Real-time Melodic Accompaniment System for Indian Music Using TMS320C6713

Prateek Verma
Department of Electrical Engineering
IIT Bombay
Mumbai , India
prateekv@ee.iitb.ac.in

Preeti Rao
Department of Electrical Engineering
IIT Bombay
Mumbai , India
prao@ee.iitb.ac.in

***Abstract*— An instrumental accompaniment system for Indian classical vocal music is designed and implemented on a Texas Instruments Digital Signal Processor TMS320C6713. This will act as a virtual accompanist following the main artist, possibly a vocalist. The melodic pitch information drives an instrument synthesis system, which allows us to play any pitched musical instrument virtually following the singing voice in real time with small delay. Additive synthesis is used to generate the desired tones of the instrument with the needed instrument constraints incorporated. The performance of the system is optimized with respect to the computational complexity and memory space requirements of the algorithm. The system performance is studied for different combinations of singers and songs. The proposed system complements the already available automatic accompaniment for Indian classical music, namely the sruti and taala boxes.**

Keywords - Pitch detection; DSP; Automatic accompaniment; Additive Synthesis .

I. INTRODUCTION

The purpose of this project is to build an automatic system that provides real-time melodic accompaniment in a music performance. Such a device can be useful in the Indian classical music performance context, in which there is a principal performer, rhythmic and melodic accompaniment and a drone instrument. Usually a *tanpura* is used as a drone instrument, which gives the tonal context to the melody. Melodic accompaniment is usually provided by a pitched instrument, such as a violin, *sarangi* or *harmonium*. The rhythmic accompaniment is usually provided by a percussive instrument, such as a *tabla* or *mridangam.* Typically the melodic accompaniment follows the singer's musical composition with some delay. For instance Carnatic vocalists are usually accompanied by the violin which shadows the singer's melody while also filling in the silent intervals. Likewise the harmonium tracks the Hindustani vocalist`s melody. Pitch-continuous instruments like the violin and sarangi follow the vocal melodic contour closely including the expressive ornamentation (known as *gamaka*). On the other hand, the harmonium is operated via keys providing a set of discrete, pre-tuned pitch values only. Thus the harmonium can be constrained in following some continuous and rapid pitch modulations of the voice. However it is still favored by Hindustani musicians due to its rich tonal quality.

These days tanpuras are increasingly being replaced by electronic tanpuras or *sruti* boxes. Also virtual software for taals (rhythmic accompaniment) is available since it is pre-decided before a particular vocal performance. Some accompaniment systems such as MySong® [1] and LaDida® [2] can create accompanied music but they do not operate in real time but rather add only chords and beats after recording the music. This paper proposes an automatic accompaniment system based on a digital signal processor (DSP) which can synthesize the sound of a selected instrument such as violin or harmonium, and provide real-time melodic accompaniment. This can be useful for musicians to practice by providing a virtual concert like scenario. It can also be used in concerts as a virtual accompanying instrument. Some of the challenges faced in developing such an embedded system application are the response time, memory space, throughput and accuracy. All these factors will play a major role in the design of the algorithm and its implementation on the DSP.

Unlike composition-driven western classical music, an Indian classical vocal performance is heavy in improvisation. Since the Indian classical vocalist creates the melody during the performance within a decided Raga and Tala framework, there is no musical score already available to the accompanist. Rather the accompanist listens to the main performer and reproduces the melody on the fly with at most a short delay. This makes the design of a real-time automatic melody accompanier in Indian classical music difficult and challenging. Further, unlike Western classical music, which uses a fixed tuning system, in Indian classical music the singer is free to choose the actual pitch of the tonic note at the start of every performance. Consequently, the system must be able to automatically tune itself to the reference key. In our design the system imitates the actual pitch of the singer/lead artist. Therefore it works similarly for both improvisation and fixed compositions.

The organization of the paper is as follows: Section II gives a brief review of the algorithm involved, followed by the description of each sub-system. Section III describes the DSP implementation of the system. Section IV describes the evaluation of the accompaniment system followed by the conclusion and future work.

