Pitch tracking in polyphonic audio by clustering local fundamental frequency estimates

Martín Rocamora,[1] and Pablo Cancela[1]

[1] Universidad de la República, FING, IIE, Departamento de Procesamiento de Señales
Montevideo, Julio Herrera y Reissig 565, C.P. 11200, Uruguay

rocamora@fing.edu.uy, pcancela@fing.edu.uy

ABSTRACT

A novel way of performing pitch tracking by means of clustering local fundamental frequency (f0) candidates is described. The technique is based on an existing pitch salience representation for polyphonic music called F0gram which relies on the Fan Chirp Transform [1]. The grouping is performed by applying a Spectral Clustering method, since it can handle filiform shapes such as pitch contours. The approach seems appealing since many sound sources can be tracked simultaneously and the number of contours and sources is derived from the data. Results of a melody detection evaluation indicate the introduced method is promising, despite that various aspects of the technique deserve further work.

1 INTRODUCTION

Multiple fundamental frequency (f0) estimation is one of the most important problems in music signal analysis and constitutes a fundamental step in several applications such as melody extraction, sound source identification and separation. In our previous work [1] the Fan Chirp Transform (FChT) was applied to polyphonic music analysis, a technique based on decomposing the audio signal into harmonically related chirps. In addition, a pitch salience representation for music analysis called F0gram was proposed that provides a set of local fundamental frequency candidates together with a pitch change rate estimate for each of them. To continue the analysis temporal integration of local pitch candidates has to be performed, which is the problem tackled herein. There is a vast amount of research on pitch tracking in audio, often comprising an initial frame by frame f0 estimation followed by formation of pitch contours exploiting estimates continuity over time. Techniques such as dynamic programming, linear prediction, hidden Markov models, among many others (see [2] for a review), were applied to the temporal tracking.

The herein proposed technique for pitch contours formation does not involve a classical temporal tracking algorithm. Instead the pitch tracking is performed by unsupervised clustering of F0gram peaks. The spec-

tral clustering method [3] is selected for this task as it imposes no assumption of convex clusters, thus being suitable for filiform shapes such as pitch contours. The pitch change rate estimates provided by the FChT analysis play an important role in the definition of similarity between pitch candidates. The clustering is carried out within overlapped observation windows corresponding to several signal frames. Then contours are formed by simply joining clusters that share elements. This short-term two-stage processing proved to be more robust than aiming a straightforward long-term clustering.

There are very few applications of spectral clustering for tracking a sound source. Blind one-microphone separation of two speakers is tackled in [4] as a segmentation of the spectrogram. A method is proposed to learn similarity matrices from labeled datasets. Several grouping cues are applied such as time-frequency continuity and harmonicity based. A simple multiple pitch estimation algorithm is part of the feature extraction. The mixing conditions are very restrictive (equal strength and no reverberation). Performance is assessed through a few separation experiments.

Clustering of spectral peaks is applied in [5], for partial tracking and source formation. Connecting peaks over time to form partials and grouping them to form sound sources is performed simultaneously. The problem is modeled as a weighted undirected graph where the nodes are the peaks of the magnitude spectrum. The edge weight between nodes is a function of frequency and amplitude proximity (temporal tracking) and a harmonicity measure (source formation). Clustering of peaks across frequency and time is carried out for windows of an integer number of frames ($\sim$ 150 ms) using a spectral clustering method. Clusters from different windows are not connected for temporal continuation. The two more compact clusters of each window are selected as the predominant sound source.

The rest of this document is organized as follows. In the next section the application of the FChT to build a pitch salience representation is briefly discussed. Section 3 summarizes the fundamental aspects of the Spectral Clustering methods. Section 4 describes the proposed algorithm for pitch contours formation. Experimental results are presented in section 5. The paper ends with a critical discussion on the present work.

2 PITCH SALIENCE COMPUTATION

The Short Time Fourier Transform is the standard method for time-frequency analysis. In this representation the signal is supposed to be stationary within the analysis frame. However, music audio signals such as a singing voice typically exhibit rapid pitch fluctuations that are troublesome for the analysis.

