Detection of a Specific Musical Instrument Note Playing in Polyphonic Mixtures by Extreme Learning Machine and Particle Swarm Optimization

Pat Taweewat

***Abstract*—In this work, we present a system for detecting a specific musical instrument note in polyphonic mixtures based on machine learning method. By using extreme learning machine as a classifier, we generate input features from magnitude data of FFT spectrum, which are processed by an adjustable dynamic level processor. We use the particle swarm optimization to tune the parameter values of this level processor to make the system have the optimal detection performance for the specific instrument signal. Three musical instruments were used in the experiments. These were trumpet, flute and clarinet. We also compare the detection results between using ELM with linear and non-linear output functions for selected features. Furthermore, we show that our system can be used as a musical instrument source tracking system by creating a trajectory of its fundamental frequency using the information from the outputs of the system and illustrating over the spectrogram.**

***Index Terms*—Extreme learning machine, particle swarm optimization, musical signal processing; musical information retrieval**

I. Introduction

Applications of musical instrument detection in audio signal are useful for musical information mining since we can directly categorize or search in musical audio database without need to have description data embedded in the audio signal. Furthermore, some musical processing algorithms such as noise reduction and source separation need to adjust the right parameters to give the best processing results. Therefore, knowing the type of musical instruments from audio signal makes these algorithms able to adjust parameters automatically. In this work, we present a system that is capable of detecting a single note of a specific (solo) musical instrument in musical polyphonic mixtures. This means the system has abilities for both detecting a musical instrument type and estimating its pitch (or playing note). The system can operate on block-by-block of audio signal, which is able to use it as a musical source tracking in audio signal by visualizing the trajectory of the fundamental frequency of the solo instrument (the specified instrument detected by the system) created by information given by our system on a time-frequency representation such as the spectrogram. In our point of view, this kind of detection is considered as a pattern classification task. A classifier in the system is used for mapping between input feature vector and indicating outputs. Each output represents each playing note of the specified (solo) musical instrument.

Manuscript received May 12, 2012; revised June 20, 2012.

Pat Taweewat is with School of Electrical and Information Engineering the University of Sydney,NSW, Australia(e-mail:pleaseaskpat@gmail.com)

Basically, the instrument detection system can be designed to use input feature vectors compiled by MPEG-7 standard [1]. However, our system needs more information about the fundamental frequency since it is designed to identify a playing note or F0 of the specific (solo) instrument, which specified by the user during training phase. The features should be able to provide information for estimating the fundamental frequency. Instead, features generations using information from FFT-magnitude spectrum are be used. We perform evaluations of different variations of methods for generating input feature vectors and present procedures to tune the parameter values of the level compressor using a variant of particle swarm optimization or PSO.

This paper is organized as follows. Section II is an over view of the system and method. Sections III-VI will explain about the feature extraction, the dynamic level processor, ELM and PSO respectively. We describe about our experiments in Section VII followed by the result discussions in VIII. Section IX will explain about visual experiment to create the trajectory while conclusion is in Section X.

II. The System and Proposed Method

The main concept of our work is to design a system that is capable of detecting a playing note of a specific musical instrument. This can be used for detecting and tracking the instrument signal even some polyphonic accompaniments exist in the background. The input of the system is audio signal and the outputs of the system are the playing notes of the specific musical instrument. Ideally, the outputs of the system should be active when considered instrument (specified by user during training phase) is played but, not active when other instruments are played as they are background music or noise. Therefore, this can be achieved by using a common pattern classification system that is composed of three parts: feature extraction, classification, and output mapping. The feature vector is generated in feature extraction part by using FFT and adjustable level processor while the classifier is Extreme Learning Machine (ELM), which its decision is made by selecting the highest firing output.

To optimize the system for having the best performance on the specific instrument signal, we use PSO to search for the optimal values for all parameters of the dynamic level procesisor used by the feature extraction part. The reason we rely on the level processor is that different musical instruments have different dynamic ranges and we assume that these differences can be used as dominating features to distinguish between different types of the instruments. We

DOI: 10.7763/IJIEE.2012.V2.198

show a simple diagram of our system in Fig. 1. The first part of the system is the feature extraction. We need the system to be able to operate on block-by-block of audio, which each block can be a single frame or multiple frames depending on a method for extracting the features (this is to prevent confusion with frame-by-frame when multiple frame features are used).

