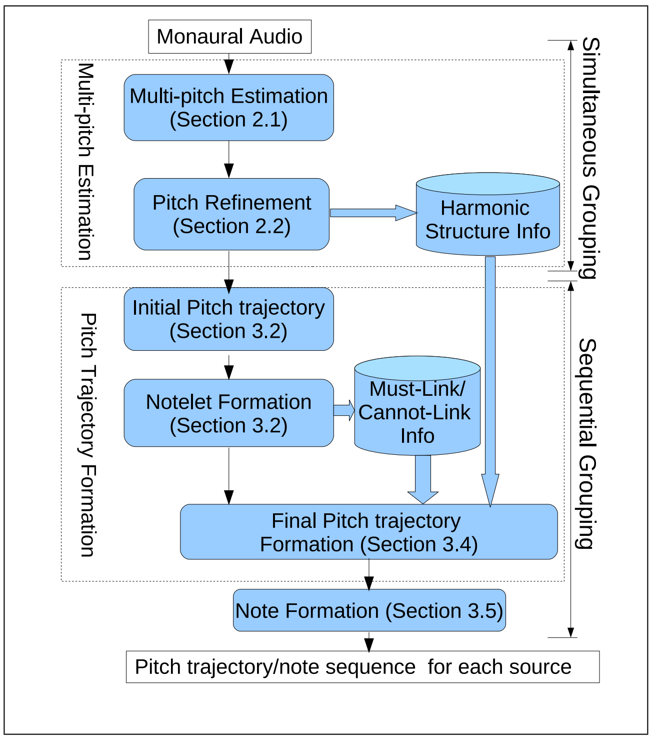
Monaural Audio
Multi-pitch Estimation (Section 2.1)
Pitch Refinement (Section 2.2)
Harmonic Structure Info
Initial Pitch trajectory (Section 3.2)
Notelet Formation (Section 3.2)
Must-Link/ Cannot-Link Info
Final Pitch trajectory Formation (Section 3.4)
Note Formation (Section 3.5)
Pitch trajectory/note sequence for each source
Multi-pitch Estimation
Pitch Trajectory Formation
Simultaneous Grouping
Sequential Grouping

Figure 1. System overview.

For the detailed formulation of the likelihood function, and the description about how we learn the model parameters from training data, please refer to [1].

In Eq. (1), the search space of the maximum likelihood solution $\hat{\boldsymbol{\theta}}$ is combinatorially explosive of the polyphony. We avoid this problem with a greedy search strategy. We start from an empty set $\hat{\boldsymbol{\theta}}^0$. In each iteration, we add a F0 estimate to $\hat{\boldsymbol{\theta}}^n$ to get a new set $\hat{\boldsymbol{\theta}}^{n+1}$ that gets maximum likelihood among all the possible values of the newly added F0. It is found that $\mathcal{L}(\hat{\boldsymbol{\theta}}^n)$ increases with n. This iteration terminates when $n = N$, the given polyphony.

If the polyphony is not given, we need to decide when to terminate. To do so, we propose a simple threshold-based method. The minimum number of F0s that can achieve the likelihood excess to a threshold is returned as the polyphony estimate:

$$\begin{aligned} N &= \min_{1 \le n \le M} n, \\ s.t. \quad & \Delta(n) \ge T \cdot \Delta(M) \end{aligned} \qquad (2)$$

where $\Delta(n) = \mathcal{L}(\hat{\boldsymbol{\theta}}^n) - \mathcal{L}(\hat{\boldsymbol{\theta}}^1)$ is the maximum increase of likelihood that could be achieved when the polyphony is set to be n. M is the maximum allowed polyphony; T is a learned threshold. For all experiments in this paper, the maximum polyphony M is set to 9. T is empirically determined to be 0.88. This polyphony estimation method is found to work well on a large data set containing both music pieces and musical chords with different polyphony.

2.2 Refine Pitch Estimates Using Neighboring Frames

We propose a multi-pitch refinement method using estimates in neighboring frames: For each frame t, we build a weighted histogram in the frequency domain, where each bin corresponds to a semitone in the pitch range. Then, a triangular weighting function centered at t is imposed on a neighborhood of t, whose radius is r frames. The refined polyphony estimate N is calculated as the weighted average of polyphony estimates in all the frames in this neighborhood. Then N bins with the highest histogram values are selected to reconstruct refined pitch estimates. For each of these bins, if there is an original pitch estimate in frame t that falls inside this bin, the original pitch estimate is used as the refined pitch estimate directly. Otherwise, the refined pitch estimate is calculated as the weighted average frequency of all the pitch estimates in this neighborhood that fall inside this bin. In our system, the radius r is set to 9 frames. After this refinement, a number of inconsistent estimation errors are removed.

3. PITCH TRAJECTORY FORMATION

Given pitch estimates in all frames, we view pitch trajectory formation as a constrained clustering problem, where each *pitch trajectory* corresponds to a cluster.

Constrained clustering [3,4] is a class of semi-supervised learning algorithms. Constraints can be imposed the instance level, where there are two basic forms: *must-link* and *cannot-link*. A must-link (cannot-link) specifies that two instances should (not) be assigned to the same cluster.

3.1 Multi-pitch Tracking as Clustering

For our pitch trajectory formation problem, we adopt these two constraints (as described in Section 3.2). We then formulate the clustering problem to minimize the intra-class distance J, as the K-means algorithm does:

$$J = \sum_{k=1}^{K} \sum_{x_i \in T_k} \|\mathbf{x}_i - \mathbf{c}_k\|^2 \qquad (3)$$

where K is the number of pitch trajectories; $\mathbf{x}_i$ is a feature vector in trajectory T_k and $\mathbf{c}_k$ is the mean feature vector in trajectory T_k; $\|\cdot\|$ denotes the Euclidean distance.