A different approach to perform the analysis is considering the projection over frequency modulated sinusoids (chirps), in order to obtain a non-Cartesian tiling of the time-frequency plane. The modulation rate of the chirp can be selected in order to closely match the pitch change rate of the musical sound source. Among the chirp based transforms, the FChT offers optimal resolution simultaneously for all the partials of a harmonic chirp (harmonically related chirps). This is well suited for music analysis because many sounds have an harmonic structure. The FChT can be formulated as [1],

$$X_w(f,\alpha)=\int_{-\infty}^{\infty} x(t)\; w(\phi_\alpha(t))\; \phi'_\alpha(t)\; e^{-j2\pi f \phi_\alpha(t)} dt \quad (1)$$

where $\phi_\alpha(t) = (1 + \frac{1}{2}\alpha t)\, t$, is a time warping function and $w(t)$ stands for a time limited window, such as Hann. Notice that by the variable change $\tau = \phi_\alpha(t)$, the formulation can be regarded as the Fourier Transform (FT) of a time warped version of the signal $x(t)$, which enables an efficient implementation based on the FFT. Given the previous formulation, the FChT representation of a harmonic linear chirp is composed of deltas convolved with the FT of the window, provided the appropriate chirp rate α is applied in the warping.

Computing the FChT for consecutive short time signal frames a time-frequency representation in the form of a spectrogram can be built. For polyphonic music analysis the approach followed in [1] is to compute several FChT instances with different α values. This produces a multidimensional representation made up of various time-frequency planes.

The time-frequency representation described above can be applied to obtain a detailed description of the melodic content of an audio signal, as proposed in [1]. Pitch salience is computed for each signal frame in a certain range of fundamental frequency values. Given S_α the power spectrum produced by the FChT for a chirp rate value α, salience of fundamental frequency f_0 is obtained by gathering the log-spectrum at the positions of the corresponding harmonics, $\rho(f_0,\alpha) = \frac{1}{n_H}\sum_{i=1}^{n_H} log|S_\alpha(if_0)|$, where n_H is the number of harmonics that are supposed to lie within the analysis bandwidth. Some postprocessing steps are carried out in order to attenuate spurious peaks at multiples and submultiples of the true pitches, and to balance different fundamental frequency regions [1]. Finally, for each f0 in the grid the highest salience value is selected among the different available α values. In this way an F0gram is obtained, that shows the evolution of pitch for all the harmonic sounds in the signal. An example of an F0gram is depicted in Figure 1 for an audio fragment that is used throughout the paper.

3 SPECTRAL CLUSTERING

The goal of clustering can be stated as dividing data points into groups such that points in the same cluster are similar and points in different clusters are dissimilar. An useful way of representing the data is in the form of a similarity graph, each vertex corresponding to a data point. Two vertices of the graph are connected if their similarity is above certain threshold, and the edge between them is weighted by their similarity value. In

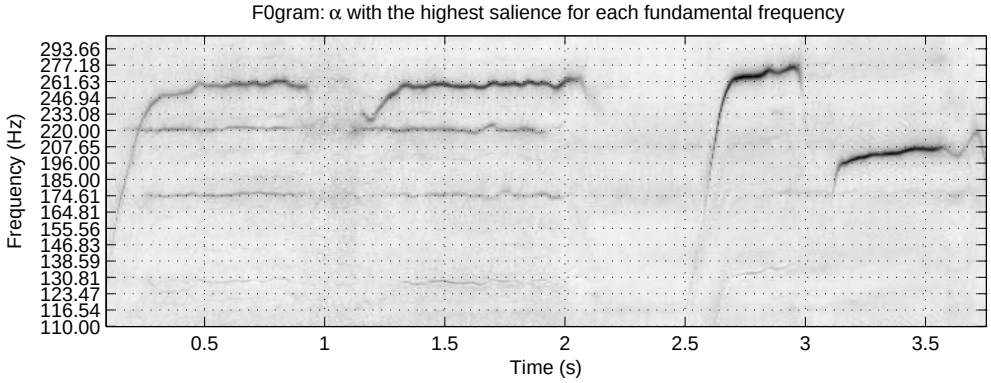

Figure 1: F0gram for a fragment of the audio file pop1 (used throughout the paper) from the MIREX melody test set. It consist of three simultaneous singing voices followed by a single voice, with a rather soft accompaniment.

terms of the graph representation, the aim of clustering is to find a partition of the graph such that different groups are connected by very low weights whereas edges within a group have high weights.