As mentioned, the features can be extracted from either single or multiple frames of FFT. Only magnitudes data are needed. The magnitudes are then processed using an adjustable dynamic level processor. To have the best performance, we need to tune the parameter values by PSO.

We choose ELM as the classifier because of its immediately learning ability. Finally, we design outputs of the classifier to have 89 outputs. These are for 88 active notes and one inactive note. Totally, we have 89 classes and only one class can active at time. The reason that we need one output for inactive note is to reduce the learning complexity of the problem since this configuration makes classification problem become multi-class problem instead of multi-label problem (two or more classes can be active at the same time). There are two situations to use the outputs of the system: instrument note detection and fundamental frequency trajectory tracking.

To detect the note, we apply a maximum function on the outputs of the ELM to find which output gives the highest firing value. We then assign that output to be active and the rest are non-active. The active output is used to indicate that the instrument in our consideration is active and playing the note which can be determined by the active output. For example, if the output number 40 is active, this instrument is playing C4.

To create the trajectory, we need to convert the note number from the output of the system and create a short continuous line between adjacent blocks (or frames) that have active notes. Normally, the conversion can be achieved by simple multiplication, depending on frequency scale used for the visual representation. For example, if we use 12-tet scale, there is no need for the conversion.

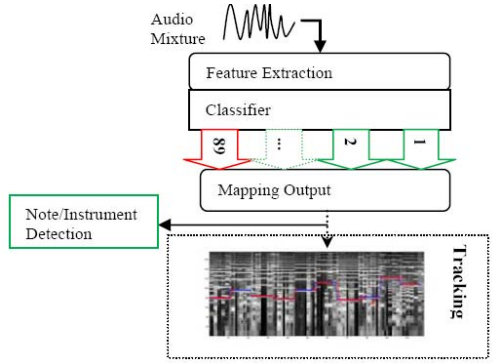

Fig. 1. Musical instrument detection system the output can be used for both detecting and tracking the specific instrument signals.

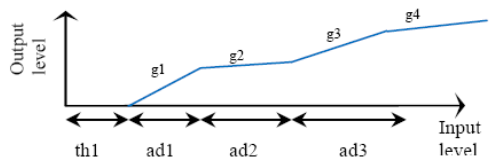

Fig. 2. The relationship between the parameters and the transfer function of the dynamic level processor.

III. Feature Extractions

There are 7 methods to extract the features in our works. These can be divided into two groups: single and multiple frame features. We show the summary of these methods in Table I.

A. Single-Frame Features

These features are extracted by taking FFT on a short block of audio signal (4096 samples). Only the first 512 of magnitude data from FFT are kept. These magnitude data are processed by the dynamic level processor and normalized to have the suitable values for the ELM. The extraction methods nos.1-2 are in this category.

B. Multiple-Frame Features

Basically, we extract these features by dividing each block of audio signal into multiple frames and then taking FFT on these multiple frames. Similarly, only the first 512 magnitude data are used. The size of each frame is 4096 samples while the numbers of frames are 5 for the methods nos. 3-5 and 9 for the methods nos. 6-7. For method nos. 3-5, we use a semi-exponential hop size to slide between adjacent frames. This is to reduce the number of frames used in each audio block compare to linear hop sizes. Thus, the hop sizes are 256, 256, 512, and 512 samples. For the methods no. 6-7, the linear hop sizes of 512 are used.

IV. Dynamic Level Processor

We design dynamic level processor. This dynamic level processor can operate by two steps: dynamic level adjusting and cleaning steps. In the dynamic level adjusting step, gains of the level processor are adjusted depending on the level from the magnitudes of FFT spectrum or spectra (for the multi-frame features). Then, these processed magnitudes are passed through the cleaning step, which is used to eliminate all magnitudes having level lower than threshold values by setting them to zeros. A process of the dynamic level processor can be shown as equations in (1)-(9) and the level transfer function in term of parameters is shown in Fig. 2.

A. 2- and 4-Point Level Processing

To investigate how flexibility effect on dynamic level processor, we implement dynamic level adjusting step using two configurations. Those are dynamic level adjusting with 2 and 4 adjustable points, which can be implemented using (7) and (8) respectively. Both types use the same cleaning step shown in (9). More adjustable points give more flexibility to tune the level processor for a specific audio source, but lead to more parameters and computational steps.