II. ALGORITHM DESCRIPTION

Fig 1. shows a block diagram of the melodic accompaniment system which uses the detected input singing voice signal parameters together with tuning parameters extracted from a short training audio sample to generate the accompaniment audio of a selected melodic instrument. In this section, we provide an overview of the main modules in Fig. 1.divided into A. Tuning, B. Pre-Processing, C. Pitch Extraction and D. Instrument Synthesis. More detailed descriptions as well as implementation aspects appear in Sec. III.

A. Tuning

A knowledge of the singer's volume (amplitude level) and level of background noise, if any, impinging on the microphone are required to reliably distinguish the signal from any background noise that may be present . The mean energy of the singer is calculated and used for thresholding of silence/non silence region. Further, knowledge of the singer's tuning i.e. the locations of notes in pitch space is important information if the selected accompaniment is a keyed instrument. To obtain this information a singer is first asked to sing a brief excerpt, e.g. a few musical notes, as training data. Next the assumption of musical equal-temperament scale is used to describe the entire pitch space. The computation of the pitches of the singer in terms of the note locations is done for a musical piece. In general singers do not exceed a three octave pitch range.

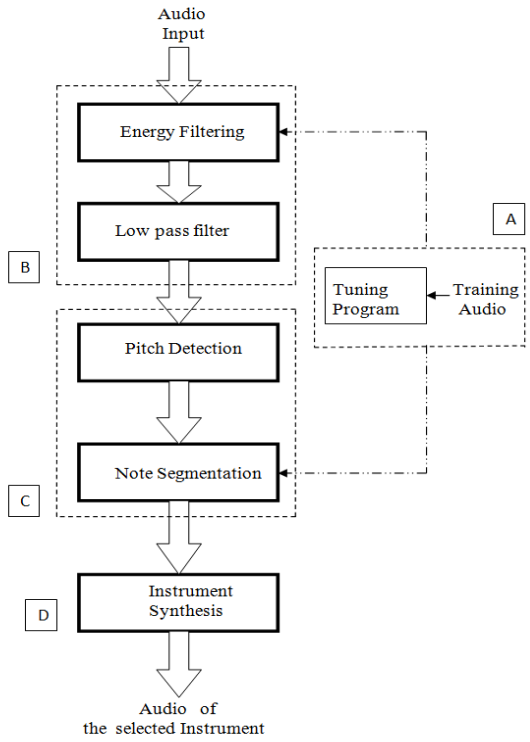

Fig 1. Block diagram of the proposed accompaniment system

B. Pre-Processing

It is very important to separate the silence and the non-silence regions of the signal in order to avoid the problem of spurious pitch values. Hence energy filtering is done wherein energy of each frame is calculated and then silence/voice determination is done. The average energy content of the musical piece is calculated from the tuning data with 0.55 of this value is taken as threshold on short-time energy to separate between the silence and non silence regions.

Voice signals have a rich harmonic structure spanning the frequency range 80 Hz to 5000 Hz. A pitched vocal utterance may contain 30-40 harmonic components. Moreover, the fundamental may not be the strongest one i.e. the first 2-8 harmonic components are usually stronger than fundamental component. The harmonic structure make pitch tracking complex with the possibility of octave errors. In order to improve the reliability and reduce the influence of strong higher order harmonics, low pass filtering is carried out.

C. Pitch Extraction and Note Segmentation

Pitch is detected every 40 ms by the short-time analysis of a windowed segment of the input signal. There are various pitch detection algorithms available both in the time and frequency domain based on the temporal periodicity and spectrum harmonicity respectively [4]. Table 1 depicts the comparison of well-known pitch detection methods, viz. the Average Magnitude Difference Function (AMDF), Auto Correlation Function (ACF) and Harmonic Product Spectrum (HPS). N denotes the number of samples in a frame and R is the number of harmonics considered in the HPS algorithm [5]. Generally the time domain approaches are computationally simpler. The AMDF algorithm only needs addition, subtraction and absolute value operations. Here we choose AMDF due to its low complexity and therefore suitability for real-time applications. A disadvantage of the AMDF method is that it is highly susceptible to background noise and intensity variation and therefore certain modifications are incorporated as described next.