The simplest way to construct a partition is to solve the mincut problem. Given a number of clusters k, it consist in finding a partition $A_1, \ldots, A_k$ that minimizes

$$\text{cut}(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \bar{A}_i), \qquad (2)$$

where $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$ is the sum of weights of vertices connecting partitions A and B, and $\bar{A}$ stands for the complement of A. This corresponds to finding a partition such that points in different clusters are dissimilar to each other. The problem with this approach is that it often separates one individual vertex from the rest of the graph. An effective way of avoiding too small clusters is to minimize the Ncut function,

$$\text{Ncut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)}, \qquad (3)$$

where $\text{vol}(A) = \sum_{i \in A} d_i$ is the sum of the degree of vertices in A. The degree of a vertex is defined as $d_i = \sum_{j=1}^{n} w_{ij}$, so $\text{vol}(A)$ measures the size of A in terms of the sum of weights of those edges attached to their vertices. The Ncut criterion minimizes the between cluster similarity (in the same way as mincut), but it also implements a maximization of the within cluster similarities. Notice that the within cluster similarity can be expressed as, $W(A, A) = \text{vol}(A) - \text{cut}(A, \bar{A})$ [6]. In this way the Ncut criterion implements both objectives: to minimize the between cluster similarity, if $\text{cut}(A, \bar{A})$ is small, and to maximize the within cluster similarity, if $\text{vol}(A)$ is large and $\text{cut}(A, \bar{A})$ is small.

The mincut problem can be solved efficiently. However with the normalization term introduced by Ncut it becomes NP hard. Spectral clustering is a way to solve relaxed versions of this type of problems. Relaxing Ncut leads to the normalized spectral clustering algorithm. It can be shown [6] that finding a partition of a graph with n vertices into k clusters by minimizing Ncut, is equivalent to finding k indicator vectors $h_j = (h_{1j}, \ldots, h_{nj})'$ with $j = 1, \ldots, k$ of the form, $h_{ij} = 1/\text{vol}(A_j)$ if vertex $v_i \in A_j$ and zero otherwise. In this way, the elements of the indicator vectors point out to which cluster belongs each graph vertex. This problem is still NP hard, but can be relaxed by allowing the elements of the indicator vectors to take, instead of two discrete values, any arbitrary value in $\mathbb{R}$. The solution to this relaxed problem corresponds to the first k generalized eigenvectors of $(D - W)\, u = \lambda\, D\, u$, where D is an n by n diagonal matrix with the degrees of the graph vertices $d_1, \ldots, d_n$ on the diagonal, and $W = (w_{ij})_{i,j=1\ldots n}$ is the matrix of graph weights.

The vectors u of the solution are real-valued due to the relaxation and should be transformed to discrete indicator vectors to obtain a partition of the graph. To do this, each eigenvalue can be used in turn to bipartition the graph recursively by finding the splitting point such that Ncut is minimized [3]. However, this heuristic may be too simple in some cases and most spectral clustering algorithms consider the coordinates of the eigenvectors as points in $\mathbb{R}^k$ and cluster them using an algorithm such as k-means [6]. The change of representation from the original data points to the eigenvector coordinates enhances the cluster structure of the data, so this last clustering step should be very simple if the original data contains well defined clusters. In the ideal case of completely separated clusters the eigenvectors are piecewise constant so all the points belonging to the same cluster are mapped to exactly the same point.

Finally, the algorithm can be summarized as [6],

input: similarity matrix $S \in \mathbb{R}^{nxn}$, number of clusters k

steps:

1. build a similarity graph using matrix S
2. compute the unnormalized Laplacian of the graph $L = (D - W)$
3. compute the first k generalized eigenvectors of $(D - W)\, u = \lambda\, D\, u$

4. consider the eigenvectors $u_1, \ldots, u_k$ as columns of a matrix $U \in \mathbb{R}^{nxk}$
5. consider the vectors $y_i \in \mathbb{R}^k \quad i = 1, \ldots, n$ corresponding to the rows of U
6. cluster the points (y_i) in $\mathbb{R}^k$ with k-means into clusters $C_1, \ldots, C_k$

output: clusters $A_1, \ldots, A_k$ / $A_i = \{j \mid y_j \in C_i\}$.

4 PITCH CONTOURS FORMATION

In order to apply the spectral clustering algorithm to the formation of pitch contours several aspects must be defined. In particular, the construction of the graph involves deciding which vertices are connected. Then, a similarity function has to be designed such that it induces meaningful local neighbours. Besides, an effective strategy has to be adopted to estimate the number of clusters. In what follows, each of these issues are discussed and the proposed algorithm is described.

4.1 Graph construction

Constructing the similarity graph is not a trivial task and constitutes a key factor in spectral clustering performance. Different alternatives exist for the type of graph, such as k-nearest neighbor, ϵ-neighborhood or fully connected graphs, which behave rather differently. Unfortunately, barely any theoretical results are known to guide this choice and to select graph parameters [6]. A general criteria is that the resulting graph should be fully connected or at least should contain significantly fewer connected components than the clusters we want to detect. Otherwise, the algorithm will trivially return connected components as clusters.

