BAB II

LANDASAN TEORI

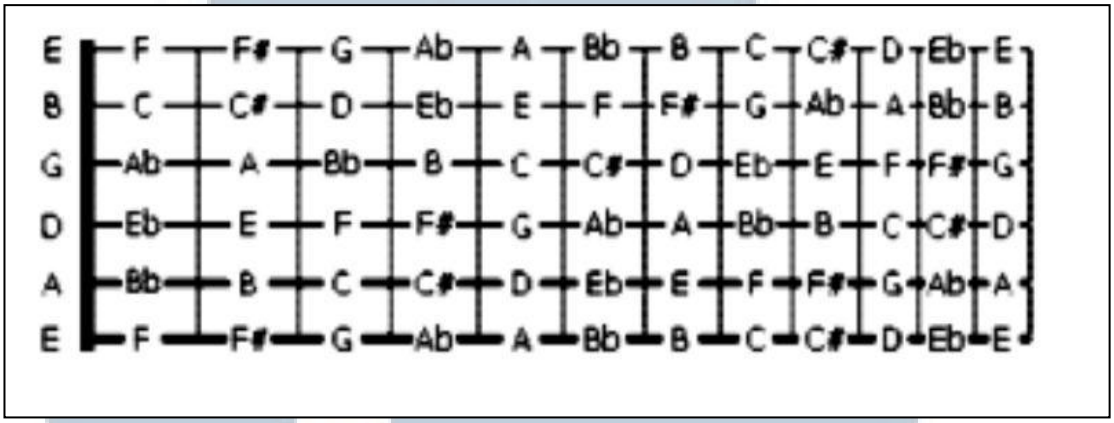
2.1 Tangga nada dan *Chord*

Tangga nada merupakan kumpulan dari nada-nada yang harmonis. Tangga nada kromatik merupakan kumpulan dari semua nada dalam musik. Karena ada nada yang berulang untuk setiap oktaf, istilah 'tangga nada kromatik' sering dipakai untuk ke-12 nada dari tiap oktaf. Tabel berikut menunjukkan frekuensi nada tiap fret dari fret 0 hingga fret 13 pada *tuning* standar [1].

Tabel 2.1 tabel frekuensi nada [1]

Fret	Senar 6	Senar 5	Senar 4	Senar 3	Senar 2	Senar 1
0	2.41	110.00	146.83	196.00	246.94	329.63
1	87.31	116.54	155.56	207.65	261.63	349.2
2	92.50	123.47	164.81	220.00	277.18	369.99
3	98.00	130.81	174.61	233.08	293.66	392.00
4	103.83	138.59	185.00	246.94	311.13	415.30
5	110.00	146.83	196.00	261.63	329.63	440.0
6	116.54	155.56	207.65	277.18	349.23	466.16
7	123.47	164.81	220.00	293.66	369.99	493.88
8	130.81	174.61	233.08	311.13	392.00	523.25
9	138.59	185.00	246.94	329.63	415.30	554.37
10	146.83	196.00	261.63	349.23	440.00	587.33
11	155.56	207.65	277.18	369.99	466.16	622.25
12	164.81	220.00	293.66	392.00	493.88	659.26
13	174.61	233.08	311.13	415.30	523.25	698.46

Dan berikut ini adalah tangga nada kromatik pada gitar.



Gambar 2.1 tangga nada kromatik pada gitar (fret 0 – 12) [1]

Chord merupakan gabungan dari beberapa not yang dimainkan secara bersamaan membentuk harmoni sebagai pengiring melodi yang dimainkan. *Chord* dapat dimainkan dengan variasi *strumming* sesuai dengan tipe dan tempo lagu yang dimainkan, atau juga dipetik. Ada banyak jenis *chord*, antar lain *major, minor, diminished, augmented, minor* 7^{th}*, major* 7^{th}*, suspended* dan lain-lain. Masing-masing *chord* memiliki karakteristik dan aturan pembentukan sendiri [7].

Berdasarkan tujuan dari topik penelitian ini, maka *chord-chord* natural yang akan dijadikan sampel adalah *chord major* yang terdiri dari:

- C (C – E – G)
- D (D – F# – A)
- E (E – G# – B)
- F (F – A – C)

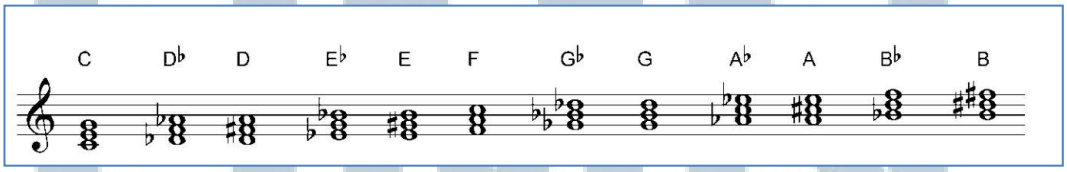
- G (G – B – D)
- A (A – C# – E)

Dan *chord dominant seventh* yang terdiri dari:

- C7 (C – E – G – Bb)
- D7 (D – F# – A – C)
- E7 (E – G# – B – D)
- F7 (F – A – C – Eb)
- G7 (G – B – D – F)
- A7 (A – C# – E – G)

2.1.1. Chord Major

Sebuah *chord major* terdiri dari sebuah *root*, sebuah *major third* dan sebuah *perfect fifth*. Sebagai contoh, Chord C major terdiri dari nada C, E dan G dengan E sebagai *major third* di atas C, dan G adalah *perfect fifth* di atas C [13].