TABLE I. COMPARISON OF VARIOUS PITCH DETECTION ALGORITHMS IN TERMS OF NUMBER OF COMPUTATIONS INVOLVED.

Algorithm	No. of Multiplications	No. of Additions
AMDF	-	N
ACF	N	N
HPS	2N(log N + R)	2N log N

The short time average magnitude difference function of a frame of the filtered input signal is defined as

$$x_w(m) = \frac{1}{N-m-1} \sum_{n=0}^{N-m-1} |s_w(n+m) - s_w(n)| \quad (1)$$

where $s_w(n)$ is the filtered voice signal, N is the analysis

window duration, and the lag m varies between 0 and N-1. Here we see that AMDF calculations require no multiplications, a desirable property for real time applications. We see that the function $x_w(m)$ is minimized at the integer multiples of the pitch period (i.e. whenever m=0, ±T, ±2T….). Ideally $x_w(n)$ should approach zero at multiples of the period T but since the signal is quasi-periodic the difference only attains local minimum.

Further, the position of the global minimum may not match the pitch period because of the effect of the amplitude variations, noise and formants of the voice signal [3]. These can cause octave errors in the obtained pitch. There are several methods available to reduce these errors. YIN [3] has employed a cumulative mean difference function (CMNDF) in order to reduce the occurrence of sub-harmonic errors. It de-emphasizes higher period dips in the difference function. The CMNDF is given below.

$$x'_{nw}(m) = \begin{cases} 1 & if\ m = 0\ , \\ x_w(m) \Big/ \left[(1/\mathrm{m}) \sum_{j=1}^{\mathrm{m}} x_w(j) \right] & otherwise. \end{cases} \quad (2)$$

Theoretically $x'_{nw}(n)$ should tend to zero for integral multiples of the period but actually a local minima exists at these points. The position of the minimum value is found. Knowing the sampling frequency and this value, the pitch value of the frame is found. The minimum value will not always correspond to the pitch period. If we search for the global minimum then a lot of octave and sub harmonic errors occur. To avoid this and improve the accuracy of the algorithm, only local minima below a given absolute threshold are considered [3].

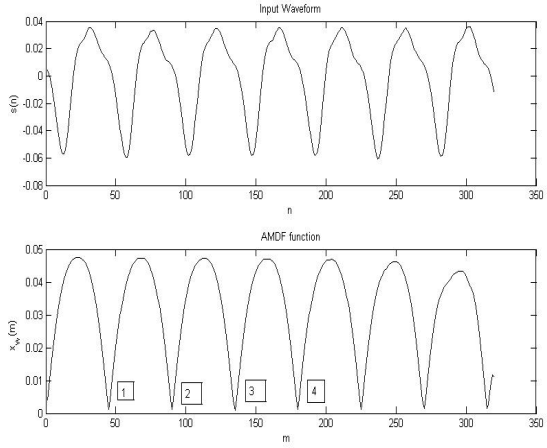

Fig 2. Example of a sub harmonic error.

For example, consider the AMDF function for a particular signal segment with actual pitch period = 45 samples in Fig 2. Clearly the global minimum, i.e. point 3 is not the pitch period. Hence first we store the local minima occurring in a frame in increasing order. Then a value for the threshold and the number of minima points to look for is set from experimental observation. Now out of the first n number of local minima after arranging in the increasing order (7 in the above case), the lag corresponding to the local minimum closest to m=0, and lying below a threshold, is taken to be the pitch period.

For error correction post processing of the pitch values is done. Here it is assumed that there cannot be a pitch variation of more than two octaves between two successive frames as they are only 40 ms apart . Also the ground truth pitch value of the voice is assumed not to exceed 800 Hz. Hence in such cases the previous valid pitch value is assigned as pitch value of the current frame.