To include information on temporal proximity a local fixed neighborhood is defined, such that f0 candidates at a certain time frame are connected only to candidates in their vicinity of a few frames (e.g. two neighbor frames on each side). In this way the graph is in principle fully connected, as can be seen in Figure 2, and resulting connected components are determined by similarity between vertices. Two candidates distant in time may nevertheless belong to the same cluster by their similarity to intermediate peaks. Note that in Figure 2 only one neighbor frame on each side is taken into account to link peaks. In this case, if a peak is missing the given contour may be disconnected. For this reason, a local neighbourhood of two or three frames on each side is preferred. Similarity of not connected components is set to zero, so a sparse similarity matrix is obtained.

In addition, a contour should not contain more than one f0 candidate per frame. To favour this, candidates in the same frame are not connected. Specifying cannot-link constrains of this type is a common approach for semi-supervised clustering [7]. However, this not strictly prohibits two simultaneous peaks to be grouped in the same cluster if their similarity to neighbor candidates is high. For this reason, clusters should be further processed to detect this situation and select the most appropriate candidate in case of collisions.

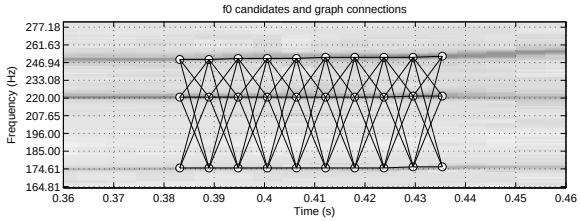

Figure 2: Graph connections considering only one neighbor frame on each side for an observation window of 10 frames. The resulting graph is fully connected.

4.2 Similarity measure

To define a similarity measure between F0gram peaks it is reasonable to base it on the assumption of slow variation of pitch contours in terms of fundamental frequency and salience (as defined in section 2).

Fundamental frequency distance between two graph vertices v_i and v_j may be better expressed in a logarithmic scale, that is as a fraction of semitones. To do this, pitch value of a vertex is expressed as the corresponding index in the logarithmically spaced grid used for pitch salience computation[1]. Then, this can be converted to a similarity value $s_{f0}(v_i, v_j) \in (0, 1]$ using a Gaussian radial basis function,

$$s_{f0}(v_i, v_j) = e^{-\frac{d^2_{f0}(v_i, v_j)}{\sigma^2_{f0}}} \tag{4}$$

where $d_{f0}(v_i, v_j) = |f0_i - f0_j|$ stands for pitch distance and σ_{f0} is a parameter that must be set which defines the width of local neighborhoods. In a similar way, a similarity function can be defined that accounts for salience proximity. To combine both similarity functions they can be multiplied, as in [5].

Although this approach was implemented and proved to work in several cases, the similarity measure has some shortcomings. Pitch based similarity is not able to discriminate contours that intersect. In this case, salience may be useful but it also has some drawbacks. For instance, points that are not so near in frequency and should be grouped apart, may be brought together by their salience similarity. This suggest the need for a more appropriate way of combining similarity values.

A significant performance improvement was obtained by combining the pitch value of the candidates and the chirp rates provided by the FChT. The chirp rate can be regarded as a local estimation of the pitch change rate. Thus, the pitch value of the next point in the contour can be predicted as $\overrightarrow{f0}^k_i = f0^k_i\,(1+\alpha^k_i\,\Delta t)$, where $f0^k_i$ and α^k_i are the pitch and chirp rate values, i and k are the candidate and frame indexes respectively, and Δt is the time interval between consecutive signal frames. Figure 3 depicts most prominent f0 candidates and their predictions for a short region of the example. Note that there are some spurious peaks in the vicin-

[1] In which a 16th semitone division is used (192 points per octave).

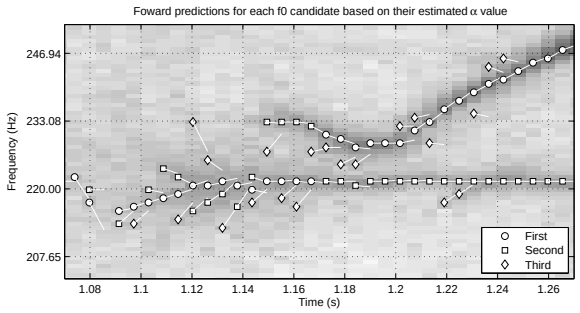

Figure 3: Forward predictions of the three most prominent f0 candidates for a short interval of the example. Although the clusters seem to emerge quite defined, spurious peaks may mislead the grouping. This can be improved if a backward prediction is also considered.