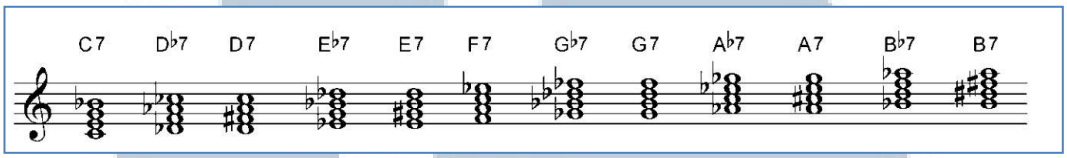


Gambar 2.2 major chords dari setiap nada [13].

2.1.2. Chord Dominant Seventh

Dominant seventh, kadang disebut pula sebagai "seventh", mengambil *major triad* dengan menambahkan sebuah *minor seventh* di atasnya. Dengan kata lain, *dominant seventh* adalah sebuah *chord major* dengan *seventh* yang diturunkan.

Jadi secara garis besar, *dominant seventh* terdiri dari sebuah *root*, *major third*, *perfect fifth* dan *minor seventh* [13].

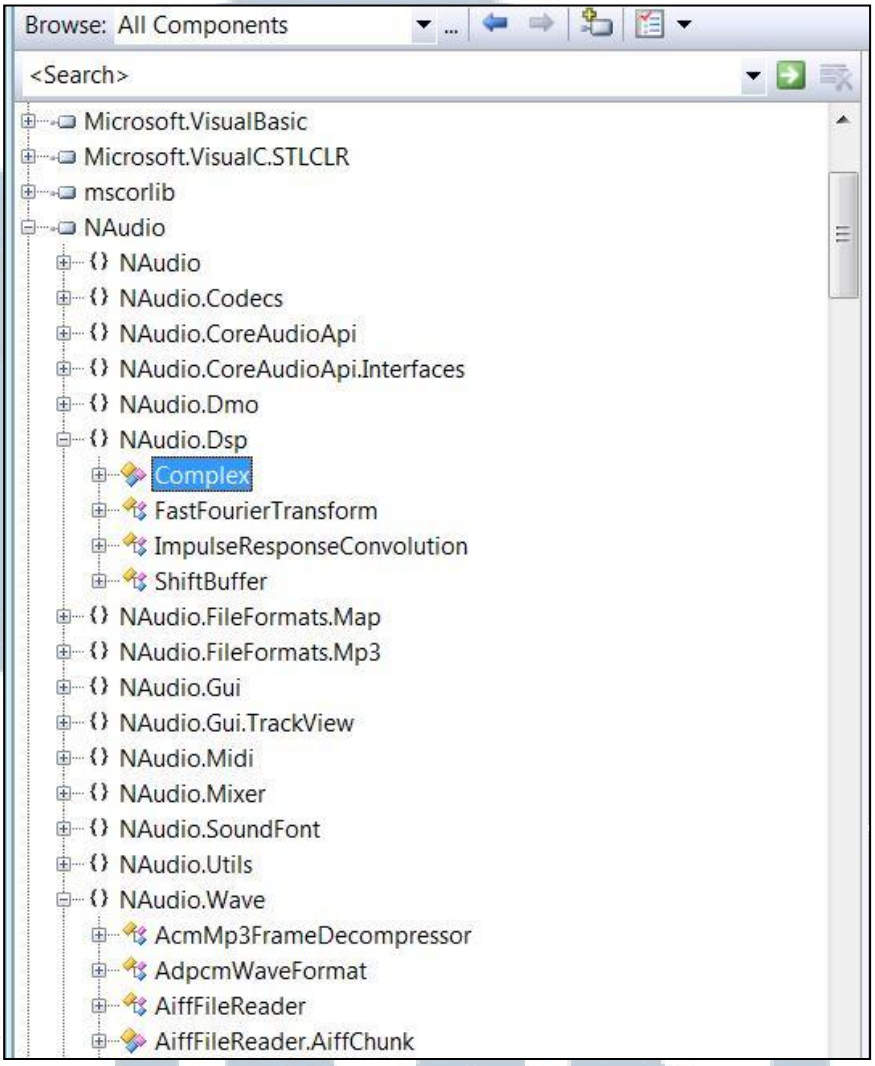


Gambar 2.3 *chord dominant seventh* dari setiap nada [13].