Then we propose an iterative greedy algorithm that minimizes Eq. (3) in Section 3.4.

3.2 Initial Pitch Trajectory and Notelet Formation

To get the initial clustering with which our algorithm starts, we simply sort pitches in each frame from high to low and assign labels from 1 to K. This is possible, since there are at most K pitches in each frame.

Then must-links are imposed on similar pitches that are in adjacent frames *and* have the same initial trajectory label. The maximal must-link difference between pitches in adjacent frames is set to 0.3 semitones (30 cents). Pitches connected by must-links form a short trajectory, which we call a *notelet*, since it is supposed to be some part of a note. Once notelets are formed, cannot-links are imposed between all pitches in two notelets that overlap more than 30ms. We say that two such notelets are in a *cannot-link relation*. We allow the 30ms overlap within a melodic line as it may be reverberation or ringing of a string. We chose conservative values for these parameters to ensure that they are reasonable for common real-world scenarios.

3.3 Harmonic Structure

Feature vectors in Eq. (3) should have the property that they are similar in the same trajectory and far in different trajectories. Harmonic structure has been proven to be a good choice for harmonic instrumental sources, which is defined as the vector of relative amplitudes of harmonics of a pitch [2]. Harmonic structures of the same instrument are similar, even if their pitches and loudness are different. On the other hand, different instruments usually have very different harmonic structures.

We calculate harmonic structure as follows: First, harmonics of each pitch are found from spectral peaks. For overlapping harmonics of different pitches, the peak likelihood in Eq. (1) is used to distribute energy to each pitch. Harmonic structures are then normalized to have the same total energy. The first fifty harmonics are used here.

3.4 Final Pitch Trajectory Formation

From the initial pitch trajectory, we now consider re-assigning notelets to different trajectories to minimize Eq. (3).

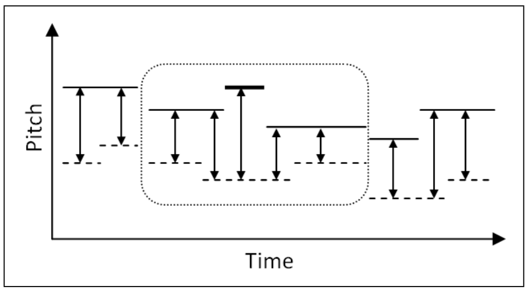

Figure 2. Illustration of a swap-set (the rounded rectangle) for a notelet (the bold solid line). Solid and dashed lines represent notelets in trajectory T_k and T_l, respectively. Cannot-link relations are indicated by arrows.

Suppose we want to change the trajectory label of a notelet $\mathfrak{n}$ from T_k to T_l. We cannot do this in isolation, since there may be a notelet in T_l that overlaps $\mathfrak{n}$ and we assume monophonic pitch trajectories. We could simply swap the trajectories for two overlapping notelets. This, however may cause a chain reaction, since the swap may cause new overlaps within a trajectory. Instead, we select two trajectories and pick a notelet $\mathfrak{n}$ from one of the trajectories. We then find all notelets in these two trajectories,

```
For each notelet $\mathfrak{n}$, with trajectory $T_k$
  $J_0$ = cost of current trajectory assignments (Eq. (3))
  $J_{best}$ = $J_0$
  For each trajectory $T_l$, such that $T_l \neq T_k$
    Find the swap-set between $T_l$ and $T_k$ containing $\mathfrak{n}$
    $J$ = (Eq. (3)) if we swap every notelet in the swap-set
    If $J < J_{best}$, $J_{best} = J$, End
  End
  If $J_{best} < J_0$, Perform swap that produces $J_{best}$, End
End
```

Table 1. The pitch trajectory formation algorithm.

T_k and T_l that connect to $\mathfrak{n}$ via a path of cannot-link relations (defined in Section 3.2). We call this the *swap-set*, as illustrated in Figure 2. These are the notelets affected by a potential trajectory swap between two notelets. All notelets in a swap set are swapped together, rather than individually, and the new trajectory are evaluated with the cost function in Eq. (3).

Table 1 describes the process we use to swap trajectories for notelets until the trajectories of all notelets reach fixed points. In our experiment, this usually takes 3 to 4 rounds (where a round is a traversal of all notelets).

3.5 Note Formation

After pitch trajectories are formed, we form notes in each trajectory from the notelets. Two notelets are considered to be in the same note if the time gap between them is less than 100ms and their frequency difference is less than 0.3 semitone. Then the pitches in the gap are reconstructed using the average frequency of the note. Notes of length less than 100ms are considered spurious and removed. The 0.3 semitone parameter is the same as the one in imposing must-links. The 100ms parameter is set without tuning to adapt to the tempo and note lengths of the test music.

4. RESULTS

5. CONCLUSION

In this paper, we briefly described the system we submit to MIREX 2009 "Multiple Fundamental Frequency Estimation and Tracking" task. Our system first estimated pithes and polyphony in each time frame. Then pitch trajectories were formed by constrained clustering pitch estimates across frames.

6. REFERENCES

[1] Z. Duan, J. Han and B. Pardo, "Harmonically informed multi-pitch tracking," *Proc. ISMIR*, 2009.

[2] Z. Duan, Y. Zhang, C. Zhang and Z. Shi: "Unsupervised single-channel music source separation by average harmonic structure modeling," *IEEE Trans. Audio Speech Language Process.*, Vol. 16, No. 4, pp. 766-778, 2008.

[3] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl: "Constrained K-means clustering with background knowledge," *Proc. ICML*, 2001.

[4] I. Davidson and S. S. Ravi: "Clustering with constraints: feasibility issues and the k-means algorithm," *Proc. SDM*, 2005.