Now after this stage some rules are incorporated in making the rendition of the "virtual instrument" as close to real instrument as possible. For a pitch-continuous instrument such as the violin, no further processing is needed. However for a keyed instrument such as the harmonium, we need a step of note segmentation and labeling. Fig 3. shows a comparison between time aligned pitch contour of the vocalist to that of a harmonium player. The continuous pitch contour corresponds to the vocals whereas discrete valued graph correspond to the same track being played by a harmonium player with time alignment. Based upon such data, some simple rules are incorporated in the algorithm for making our system as close to the instrument as possible.

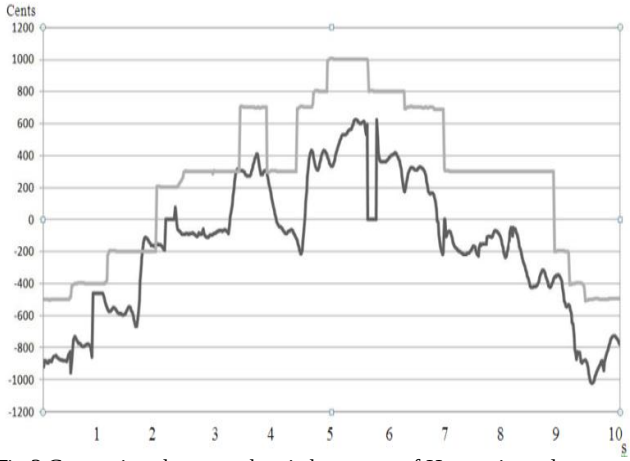

Fig.3 Comparison between the pitch contour of Harmonium player vs. Vocalist. A vertical offset has been applied to improve clarity.

If the raga is known beforehand (which is usually the case in case of classical music concerts) then all the swars which are not in a particular raga can be omitted at the time of note segmentation. We also see that there is a minimum duration for which a note is played in the actual instrument. This minimum note duration depends on the musical piece. Hence in our system also there is a provision of incorporating this, i.e. a note is played for a particular time duration which is

adjustable. Also there is always some delay present between the accompanist and the vocalist. Each note has some link with its preceding and succeeding note. These linking notes are called grace notes or *Kan-swars*. It was observed that these Kan-swars are ignored most of the times by the accompanist. We thus use rules on allowed note locations and duration to convert the vocalist's tracked pitch contour into a note sequence suitable for harmonium. We thus use rules on allowed note locations and duration to convert the vocalist's tracked pitch contour into a note sequence suitable for harmonium.

D. Instrument Synthesis

In human ear, pitch perceived is logarithmically related to the physical fundamental frequency. The cent is a logarithmic unit of measure used for musical intervals. Twelve-tone equal temperament divides the octave into 12 semitones of 100 cents each. Typically, cents are used to measure extremely small finite intervals, or to compare the sizes of comparable intervals in different tuning systems. In a piano the adjacent keys are 100 cents apart. An octave comprises of 1200 cents.

If the selected accompaniment is a keyed instrument such as the harmonium, the next step involves gridding, i.e. snapping the instrumental pitch values to a pre-determined note-grid. This note-grid is determined as 100 cent intervals as computed in the Tuning block. At every analysis time-instant (spaced 40 ms apart) the singer's pitch value is snapped to the nearest grid location.

The system uses additive synthesis to generate the instrumental sound. Additive synthesis is defined as the addition of one or more pure tones to generate the required musical timbre. If we analyze the sound of different musical instruments we see that the same note (pitch) differs in sound quality (timbre) due to following:

1. Non-harmonic components present.
2. Amplitude of each harmonic component according to the spectral envelope corresponding to the instrument timbre .
3. The temporal volume envelope of the played note .