ity of a true pitch contour whose estimate lie close to a member of the contour and can lead to an incorrect grouping. A more robust similarity measure can be obtained by combining mutual predictions between pitch candidates. This is done by computing for each candidate also a backward prediction $\overleftarrow{f0}$ in the same way as before. Then, distance among two candidates v_i^k and v_j^{k+1} is obtained by averaging distances between their actual pitch values and their mutual predictions,

$$d_{f0}(v_i^k, v_j^{k+1}) = \frac{1}{2}\left[|f0_i^k - \overleftarrow{f0}_j^{k+1}| + |\overrightarrow{f0}_i^k - f0_j^{k+1}|\right] \quad (5)$$

Using this mutual distance measure the similarity function is defined as in Equation (4). Additionally, the same reasoning can be extended to compute forward and backward predictions for two or three consecutive frames. This similarity values are used as graph weights for candidates in their temporal proximity.

Still remains to set the value σ_{f0}, which plays the role of determining the actual value assigned to points in the vicinity and to outlying points. Self tunning sigma for each pair of data points was tested based on the distance to the k-th nearest neighbor of each point, as proposed in [8]. This approach can handle cluster with different scales, but applied to this particular problem it frequently grouped noisy peaks far apart from each other. It turned out that, given the filiform shape of clusters that are to be detected, a fixed value for σ_{f0} was more effective. Since pitch predictions become less reliable as the time interval grows, a more restrictive value for σ_{f0} is used for measuring similarity to points at the second and third consecutive frame (reported results correspond to $\sigma_{f0}^1 = 0.8$, $\sigma_{f0}^2 = 0.4$).

Figures 4 and 5 show two different examples of the local clustering, which correspond to three well-defined clusters and two clusters with spurious peaks. An observation window of 10 signal frames is used and the three most prominent F0gram peaks are considered. A neighborhood of two frames on each side is used. Similarity matrix is sorted according to the detected clusters, producing a sparse band diagonal matrix, where clusters can be visually identified as continuous bands.

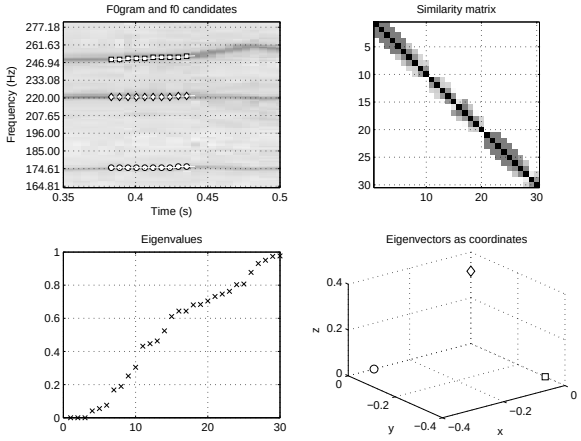

Figure 4: Local clustering for a short time interval of the audio example. Similarity matrix, eigenvalues and eigenvectors as coordinates are depicted. Three well-defined clusters can be identified in the data, as well as the corresponding bands in the similarity matrix. The multiplicity of eigenvalue zero coincides with the number of connected components. All members of a cluster are mapped to the same point in the transformed space.

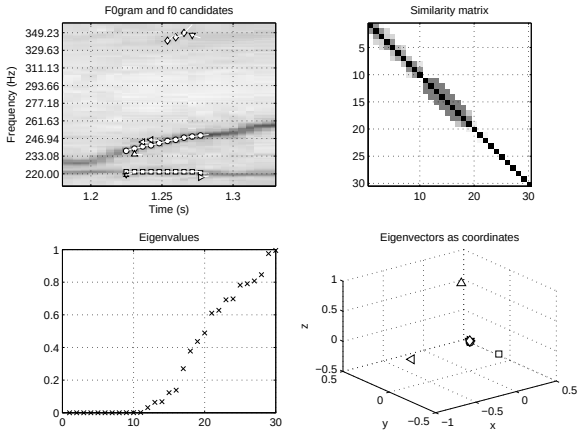

Figure 5: Local clustering example with two true pitch contours and several spurious peaks. The two corresponding bands in the similarity matrix can be appreciated. Multiplicity of eigenvalue zero not only indicates the relevant connected components but also isolated points. The true contours are correctly identified by the algorithm and spurious peaks tend to be isolated.

4.3 Number of clusters determination

Automatically determining the number of clusters is a difficult problem and several methods have been proposed for this task [7]. A method devised for spectral clustering is the eigengap heuristic [6]. The goal

is to choose the number k such that all eigenvalues $\lambda_1, \ldots, \lambda_k$ are very small, but λ_{k+1} is relatively large. Among the various justifications for this procedure, it can be noticed that in the ideal case of k completely disconnected components, the graph Laplacian has as many eigenvalues zero as there are connected components, and then there is a gap to the next eigenvalue.

