ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 12, December 2016

Android software based musical analyser

Minal R. Wadyalkar
Electronics and Telecommunication department
G. H. Raisoni College of Engineering
Nagpur, India

Dr. Kalyani Akant
Associate Prof., Electronics and Telecommunication dep.
G. H. Raisoni College of Engineering
Nagpur, India

Abstract— **In this system we are performing real time note tracking for musical input using microphone of the android smart phone. For doing this pitch detection algorithm is used. We are finding out the pitch frequency of the note sound. The musical input will be given. The sung musical clip is processed on real time. Pitch of the notes sung will be found out. Pitch is nothing but the fundamental frequency of the sound. The pitch of note is found out using Fourier of Fourier Transform. The exact value of pitch is found out using Parabolic Interpolation to the spectral peaks. To track the pitch over time, the music sample is divided into overlapping windows of 46.4 ms duration.**

The android mobile phone is used to detect the sound clip through microphone. The android program is used for generating the android file for mobile phone input module. The graph is used for the values which are shown in previous seminar to detect the pitch of the input sound in real time basis. The eclipse software is used for generating the graph of the input sound clip. Then the FFT of the input values are shown. The FFT is nothing but the Fast Fourier Transform. FFT of FFT are also shown.

Keywords— pitch, Fourier of Fourier Transform, parabolic interpolation

I. Introduction

In this, we are going to prepare the android app for detecting the pitch of any input sound clip. Input signal is sound clip with various frequencies. Processing and scanning of input signal using Fourier of Fourier Transform. Analysis of input signal is done using FFT2 which is nothing but the Fourier of Fourier Transform. Obtaining graph of pitch at the output and also calculate the pitch of the signal at the output.

A Pitch detector is an essential component in a variety of speech processing systems. In this system we are performing real time note tracking for musical input using microphone of the android smart phone. For doing this pitch detection algorithm is used. We are finding out the pitch frequency of the note sound. The musical input will be given. The sung musical clip is processed on real time. Pitch of the notes sung will be found out. Pitch is nothing but the fundamental frequency of the sound. The pitch of note is found out using Fourier of Fourier Transform. The exact value of pitch is found out using Parabolic Interpolation to the spectral peaks. To track the pitch over time, the music sample is divided into overlapping windows of 46.4 ms duration.

Due to rapid explosion in size of music databases on the internet, demand for Content Based Music Information Retrieval applications has increased significantly. Melody extraction of singing voice in polyphonic context is one of the applications of Music Information Retrieval (MIR) which requires accurate estimation of pitch of vocal segments. To do this we need to identify vocal and non-vocal segments correctly. The problem of extraction of pitch contour of singing voice in the context of the polyphonic recordings of Indian Classical Music (ICM) is addressed[5]. Raga is the most important concept in Indian music, making accurate recognition a prerequisite to almost all musical analysis [8]. Background noise also presents a major difficulty to hearing aid wearers, and noise reduction is considered a great challenge for hearing aid design. Natural speech contains both voiced and unvoiced portions, and voiced portions account for about 75-80% of spoken English. The whole sound is recorded and processed.

The android mobile phone is used to detect the sound clip through microphone. The android program is used for generating the android file for mobile phone. The graph is used for the values to detect the pitch of the input sound in real time basis. The eclipse software is used for generating the graph of the input sound clip. The input is musical note on real time basis. Then this signal is processed by using Fourier of Fourier Transform. This signal obtained which are shown using graph. The graph which is represented are having the pitch values. Hence pitch of the graph is shown as a output for this project. Android sdk software is used for android programming. Program is used to prepare for various purposes in android software. For generation of android package file eclipse 3.4 is used and this file is used to show the graph in android mobile phones. Then the pitch detection algorithm is used and in android phone the graph is shown.

II. Methodology

This section describes the general operation of the FFT. In complex notation, the time and frequency domains each contain *one signal* made up of *N complex points*. Each of these complex points is composed of two numbers, the real part and the imaginary part. For example, when we talk about complex sample $X[42]$, it refers to the combination of $ReX[42]$ and $ImX[42]$. In other words, each complex variable holds two numbers. When two complex variables are multiplied, the four individual components must be combined to form the two components of the product. The following discussion on *"How the FFT works"* uses this jargon of complex notation. That is, the singular terms: *signal, point, sample*, and *value*, refer to the *combination* of the real part and the imaginary part.

The FFT operates by decomposing an *N* point time domain signal into *N* time domain signals each composed of a single point. The second step is to calculate the *N* frequency spectra corresponding to these *N* time domain signals. Lastly, the *N* spectra are synthesized into a single frequency spectrum.