For 2-point level processing, 5 parameters are used. These are *th1*, *ad1*, *g1*, *g2* and *ct*. The *th1* and *ad1* are used to control threshold level while g1 and g2 are used to control the gains. The cleaning process is controlled by ct. For 4-point level processing, 9 parameters are used. These are *th1*, *ad1*, *ad2*, *ad3*, *g1*, *g2*, *g3*, *g4* and *ct*.

$$th2 = th1 + ad1 \quad (1)$$

$$th3 = th2 + ad2 \quad (2)$$

$$th4 = th3 + ad3 \quad (3)$$

$$ag1 = g1 \cdot ath2 \quad (4)$$

$$ag2 = g2 \cdot (th3 - th2) + ag1 \quad (5)$$

$$ag3 = g3 \cdot (th4 - th3) + ag2 \quad (6)$$

$$Aout = \begin{cases} 0 & ; \quad in < th1 \\ g1 \cdot in & ; \quad th1 \le in < th2 \\ g2 \cdot (in - th2) + ag1 & ; \quad th2 \le in \end{cases} \quad (7)$$

$$Aout = \begin{cases} 0 & ; \quad in < th1 \\ g1 \cdot in & ; \quad th1 \le in < th2 \\ g2 \cdot (in - th2) + ag1 & ; \quad th2 \le in < th3 \\ g3 \cdot (in - th3) + ag2 & ; \quad th3 \le in < th4 \\ g4 \cdot (in - th4) + ag3 & ; \quad th4 \le in \end{cases} \quad (8)$$

$$Cout = \begin{cases} 0 & ; \quad Aout < ct \cdot Average \\ Aout & ; \quad ct \cdot Average \le Aout \end{cases} \quad (9)$$

B. Band-Wise Processing

The level processor can also operate independently at different frequency bands by dividing the spectrum into 8 frequency bands. Two configurations for the band-wise processing are used. The first one, the bandwidths are exponentially increased toward high frequency whereas the second configuration, the bandwidths are constantly fixed over the whole spectrum. We use abbreviation E and L respectively in the Table I.

TABLE: SUMMARY OF THE FEATURES EXTRACTION METHODS.

Method	No.of Frames	No.of Level Adjustable points	No.of Frequency Bands
1	1	4	1
2	1	2	1
3	5 (See III-B)	2	1
4	5 (See III-B)	2	8E (See IV-B)
5	5	2	8L (See IV-B)
6	9	2	8E (See IV-B)
7	9	2	8L (See IV-B)

V. EXTREME LEARNING MACHINE

The extreme learning machine or ELM is a recent method for training single hidden-layer feed-forward neuron networks (SLFNs) without using of gradient-based parameter adjustment between input and hidden layers [2]. This concept is closely relating to a random vector field layer in functional-link neural network. We can train the neural network by randomly setting both bias and weight values of the hidden layer, then feeding all training examples to the network. Ignoring those weight and bias adjustments in the hidden layer, the neural network learns immediately by calculating the output-connection weights between the hidden and output layers using basic matrix operation (14) or another word, least-square solution. In this work, two ELM output types will be compared: linear and non-linear outputs.

A. Linear Output (Common Case)

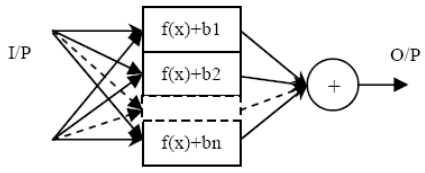

Fig. 3. ELM with linear output.

Basically, ELM is composed of two network layers. The first layer is hidden layer which has non-linear output units while the second layer is the output layer which contains the linearly summing unit to sum up values from the hidden-layer outputs as shown in Fig 3.

Assuming that we have N examples of n-dimension inputs and m-dimension outputs, the input patterns (example input vectors) are in *X* and output patterns (example output vectors) are in *Y* as show in matrix forms in (10) and (11) respectively. To train ELM with *L* hidden-layer units, we first randomly assign input-connection weights *w* and hidden biases *b* and calculate all *h* values using (12). The hidden-layer activation function is *tanh*. After that, we construct hidden-layer output matrix *H using (13)*. Finally, we calculate the output-connection weights *C by (14)*. After training, we use ELM to predict output values in matrix *T* by using (15).