This heuristic was implemented, but it sometimes failed to detect the correct number of cluster (e.g. when clusters are not so clear there is no well-defined gap). The following iterative strategy gave better results. It consist in firstly estimating the number of connected components using the multiplicity of eigenvalue zero by means of a restrictive threshold. Then, the compactness of the obtained clusters is evaluated. To do this, different measures were tested and a threshold on the sum of distances to the centroid in the transformed space was selected. As mentioned before, in case of completely separeted connected components all members of the same cluster are mapped to a single point in the transformed space. For this reason, the detection of poor quality clusters showed not to be too sensitive to the actual value used for thresholding. Each of the not compact clusters is further divided until all the obtained clusters conform to the threshold. This is done repeatedly by running k-means only to points in the cluster, starting with $k = 2$ for a bipartition and incrementing the number of desired clusters until the stop condition is met. This strategy tends to isolate each spurious peak as a single cluster (see Figure 5), what in turn favours to ignore them in the formation of pitch contours.

4.4 Filtering simultaneous members

Despite of the introduction of cannot-link constrains some clusters can occasionally contain more than one member at the same time instant. The best f0 candidate can be selected based on pitch distance to their neighbors. This approach was explored but difficulties were encountered for some particular cases. For instance, when a contour gradually vanishes F0gram peaks are less prominent, their pitch change rate estimate is less reliable and spurious peaks appear in the nearby region. Therefore, under the assumption of slow variation of contour parameters, salience similarity was introduced as another source of information. To do this, the most prominent peak of the cluster is identified and the cluster is traversed in time from this point in both directions, selecting those candidates whose salience is closest to the already validated neighbors.

4.5 Formation of pitch contours

The above described local clustering of f0 candidates has to be extended to form pitch contours. Increasing the length of the observation window showed not to be the most appropriate option. The complexity of the clustering is increased for longer windows, since a higher number of clusters inevitably arise mainly because of spurious peaks. Additionally, computational burden grows exponentially with the number of graph vertices. Thus, an observation window of 10 signal frames was used in the reported simulations ($\sim$ 60 ms).

Neighboring clusters in time can be identified based on the similarity among their members. A straightforward way to to this is by performing local clustering on overlapped observation windows and then grouping clusters that share elements. Figure 6 shows the clustering obtained using half overlapped observation windows for the two previously introduced examples.

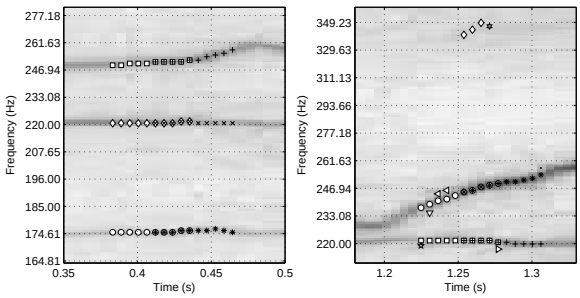

Figure 6: Examples of clustering using half overlapped observation windows. The pitch contours are correctly continued since several of their members are shared.

5 RESULTS AND DISCUSSION

The contours obtained by applying the proposed algorithm to the example audio excerpt are depicted in Figure 7. The three most prominent peaks of the F0gram are considered for pitch tracking. Several issues can be noted from these results. Firstly, the main contours present are correctly identified, without the appearance of spurious detections when no harmonic sound is present (e.g. around $t = 1.0\,s$). The example shows that many sound sources can be tracked simultaneously with this approach. No assumption is made on the number of simultaneous sources, which is only limited by the number of pitch candidates considered. The total number of contours and concurrent voices at each time interval is derived from the data.

It can also be seen that the third voice of the second note (approximately at $t = 1.0 - 2.0\,s$) is only partially identified by two discontinued portions. Because of the low prominence of this contour some of the pitch candidates appear as secondary peaks of the more prominent sources. This situation can be improved by increasing the number of prominent peaks considered.

Apart from that, there are the three short length contours detected at interval $t = 2.1 - 2.5\,s$ that seem to be spurious. However, when carefully inspecting the audio file it turned out that they correspond to harmonic sounds from the accompaniment. Although this contours have a very low salience they are validated because of their structure. It depends on the particular problem where this algorithm finds application if these contours may be better filtered out based on salience.