Figure 2.1 shows an example of the time domain decomposition used in the FFT. In this example, a 16 point signal is decomposed through four

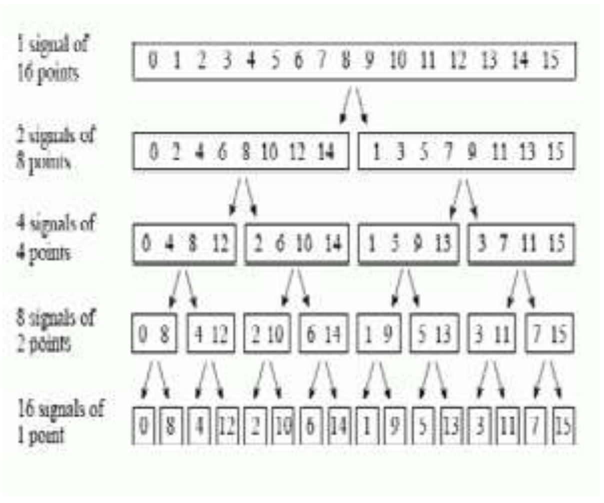
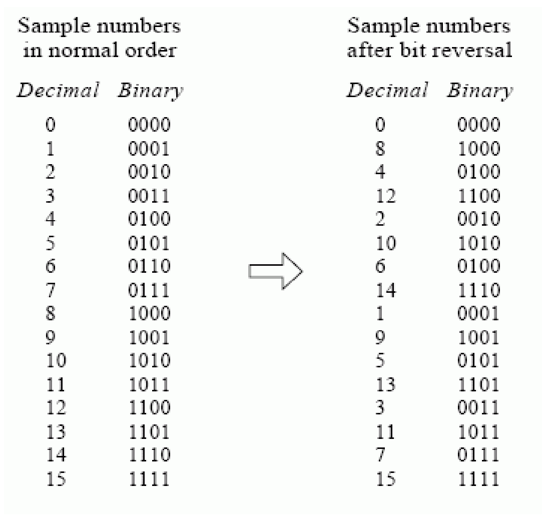


Figure 2.1: The FFT decomposition. An N point signal is decompose into N signals each containing a single point. Each stage uses an interlace decomposition, separating the even an odd numbered samples.



Sample numbers in normal order			Sample numbers after bit reversal	
Decimal	*Binary*		*Decimal*	*Binary*
0	0000		0	0000
1	0001		8	1000
2	0010		4	0100
3	0011		12	1100
4	0100		2	0010
5	0101		10	1010
6	0110	⇨	6	0100
7	0111		14	1110
8	1000		1	0001
9	1001		9	1001
10	1010		5	0101
11	1011		13	1101
12	1100		3	0011
13	1101		11	1011
14	1110		7	0111
15	1111		15	1111

Figure 2.2: The FFT bit reversal sorting. The FFT time domain decomposition can be implemented by sorting the samples according to bit reversed order.

separate stages. The first stage breaks the 16 point signal into two signals each consisting of 8 points. The second stage decomposes the data into four signals of 4 points. This pattern continues until there are *N* signals composed of a single point. An **interlaced decomposition** is used each time a signal is broken in two, that is, the signal is separated into its even and odd numbered samples. The best way to understand this is by inspecting Fig. 2.2 until you grasp the pattern. There are Log_2N stages required in this decomposition, i.e., a 16 point signal (2^4) requires 4 stages, a 512 point signal (2^7) requires 7 stages, a 4096 point signal (2^{12}) requires 12 stages, etc. Remember this value, Log_2N; it will be referenced many times in this chapter.

Now that you understand the structure of the decomposition, it can be greatly simplified. The decomposition is nothing more than a *reordering* of the samples in the signal. Figure 2.3 shows the rearrangement pattern required. On the left, the sample numbers of the original signal are listed along with their binary equivalents. On the right, the rearranged sample numbers are listed, also along with their binary equivalents. The important idea is that the binary numbers are the *reversals* of each other. For example, sample 3 (0011) is exchanged with sample number 12 (1100). Likewise, sample number 14 (1110) is swapped with sample number 7 (0111), and so forth. The FFT time domain decomposition is usually carried out by a **bit reversal sorting** algorithm. This involves rearranging the order of the *N* time domain samples by counting in binary with the bits flipped left-for-right (such as in the far right column in Fig. 2.3).

The next step in the FFT algorithm is to find the frequency spectra of the 1 point time domain signals. Nothing could be easier; the frequency spectrum of a 1 point signal is equal to *itself*. This means that *nothing* is required to do this step. Although there is no work involved, don't forget that each of the 1 point signals is now a frequency spectrum, and not a time domain signal.