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nn} \end{bmatrix} \quad (10)$$

$$Y = \begin{bmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{N1} & \cdots & y_{Nm} \end{bmatrix} \quad (11)$$

$$h_{ij} = f(\sum_{k=1}^{n} x_{ik} w_{kj} + b_j) \quad (12)$$

$$H = \begin{bmatrix} h_{11} & \cdots & h_{1L} \\ \vdots & \ddots & \vdots \\ h_{N1} & \cdots & h_{NL} \end{bmatrix} \quad (13)$$

$$C = (H^{\intercal}H)^{-1}H^{\intercal}Y \quad (14)$$

$$T = HC \quad (15)$$

B. Non-Linear Output

It is common for the neural network to have non-linear outputs to make strong output decision. This can be seen as Fig. 4 and equation (16) in ELM output layer without output-layer bias where *g* is the non-linear output activation function. In this case, we use (17) to calculate the output weights.

$$Y = g(HC) \quad (16)$$

$$C = (H^{\intercal}H)^{-1}H^{\intercal}g^{-1}(Y) \quad (17)$$

In this work, when ELM with non-linear output is mentioned, we mean *tanh* function is used as the output-layer activation function. Therefore, during prediction phase, we can calculate the ELM outputs using (18).

$$T = \tanh(HC) \quad (18)$$

In order to ensure that values from inverse of *tanh* function can be calculated, we scale the output patterns to have values between -0.9 and 0.9.

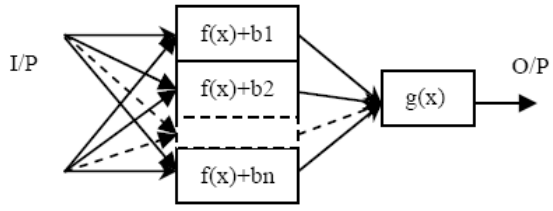

Fig. 4. ELM with non-linear output without output-layer bias.

VI. PARTICLE SWARM OPTIMIZATION

Finding the best parameters for the level processor is a

hard task as there are many possible solutions. In order to accelerate parameter searching, we use a particle swarm optimization or PSO to adjust parameters of the level processor.

A. Classical and ATRE-PSO

PSO is one of meta-heuristic optimization techniques [3]. The PSO mimics the behavior of a social interaction when creatures help together for finding the best solution of the problem for instance, a group of birds finding for their best food source. Each particle is act like each flying bird looking for the best solution or food source. Generally, each bird flies closely to the bird closest to the best local food source but also looking for other better food sources as well. That mean, all birds in the swarm are not only looking for the local best food sources but, also the global best food source. This strategy can be useful for finding the best parameters in a complex system, since there are many local best solutions and make many local traps for conventional searching methods.

Although a classical PSO as mentioning above is capable of solving many problems, we found that the algorithm has less variations control between particle positions. We therefore choose to optimize the level processor using ATRE-PSO [4] instead. This ATRE-PSO is a modified version of attraction and repulsion PSO to improve variation control between positions of particles in the swarm.

B. Implementation of ATRE-PSO

In our implementation, we introduce a mechanism to reduce useless particle by forcing the swarm to replace the worst particle with a copy of the best particle then put it always from the best particle by multiplying the best particle position with larger random number. This is to add chance to find another alternative better solution by repeating the random assigning value steps or steps (21)-(23) one more time if the worst particle is found.

$$dL = Lbest_{ij} - P_{ij} \tag{19}$$

$$dG = Gbest_{ij} - P_{ij} \tag{20}$$

$$ar = a \cdot random \tag{21}$$

$$br = b \cdot random \tag{22}$$

$$cr = c \cdot random \tag{23}$$

$$Vh_i = Iw \cdot V_i + ar \cdot dL + Iw \cdot br \cdot dG + Iw \cdot cr \tag{24}$$

$$Vm_i = Iw \cdot V_i + ar \cdot dL - Iw \cdot br \cdot dG + Iw \cdot cr \tag{25}$$

$$Vl_i = Iw \cdot V_i - ar \cdot dL - Iw \cdot br \cdot dG + Iw \cdot cr \tag{26}$$

$$DS = mean(std(P)) \tag{27}$$

$$Vnew_i = \begin{cases} Vh_i & ; \quad DShigh < DS \\ Vm_i & ; \quad DSlow \le DS < DShigh \\ Vl_i & ; \quad DS < DSlow \end{cases} \tag{28}$$

$$Pnew_{ij} = P_{ij} + Vnew_i \tag{29}$$

In the equations, we keep positions in n-dimensional space of each particle in *P* while each *V* is used to keep each particle velocity. We use subscript *i* for the *ith* particle and subscript *j* for *jth* dimension, which is *jth* parameter in this work. The *Lbest* and *Gbest* are the local and global best positions respectively. We calculate diversity using (27) where std is standard derivation.