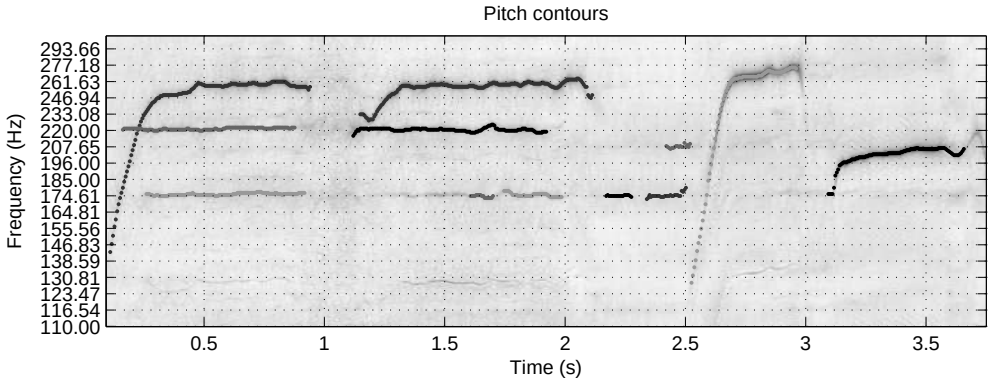

Figure 7: Pitch contours for the audio example obtained by considering the three most prominent F0gram peaks.

A melody detection evaluation was conducted following a procedure similar to the one applied in [1]. The vocal files of the 2004-2005 MIREX melody extraction test set were considered, which is a publicly labeled database available from `http://www.music-ir.org/mirex/`. It comprises 21 music excerpts for a total duration of 8 minutes.

The three most prominent F0gram peaks were selected as pitch candidates to form contours using the herein described algorithm. All the identified pitch contours were considered as main melody candidates and the ones that better match the labels were used to asses performance. Only those frames for which the melody was present according to the labels were taken into account to compute the evaluation measure according to,

$$\text{score}(f_0) = \min\{1, \max\{0, (\text{tol}_\text{M} - \Delta f_0)/(\text{tol}_\text{M} - \text{tol}_\text{m})\}\}$$

where $\Delta f_0 = 100|f_0 - f_0^{gt}|/f_0^{gt}$ is the relative error between the pitch contour value and the ground truth, and the tolerances tol_M and tol_m correspond to 3% and 1% respectively. This represents a strict soft thresholding.

The performance obtained in this way is compared to an equivalent evaluation that considers F0gram peaks as main melody estimates without performing any type of grouping into contours (as reported in [1]). Grouping the F0gram peaks into contours involves the determination of where does a contour starts and when does it ends, necessarily leaving some time intervals without melody estimation. This is avoided when isolated F0gram peaks are considered as main melody estimates, since for every melody labeled frame there is always a pitch estimation. Therefore, this performance measure can be considered as a best possible reference.

Results of the evaluation are presented in table 1. Two different values are reported for the pitch contours formation corresponding to a single run of the k-means algorithm and 10 repetitions. When the clusters in the transformed space are not well defined the k-means algorithm can get stuck in a local minima. This can be improved if several executions are performed but with different set of initial cluster centroid positions and the best performing solution is returned (i.e. lowest centroid distances). It can be noticed that the k-means repetition consistently gives a slight performance increase.

In addition, precision and recall values are reported. Precision is computed as the mean score value of the estimations within the 3% threshold. Remaining frames are considered not recalled items, as well as melody labeled frames for which there is no pitch contour.

When visually inspecting the results for individual files it turned out that most melody labeled regions for which there were no estimated contours correspond to low salience portions of the F0gram (for instance, when a note vanishes). It seems that labels are produced from monophonic files containing only the vocal melody and when mixed into a polyphonic track some regions are masked by the accompaniment. Figure 8 shows a detail of the current example where this situation can be appreciated. In order to take this into account the evaluation was repeated but ignoring low prominent melody frames. To do this a salience estimation was obtained for each labeled frame by interpolating the F0gram values. Then a global threshold was applied to discard those frames whose salience was below 30% of the F0gram maximum value (26% of the total frames).

Table 1: Results for the melody detection evaluation. The pitch contours are obtained from the three most prominent f0 candidates. An evaluation using F0gram peaks (1st to 3rd) without tracking is also reported.