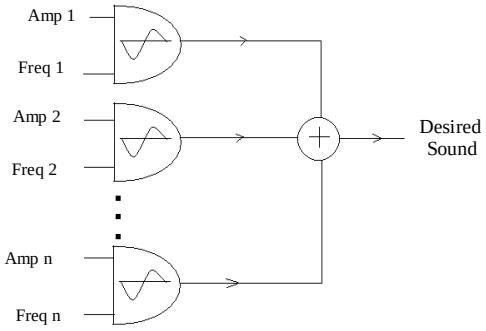

Figure 4. Additive synthesis

We simulate the desired sound accurately based on the following equation.

$$x(n) = \sum_{k=1}^{N} \left(a_k \sin \frac{2\pi f_0 kn}{F_s} + \varphi_k \right) \quad (3)$$

where a_k denotes the amplitude of the k^{th} harmonic (sinusoid) component, f_0 is the desired fundamental frequency, Fs is the chosen sampling frequency, φ_k is the phase of the kth harmonic, and N is the desired number of harmonics. The sound is synthesized using the implementation shown in Fig. 4 given the pitch, level (volume) and stored spectral envelope corresponding to the timbre of the sound. The harmonic phases are usually set to arbitrary initial values. Additive synthesis is a computationally complex but flexible sound modeling method. Inharmonic components are not considered here. Instrument temporal envelope modeling has not been considered here. But rather we assume that the harmonic amplitudes are constant over the note duration.

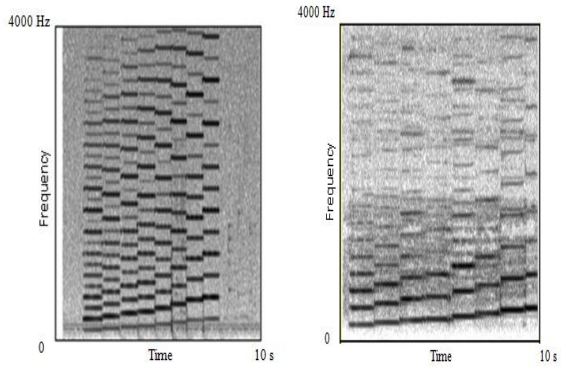

Fig 5. Spectrogram of an octave played on (a) harmonium and (b) flute.

From Fig 5. we see that harmonium is a spectrally rich instrument having dominant higher order harmonics. Thus synthesizing its sound takes up a lot of computation. On the other hand flute has at most five significant harmonics. Hence amount of computation time required for flute synthesis is less.

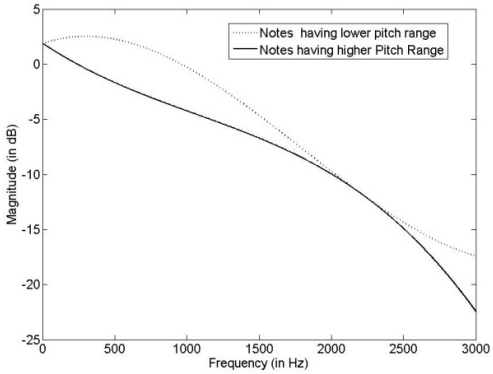

Fig.6 Comparison of spectral envelope of harmonium notes.

Fig.6. shows the smooth spectral envelope of a harmonium tone computed by averaging over a number of different notes in each of two frequency ranges:low (50-160 Hz) and high (170-512 Hz). Depending on the pitch of the desired note, the

harmonic amplitudes are read off one of these stored envelope for additive synthesis.

The human ear is very sensitive to abrupt changes in phase. Hence phase of each final value of the frame is tracked in order to make the synthesized waveform shape continuous. Otherwise the synthesis will result in artifacts manifested as clicking sounds after each frame, resulting in an unpleasant and poor audio quality.

III. DSP IMPLEMENTATION OF THE ACCOMPANIMENT SYSTEM

In this section we discuss implementation of the proposed accompaniment system on TMS320C6713 floating point Digital Signal Processor from Texas Instruments [6]. It has an optimized architecture for fast operation capable of a large number of mathematical operations on large chunks of data with minimum latency in order to facilitate real time operations. TMS320C6713 has a clock frequency of 225 MHz but an architecture giving higher MIPS (Million Instruction Per Second) compared to other processors of similar speed [7]. The advantage of this processor over a fixed point processor is that there are reduced errors due to overflow, truncation and rounding of operands. Additionally this processor has the ability to handle a large dynamic range and also algorithms are easier to implement in floating point DSP rather than fixed point. The CCS (Code Composer Studio v3. 1) is used for compiling, and creating an assembly language code and linking.