C. Parameter-Searching Work Flow

To set the optimal parameters for the level processor we perform parameter-searching steps as follows.

1) N sets of parameter values are initialized randomly. In our work N=10 and it is the number of particles in the PSO.
2) Extract the features using parameter values, we have set and train the system.
3) Test the system using both training set and validation set (test set A). Finding average detection performance of both sets using (30). In our work, we give 2-time more weight to the performance of the validation set since a number of examples in validation set is twice that of the training set.

$$Perf = 0.5 * Perf._{train} + Perf_{validate} \tag{30}$$

4) Use PSO to find the new sets of parameter values by using (29).
5) if Perf is higher than the previous one then kept the new parameter set (if two sets having the same Perf then keep only one).

VII. Experimental Data Sets

A. Preparing the Samples

We used audio samples recorded by Phiharmonia Symphonic Orchestra [5] and MIS [6] databases.

B. Datasets

We generated three data sets, which are training set, test set A and test set B. These data sets were generated by first, a specific instrument, which was a solo instrument we needed to detect, was selected. After that, the note of this specific instrument was randomly selected and mixed with other randomly selected musical instruments from 16 instruments as they were used for simulating some background noise. These were instruments left after the first three instruments were chosen for acting as solo instrument. All instruments in the experiments are 2 manufacturers of trumpet, trombone, tuba, F. horn, oboe, E. horn, bassoon, clarinet, flute and only one manufacturer of piano. This step is repeatedly until we have enough numbers of examples for training and testing the system. In the experiment, we have 1400 examples for training set and 2800 examples for each test set. The mixing ratio between the solo instrument and background noise is 1:1 and the polyphonic numbers of background instruments are 1, 2, 4 and 6 notes (all have equal numbers of examples).

C. Training the System

The system was trained by two steps. The first step was to find the optimal parameters for the level processor. This step was done as explained previously in the parameter-searching section, which we used PSO. The training set and validation set (test set A) in this step was 1:2. This was to force PSO to find optimal solution with a good generalization rate since there were more unseen data than the trained data.

In the second step, after the optimized parameter values were found, we trained the system with these parameter values. The training set in this step was from test set A (2800

examples). This was to increase a number of examples when using the system in real situation.

In our experiment, we wanted to detect three solo instruments. Thus, we trained three systems for three different instruments. These instruments are trumpet, flute, and clarinet.

D. Result Calculations

We use two formulas to calculate the system performance. These are accuracy (31) and precision (32).Accuracy is used for measure class accuracy which gives information about how well the classifier can give the correct output class. For our system, 89 classes are in considering in the formula since the ELM in our system have 89 output classes. However, since we need to know only how well the system can give the right note rather than the right classes (we have 89 classes but, only one of them is considered for showing the result), precision may be more useful. If we consider both of the equations, precision is calculated based on false positive error (FP) only while accuracy adds false negative errors (FN) in calculation. In our system, if only one missing output active, the rest will be non-active. This gives both FP and FN even only one error occurs and both of the errors lead to twice the error measurement when using the accuracy formula.

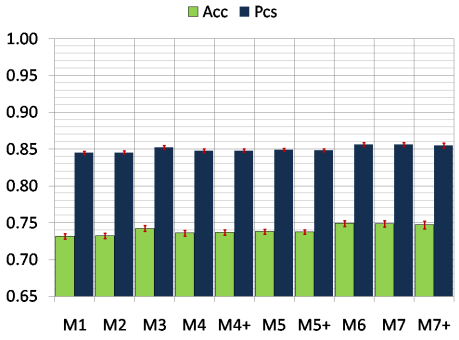

Fig. 5. Detection results for clarinet signal

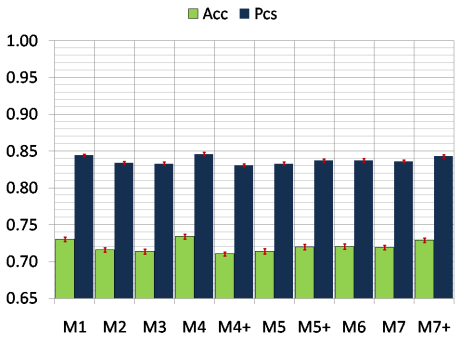

Fig. 6. Detection results for flute signal.