F0gram	no salience threshold			30% salience threshold		
peaks	score	precision	recall	score	precision	recall
1	83.38	96.93	86.03	97.22	99.01	98.19
1-2	88.24	97.59	90.42	99.20	99.59	99.61
1-3	90.33	97.91	92.26	99.61	99.74	99.87

Pitch	no salience threshold			30% salience threshold		
contours	score	precision	recall	score	precision	recall
1 k-means	81.99	90.37	84.77	96.69	98.27	97.63
10 k-means	83.21	90.38	85.53	97.20	98.40	98.15
frames	100%			74%		

The performance of the pitch contours formation

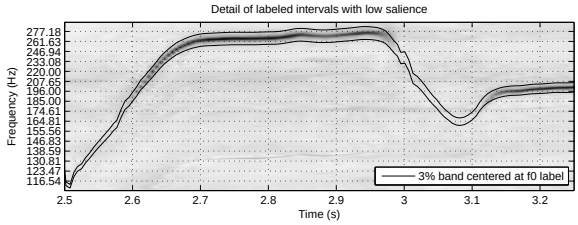

Figure 8: Some melody labeled regions of the example exhibit a very low salience (2.5-2.6 and 3.0-3.1 s).

by itself is quite encouraging. However, it decreases considerably compared to the values obtained before grouping F0gram peaks. The gap is reduced by restricting the evaluation to the most prominent peaks, which seems to confirm that low salience regions are troublesome for the algorithm. Visually inspecting the estimations for individual files gives the idea that most pitch contours are correctly identified. However, the evaluation results indicate the algorithm seems not to take full advantage of the information given by the F0gram peaks. Blindly relying on estimated α values no matter their corresponding salience is probably the most important shortcoming of the proposed algorithm.

6 CONCLUSIONS AND FUTURE WORK

In this work a novel way of performing pitch tracking by means of clustering local f0 candidates is described. The technique is based on an existing pitch salience representation called F0gram suited for polyphonic music [1]. This makes use of the Fan Chirp Transform which can produce precise representation of non stationary sound sources like singing voice.

The grouping is performed by applying a Spectral Clustering method since it can handle filiform shapes such as pitch contours. The similarity measure proposed takes advantage of the pitch change rate estimate provided by the FChT based F0gram. The determination of the number of clusters is tackled by an iterative approach, where the number of connected components is taken as an initial estimate and not compact enough clusters are further divided into an increasing number of groups. This strategy tends to isolate each spurious peak in a single cluster, what in turn favours to ignore them in the formation of pitch contours. Clustering is carried out for overlapped observation windows of a few hundred milliseconds and clusters from different time windows are linked if they share elements. In this way, groups that exhibit a coherent geometric structure emerge as pitch contours while the others are discarded.

The clustering approach to the tracking problem seems appealing because the solution involves the joint optimization of all the pitch contours present in a given time interval. Therefore, many sound sources can be tracked simultaneously and the number of contours and simultaneous sources can be automatically derived from the data. This differs from most classical multiple f0 tracking techniques in which each source is tracked in turn. In addition, the algorithm is unsupervised and relies on a few set of parameters. The influence of each parameter has not been fully assessed and the determination of optimal values will be tackled in future work. Preliminary results indicate that performance is not too sensitive to a particular configuration of some of them (e.g. number of candidates, k-means repetitions), but as it would be expected the values for σ_{f0} have to be set with more care. It is important to notice that the algorithm has low computational cost given that efficient algorithms exists for solving generalized eigenvector problems as well as for the k-means step.

Results of a melody detection evaluation indicate the introduced technique is promising for pitch tracking and can effectively distinguish most singing voice pitch contours. There is some room for improvement and the main shortcomings will be tackled in our future work. In particular, other sources of information should be included in the similarity measure in order to take full advantage of the local pitch candidates. The estimation of the pitch change rate is less reliable for low salience peaks. This could be taken into account when computing similarity, for example by adjusting the σ_{f0} value in accordance with the salience of the candidate.

REFERENCES

[1] P. Cancela, E. López, and M. Rocamora, "Fan chirp transform for music representation," in *13th Int. Conf. on Digital Audio Effects, Austria*, sep. 2010.

[2] A. de Cheveigné, "Multiple F0 estimation," in *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, D. Wang and G. Brown, Eds., pp. 45–79. IEEE / Wiley, 2006.

[3] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on PAMI*, vol. 22, no. 8, pp. 888–905, aug. 2000.

[4] F. R. Bach and M. I. Jordan, "Learning spectral clustering, with application to speech separation," *JMLR*, vol. 7, pp. 1963–2001, 2006.

[5] M. Lagrange, L. G. Martins, J. Murdoch, and G. Tzanetakis, "Normalized cuts for predominant melodic source separation," *IEEE Trans. on ASLP*, vol. 16, no. 2, pp. 278 –290, feb. 2008.

[6] Ulrike von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395–416, 2007.

[7] A. K. Jain, "Data Clustering: 50 Years Beyond K-Means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[8] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Advances in NIPS*. 2004, pp. 1601–1608, MIT Press.