The last step in the FFT is to combine the *N* frequency spectra in the exact reverse order that the time domain decomposition took place. This is where the algorithm gets messy. Unfortunately, the bit reversal shortcut is not applicable, and we must go back one stage at a time. In the first stage, 16 frequency spectra (1 point each) are synthesized into 8 frequency spectra (2 points each). In the second stage, the 8 frequency spectra (2 points each) are synthesized into 4 frequency spectra (4 points each), and so on. The last stage results in the output of the FFT, a 16 point frequency spectrum.

Figure 2.4 shows how two frequency spectra, each composed of 4 points, are combined into a single frequency spectrum of 8 points. This synthesis must *undo* the interlaced decomposition done in the time domain. In other words, the frequency domain operation must correspond to the time domain procedure of *combining* two 4 point signals by interlacing. Consider two time domain signals, *abcd* and *efgh*. An 8 point time domain signal can be formed by two steps: dilute each 4 point signal with zeros to make it an

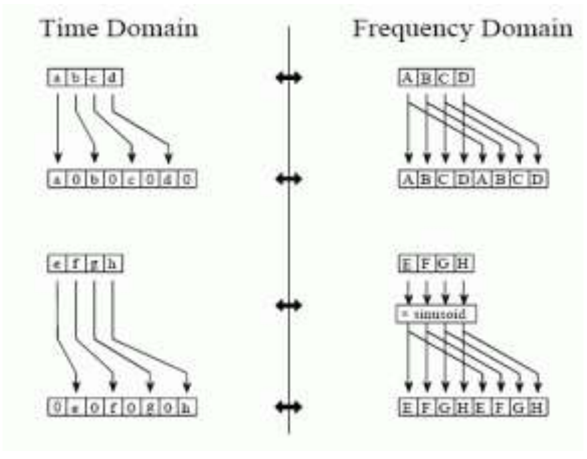


Figure 2.4: The FFT synthesis. When a time domain signal is diluted with zeros, the frequency domain is duplicated. If a time domain signal is also shifted by one sample during the dilution, the spectrum will additionally be multiplied by a sinusoid.

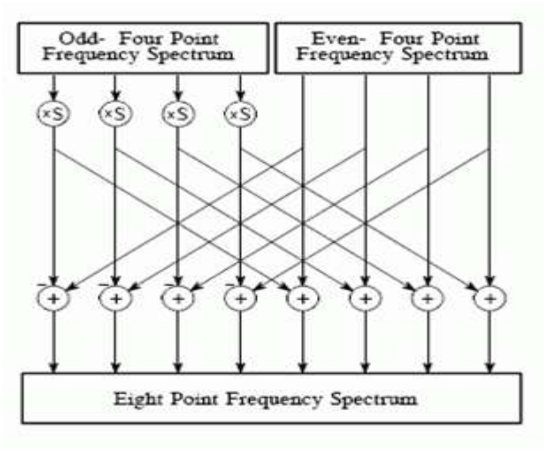

Odd- Four Point Frequency Spectrum
Even- Four Point Frequency Spectrum
×S
Eight Point Frequency Spectrum

Figure 2.5: FFT Synthesis flow diagram. This shows the method of combining two 4 point frequency spectra into a single 8 point frequency spectrum. The *s operation means that the signal is multiplied by a sinusoid wth an appropriately selected frequency.

8 point signals. and then add the signals together. That is, *abcd* becomes *a0b0c0d0*, and *efgh* becomes *0e0f0g0h*. Adding these two 8 point signals produces *aebfcgdh*. As shown in Fig. 2.4, diluting the time domain with zeros corresponds to a *duplication* of the frequency spectrum. Therefore, the frequency spectra are combined in the FFT by duplicating them, and then adding the duplicated spectra together.

In order to match up when added, the two time domain signals are diluted with zeros in a slightly different way. In one signal, the *odd points* are zero, while in the other signal, the *even points* are zero. In other words, one of the time domain signals (*0e0f0g0h* in Fig. 2.4) is shifted to the right by one sample. This time domain shift corresponds to multiplying the spectrum by a *sinusoid*. To see this, recall that a shift in the time domain is equivalent to convolving the signal with a shifted delta function. This multiplies the signal's spectrum with the spectrum of the shifted delta function. The spectrum of a shifted delta function is a sinusoid .

Figure 2.5 shows a flow diagram for combining two 4 point spectra into a single 8 point spectrum. To reduce the situation even more, notice that Fig. 2.5 is formed from the basic pattern in Fig 2.6 repeated over and over.

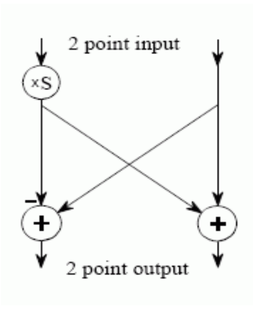


Figure 2.6: The FFT butterfly. This is the basis calculation element in the FFT, taking two complex points and converting them into two other complex points.