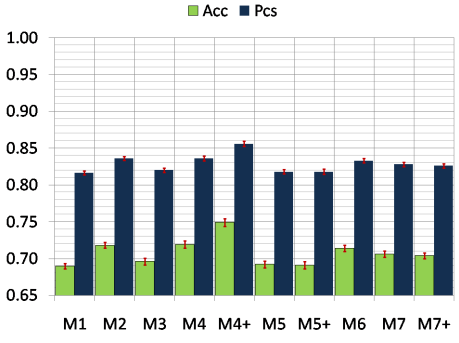

Fig. 7. Detection results for trumpet signal.

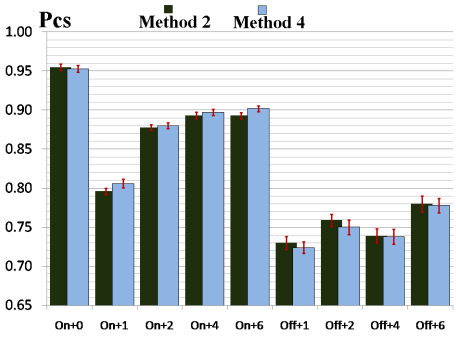

Fig. 8. Detection results for different mixing situations (Clarinet).

Thus, precision values in this work will be the same as ratio of correct detected cases/total detected cases.

$$Acc = \frac{TP}{TP+FP+FN} \quad (31)$$

$$Pcs = \frac{TP}{TP+FP} \quad (32)$$

Finally, we evaluated the system by training and testing the system for 100 times. The results are shown after finding the mean values of the detection results from this evaluation. We also put confidence interval values calculated with parameter

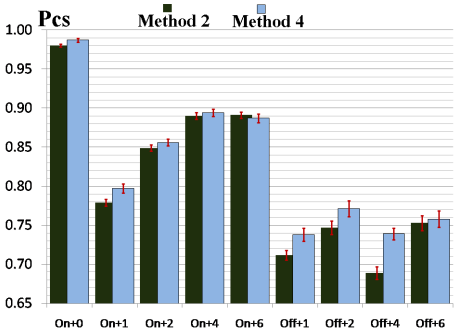

Fig. 9. Detection results for different mixing situations (Flute).

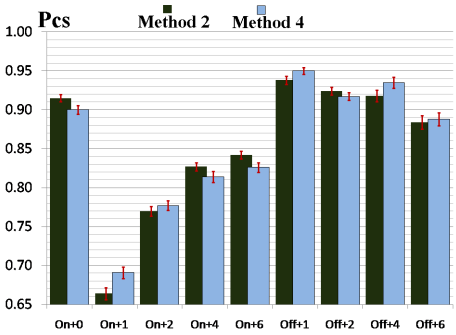

Fig. 10. Detection results for different mixing situations (Trumpet).

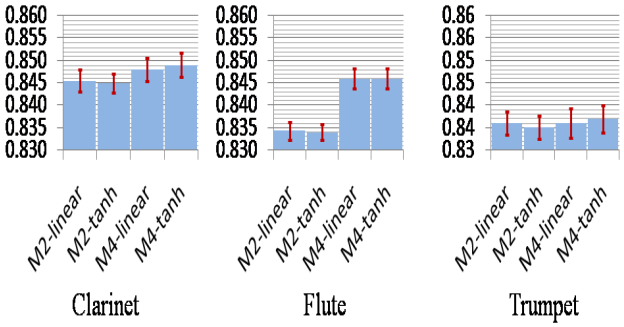

Fig. 11. A comparison between detection results using linear and non-linear (tanh) outputs.