The present algorithm involves block-based processing i.e. computations are carried out every 40ms throughout the non-silent region of the input audio signal to generate estimated pitch corresponding to the centre of the 40ms frame.

A. Programming aspects of DSP Implementation

For low pass filtering in the preprocessing stage, an elliptical filter is chosen as no other filter of equal order can have a faster transition in gain between the pass band and stop band. Lesser order means less complexity and memory consumption. A 10^{th} order low pass elliptical filter with pass band as 600Hz and stop band attenuation of -80dB is designed. For implementing the 10^{th} order filter, its transfer function is broken into second order sections and then cascaded. This is done so as to reduce the quantization effects. Each of the second order section is written in terms of its difference equation. This takes lesser number of CPU cycles as it involves only addition and storage elements. Linear buffers are used to store the delayed by one sample values. After each iteration the new value is stored in the first element of the storage buffers whereas the last value is discarded. Every iteration results in the buffer elements being right shifted by one (first in, last out).

Division generally takes much more time as compared to other mathematical operations. Hence it is avoided as much as possible. Synthesis of required audio needs a lot of harmonic components to be generated each corresponding to a sine function. Hence we define a look up table for the values of the sine function. Again the intervals in the table must be optimized by experiments done without the loss of accuracy and quality. In order to save memory the look up table for only half the sinusoidal cycle is considered [8]. Thus 256 samples of a half cycle of sinusoid are stored. Sampling frequency cannot be too high since it means lesser time available to fill up the individual buffers and thus lesser processing time for the algorithm. Frame length too dictates the real time performance as well as pitch accuracy. Therefore repeated profiling was carried out till our system worked efficiently. It executes the code rapidly and uses fewer available resources on the chip.

Finally 8 kHz sampling was chosen as the best trade-off between sound quality and computation time. The size of the buffer for a 40ms analysis frame was thus equal to 320. Block-based processing with DMA was done instead of sample by sample processing due to computational complexity of the algorithm. Ping and Pong buffers alternately receive the samples or process and extract pitch. Instead of sorting the minima after AMDF computation which may take up a lot of computational cycles, a different approach is taken. First the global minimum of the function $x_w(m)$ is calculated for a particular frame. Then using a predetermined threshold only those minima that lie within the threshold are stored. Now out of these values the minima that has the least time index "m" is chosen as the desired value of the period. The values for the amplitudes of the Fourier coefficients of a particular instrument were stored in look up table to save the computational time.

This accompaniment system has many features which makes its similar to electronic tanpura in some ways. Its output pitch can be tuned to our desired pitch range providing flexibility. Since it was observed that the delay between the accompanying instrument and the lead vocalist varies depending on composition, mood and the need delay can be varied with . As of now there is a provision of playing one of the three accompanying instruments namely harmonium, violin and flute. But virtually any non percussion instrument can be synthesized by studying its spectrum for the amplitudes to be used in additive synthesis.

IV. EVALUATION OF THE ACCOMPANIMENT SYSTEM

For evaluation a set of five musical pieces is chosen. In order to cover a large number of variations songs having a variety of musical ornaments are selected. Five different singers were chosen. In the first stage the pitch accuracy is calculated. Here the accuracy depicts the total number of frames lying within a range of 50 cents of the original pitch value. A total of 745 seconds database of audio files was thus selected and the pitch accuracy was found to be 93.45% based on ground-truth pitch values obtained via a semi-automatic melody detection interface for polyphonic music [9].

After assigning fixed pitch values based on the detected user tuning, note names are assigned to the output of our "virtual instrument" for a particular song. Thus for each song we get a

musical score wherein each note duration is assigned according to the known tempo of the audio file. To validate the note segmentation stage, it was necessary to get the ground truth note sequence corresponding to the songs. This was achieved by getting an expert harmonium player to reproduce the song on harmonium while listening to the corresponding singing audio over headphones. The harmonium audio was transcribed into notes by automatic pitch detection. Word error rate is often used to evaluate the performance of a speech recognition system. Here after getting these two strings (i.e. the automatically segmented note sequence and the human harmonium experts note sequence) they can be compared for word error rate viz. number of matches, number of substitutions, deletions and insertions between two optimally aligned strings. Word error rate is computed as :

$$WER = \frac{S+D+I}{N} \quad (4)$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, N is the number of words in the reference which are the notes played by an actual instrument player.