This simple flow diagram is called a **butterfly** due to its winged appearance. The butterfly is the basic computational element of the FFT, transforming two complex points into two other complex points.

Figure 2.7 shows the structure of the entire FFT. The time domain decomposition is accomplished with a bit reversal sorting algorithm. Transforming the decomposed data into the frequency domain involves *nothing* and therefore does not appear in the figure.

The frequency domain synthesis requires three loops. The outer loop runs through the Log_2N stages (i.e., each level in Fig. 2.2, starting from the bottom and moving to the top). The middle loop moves through each of the individual frequency spectra in the stage being worked on (i.e., each of the boxes on any one level in Fig. 2.2). The innermost loop uses the butterfly to calculate the points in each frequency spectra (i.e., looping through the samples inside any one box in Fig. 2.2). The overhead boxes in Fig. 2.7 determine the beginning and ending indexes for the loops, as well as calculating the sinusoids needed in the butterflies.

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 12, December 2016

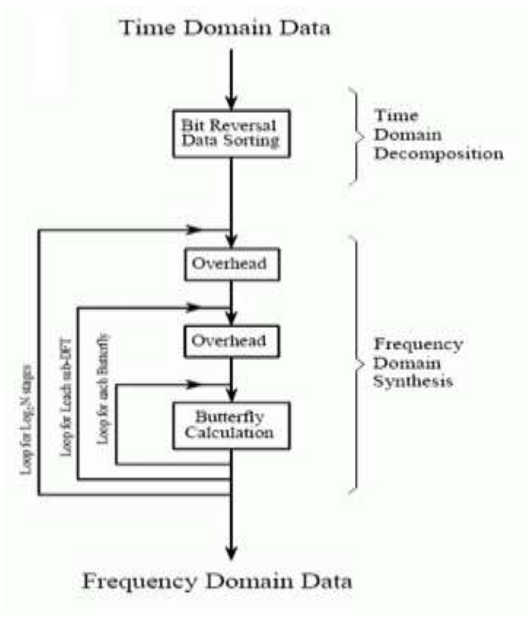


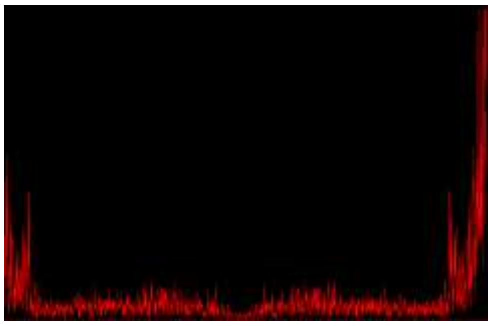
Figure 2.7: Flow diagram of the FFT. This is based on three steps: (1) decompose an N point time domain signal into N signals each containing a single point, (2) find the spectrum of the each of the N point signals (nothing required),and (3) synthesize the N frequency spectra into a single frequency spectrum.

Results

[illegible]

Screenshot of values after input

Screenshot of signal



FFT of signal

References

[1] Akant, K.A., R. Pande, and S.S. Limaye. 2011. "Pitch detection algorithm using harmonic pattern matching in Fourier of Fourier transform domain" in *ENGINEERING TODAY QUARTERLY JOURNAL* DECEMBER 2010 VOLUME II ISSUE 4, pp.149-159.

[2] Akant, K.A., R. Pande, and S.S. Limaye. 2010. "Monophony / Polyphony Classification system Using Fourier of Fourier Transform". *International Journal of Electronics Engineering*. Vol. 2, Number 2,pp.299-303.

[3] Akant, K.A., R. Pande, and S.S. Limaye. 2010. "Accurate Monophonic Pitch Tracking Algorithm for QBH And Microtone Research". *Pacific Journal of Science and Technology. 11(2):342-352.*

[4] Akant, K.A., Pande, R., Limaye, S.S. "Automatic Music Transcription of Indian Classical Music into MIDI Data", Accepted in *International conference on Advances in Communication, Network and Computing*, CNC 2012, to held on 24-25 Feb 2012.

[5] "Pitch Contour Extraction of Singing Voice in Polyphonic Recordings of Indian Classical Music" submitted to Computer Music Journal

[6] Zhenyu Zhao, Lyndon J. Brown, "Musical Pitch Tracking using Internal Model Control Based Frequency Cancellation", 42nd IEEE Conference on Decision and Control, 5, December 2003, pp. 5544 – 5548.

[7] L.R. Rabiner, et.al. "A Comparative Performance Study of Several Pitch Detection Algorithms", IEEE Trans. ASSP, 24 (5), pp. 399 – 418, October 1976

[8] Chordia, P., Rae, A.: Raag recognition using pitch-class and pitch-class dyad distributions. In Proceedings of International Conference on Music Information Retrieval (2007