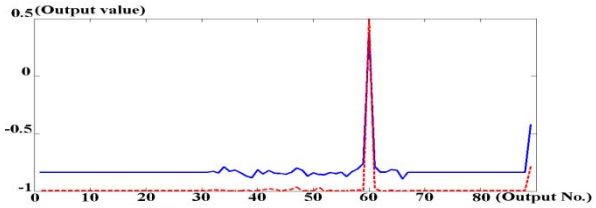

Fig. 12. A comparison between linear outputs (blue solid line) and non-linear outputs (red dash line) of the ELM.

$\alpha = 0.00001$on the top of the bars for all results in Figs 5-10. The methods 1-7 are indicated as M1-M7 in those Figs.

The results for some particular mixing situations of methods 2 and 4 are shown in Figs. 8-10. In those Figs., "on+x" labels are the results when the considered (solo) instrument was active in the test and x is a number of the polyphony in the background. In another case, when the considered instrument was not played, we use "off+x" labels. Therefore, only x polyphony instruments in the background were active.

VIII. Results and Discussions

We show the detection results for different feature extraction methods separately for clarinet, flue and trumpet signals in Figs.5-7. The feature extraction methods 1-7 are represented as M1-M7, while M4+, M5+, M7+ are method 4, 5 and 7 when parameter values of the level processor are used separately for different 8 bands. That means we optimized 40 parameter values (5 parameters x 8 bands) during the training phase. This is to make tuning mechanism more flexible and finely tune specific values to all different bands.

However, the detection results are almost the same for different feature extraction methods M4-5, and M7. This may show that separately parameter optimizing in each frequency band is not necessary. The methods M2 and M4 are selected for further evaluations as to represent single- and multiple-frame features. The different performances between single- and multiple-frame features are mixed depending on musical instruments. For clarinet and flute Figs. 8-9, multiple-frame feature seems to flavor the system but, oppositely for trumpet Fig 10. This may be because trumpet generates more varying spectra between frames so that the slightly distorted spectra can give error in the prediction of the classifier.

For linear and non-linear outputs, there are also very little different between the detection results as shown in Fig. 11. The actual outputs of the ELM are different as can be seen in Fig. 12. However, the decision of the classifier is finally depending on the maximum function, which makes the results almost the same for both types of the outputs.

Effects of different initial random input weights of ELM are also small for all features.

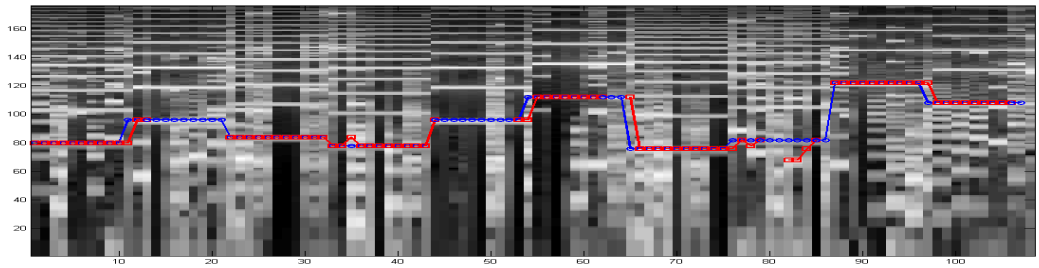

Fig. 13. Using the system to track the trumpet source. circles (blue) are the correct notes and rectangles (red) are the detected notes.

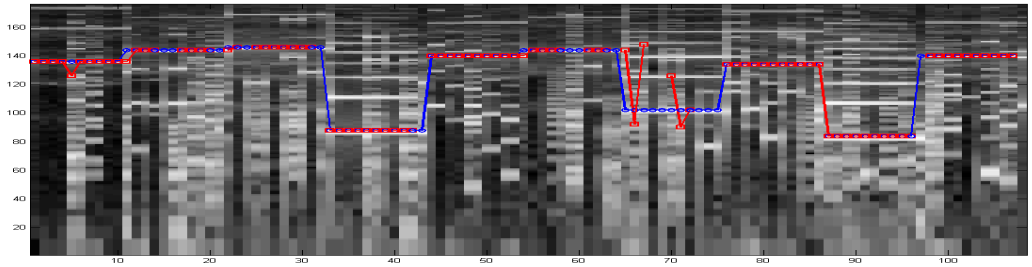

Fig. 14. Using the system to track the flute source. circles (blue) are the correct notes and rectangles (red) are the detected notes.