TABLE II. COMPARISON OF ACTUAL INSTRUMENT AND PROPOSED VIRTUAL ACCOMPANIMENT SYSTEM IN TERMS OF WORD ERROR RATE OF NOTES.

Song Name	**% Notes Matching**	**% Note Substitutions**	**% Note Insertions**	**% Note Deletions**	**% WER**
Kesar	78.32	5.59	11.89	4.2	21.68
Kaise Paheli	77.78	13.07	8.17	0.98	22.22
Total	77.88	11.66	8.87	1.59	22.12

Table II shows the results for a subset of the songs. Here we note that even though the accuracy was very high, the number of matching notes given by DSP and that of actual player is around 77%. This happens because the duration for which a note is played depends on individual player and style. Also musical ornaments *Kan-Swar* and *vibrato* are followed by our system whereas real instrument players often tend to ignore such notes. Thus there are greater number of substitutions as compared to deletions as the current system follows whatever the main artist does whereas the accompanist may follow these subtle changes depending on the expertise.

After this we move to evaluation of the accompaniment system from the point of view of hardware resources utilized. The hardware resources utilized towards the DSP implementation of the accompaniment system is reported in Table III. Here since a 40 ms frame was chosen with a sampling frequency of 8 kHz available clock cycles are 9 x 10^6.

TABLE III. HARDWARE UTILIZATION OF THE ACCOMPANIMENT SYSTEM

Resource Utilization Details	
DSP Device	TMS320C6713
Clock Frequency	225MHz
Program Memory Used	187kbytes
Clock Cycles consumed per execution	8.9671 x 10^6
%idle processor time per execution	0.3661
% Memory consumed	83.1 %

V. CONCLUSION AND FUTURE WORK

In this paper implementation of a music accompaniment system, particularly for Indian music, is implemented on a TMS320C6713. It gives satisfactory performance both in terms of sound quality and accuracy. The C-code is written keeping in mind the real time considerations and the limited memory space available on the DSP chip. We have managed to synthesize the sound of harmonium by formulation of some simple rules but these need to be further refined. The temporal envelope of the synthesized tones needs to be shaped for more authentic quality. Also from the experiments carried out, we see that the virtual harmonium is tracking the *vibrato* effect by switching between two notes whereas a harmonium player does it seldomly. There remain such differences for which musicological knowledge is required.

AKNOWLEDGEMENT

This work was done while Prateek Verma was with the Bharti Centre for Communication at IIT Bombay. He would also like to thank Vishweshwara Rao for his kind suggestions.

REFERENCES

[1] http://khu.sh/

[2] http://research.microsoft.com/en-us/um/people/dan/mysong/

[3] A. D. Chveigne And H. Kawahara, Yin, A Fundamental Frequency Estimator For Speech And Music, J. Accoust. Soc. Am. 111 (4), 2002, pp 1917-1930.

[4] D. Gerhard, Pitch Extraction And Fundamental Frequency: History And Current Techniques, Technical Report, University Of Regina, Canada, 2003.

[5] Patricio De La Cuadra, Aaronmaster And Craig Sapp, Efficient Pitch Detection Techniques For Interactive Music , Center For Computer Research In Music And Acoustics, Stanford University.

[6] TMS320C6713 Floating Point Digital Signal Processor, SPRS186L, Nov 2005.

[7] http://www.eecg.toronto.edu/~moshovos/ACA05/004-pipelining.pdf

[8] Francis Kua,Generation Of A Sine Wave Using A TMS320C54xx Digital Signal Processor, Application Report Texas Instruments SPRA 819, July 2004

[9] S. Pant, V. Rao and P. Rao, A Melody Detection User Interface For Polyphonic Music, Proc. of the National Conference on Communications (NCC), 2010, Chennai, India.