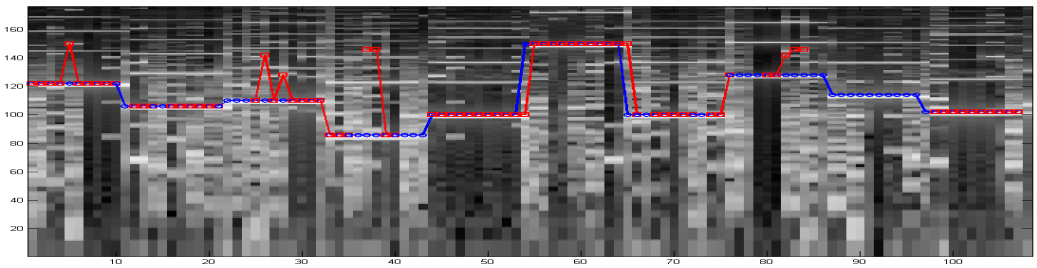

Fig. 15. Using the system to track the clarinet source. circles (blue) are the correct notes and rectangles (red) are the detected notes.

IX. Tracking Specific Musical Instruments

A higher representation of sound is useful for some application such as analysis and synthesis of musical and vocal signals. The example of this representation is sinusoidal model which is used to show the sinusoidal trajectory of an audio object [7]. The fundamental frequency trajectory of specific instrument is an alternative way to visualize the audio object and can tell where and when the considered or solo musical instrument is played. We show that our detection system can be used as musical instrument source tracking in time-frequency representation. In the visual experiment, we generated the first instrument signal using sample from RWC databases [8] and randomly mixed with sample from PHO [5] and MIS [6] databases. Although the system was not trained by the samples from RWC database, it can also track the right instrument at some accuracy. Figs. 13-15 show the trajectory tracking on top of the STFT (Short time Fourier Transform) magnitudes. In these Figs, we scaled the frequency axis (Y) using 24-tone-equal-temperament scale to make it easy to observe, while the time axis (X) is used to show the STFT frame. The trajectory is created by using information from outputs of the detection system which gives the fundamental frequency (the first harmonic) of the specific instrument. The fundamental frequency position (TFP_{pos}) in the T-F representation can be calculated using the following equation

$$TFP_{pos} = \left(\frac{27.5*2}{res}\right)^{\frac{Activeoutput-1}{n}} \quad (33)$$

where n is a number of tones per octave in n-tone equal temperament scale and $res = \frac{Sampling\ frequency}{FFT\ points}$ is frequency resolution if we put the trajectory on top STFT. Only the first 88 outputs are used for finding the position. The 89th output of the system will active when no note from the considered instrument is active. Therefore, the trajectory will stop when this output is active as we can see in the examples that sometime the system cannot detect the note even the considered instrument is playing. This is the problem that we will correct it in the future work.

X. Conclusion

We have presented the method to construct the musical instrument note detection system using ELM as the classifier. The front end of the system uses the adjustable level processor that can be tuned to have the best detection performance for the specific musical instrument by using the PSO during the training phase. The system can be trained to detect the note played by only the specific instrument, which make the system can be used in the instrument source tracking application by converting the outputs of the system to the fundamental frequency trajectory of the specified instrument and put it on the STFT magnitudes.

References

[1] P. Szczuko, et al, "MPEG-7-based Low-Level Descriptor Effectiveness In The Automatic Musical Sound Classification," *Presented at The 116th AES Covention*, Berlin, Germany.

[2] G. B. Huang, et al, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.

[3] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Presented at the IEEE International Conference on Neural Networks IV*, 1995.

[4] M. Pant, Et Al, "A Simple Diversity Guided Particle Swarm Optimization," *Presented at The CEC 2007*, 2007.

[5] P. Orchestra, The Sound Exchange [Online]. Available: http://www.philharmonia.co.uk/thesoundexchange/make_music/samples/

[6] Iowa MIS: Music Instrument Samples [Online]. Available: http://theremin.music.uiowa.edu/index.html

[7] X. Serra and J. S. III, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on A Deterministic Plus Stochastic Decomposition," *Computer Music Journal*, vol. 14, 1990.

[8] M. Goto and E. Al., "Rwc Music Database: Music Genre Database and Musical Instrument Sound Database," *In the 4th International Conference on Music Information Retrieval*, 2003.