2.2 Library NAudio

Proses *coding* program menggunakan *Integrated Development Environment* (IDE) Microsoft Visual Studio C# 2008 dengan .NET Framework 4.0 yang sudah ter-*install*, serta menggunakan *library* yang menjadi kunci pembuatan proyek ini, yakni *library* NAudio (www. naudio.codeplex.com).

NAudio merupakan *library open source* untuk .NET audio dan MIDI, yang memiliki lusinan class yang berhubungan dengan *audio* yang bertujuan untuk mempercepat pengembangan *audio related utilities* pada .NET. *Library* ini telah dikembangkan sejak 2001 dan telah tumbuh dan melingkupi berbagai fitur [8].

N 

Gambar 2.4 *screenshot* library Naudio pada Visual C#

2.3 Thresholding

Proses ini melakukan proses sinyal input dengan mengambil nilai puncak-puncak dominan [7]. Pada bagian ini, sinyal akan mengalami *powering* terlebih dahulu dengan cara mengkuadratkan sinyal, yang dijelaskan dengan persamaan [7]:

$$p = \sum_{i=0}^{i=N} \sqrt{x_i{}^2} \quad \ldots\ldots (2.1)$$

Keterangan:

p = power sinyal (dB)

N = jumlah sample per frame

x_i = frame ke-i (dB)

i = sample ke-i

Menurut [7], *powering* bertujuan untuk mengetahui letak awal dan akhir dari suatu sinyal yang berada di suatu *frame*. Dari nilai *power* didapatkan suatu nilai rata-rata. Dengan menambahkannya dengan standar deviasi, maka didapatkan nilai awal dan akhir dari suatu frame.

Standar deviasi merupakan penyimpangan data standar dari data yang ada. Jika nilai data mendekati nilai rata-rata, maka standar deviasinya lebih kecil (mendekati nol) dan sebaliknya. Ditunjukkan dengan persamaan [7]:

$$Standar\ deviasi = \sum_{i=0}^{n} \frac{x^2{}_i}{n} \quad \ldots\ldots (2.2)$$

$$Rata - rata = \sum_{i=0}^{n} \frac{x_i}{n} \quad \ldots\ldots (2.3)$$

Sinyal dengan nilai *power* di atas standar deviasi diambil sebagai awal dan akhir sinyal serta dianggap sebagai suatu *voice*.

2.4 Frame Blocking

Frame-Blocking merupakan pembagian sinyal audio menjadi beberapa *frame* yang akan memudahkan dalam perhitungan dan analisa sinyal, di mana satu *frame* terdiri dari beberapa *sample* tergantung tiap berapa detik suara yang akan disample dan besarnya frekuensi *sampling*-nya [1]. Pada proses ini dilakukan pemotongan sinyal dalam slot-slot tertentu agar dianggap *invariant*. Berikut ini rumus yang digunakan untuk pengambilan sample tiap *frame* [1] :

$$SFr = Fs\ x\ \frac{t}{1000} \quad \text{.....(2.4)}$$

Keterangan:

SFr = *Sample* per *Frame*

Fs = Frekuensi *sample* / sample rate dari format *audio wav*

T = waktu pengambilan (ms)

2.5 Windowing

Proses *windowing* bertujuan untuk mengurangi timbulnya *spectral leakage* atau *aliasing* yang mana merupakan efek dari timbulnya sinyal baru yang memiliki frekuensi yang berbeda dengan sinyal aslinya. Efek tersebut terjadi karena rendahnya jumlah *sampling rate* atau proses *frame - blocking* yang menyebabkan sinyal menjadi *discontinue* [9].

Pada aplikasi ini jenis *window* yang digunakan adalah jenis *Hamming Window*. Digunakannya *Hamming Window* adalah karena *Hamming Window*

memiliki *side lobe* yang paling kecil dan *main lobe* paling besar sehingga hasil *windowing* akan lebih halus dalam mengurangi *spectral leakage* [11].

Persamaan *Hamming Window* [11]:

$$w(n) = 0.55 - 0.46 \times \cos\left(\frac{2 \times \pi \times n}{N-1}\right) \quad (2.5)$$

Di mana:

w(n) = *windowing*

N = jumlah data dari sinyal

n = waktu diskrit

2.6 Fast Fourier Transform

Untuk mendapatkan sinyal dalam domain frekuensi dari sebuah sinyal diskrit, metode yang digunakan adalah *Discrete Fourier Transform* (DFT). Yang menjadi masalah, DFT memerlukan waktu komputasi yang sangat panjang untuk data yang besar, oleh karena itu digunakan Fast Fourier Transform untuk mengimplementasikan DFT dengan cepat. FFT Ini mengubah masing – masing frame *sample* dari domain waktu menjadi domain frekuensi [9].

Transformasi Fourier Diskrit didefinisikan oleh rumus [1]:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \quad k = 0, \ldots, N-1 \quad (2.6)$$

Keterangan:

$x_0, \ldots, x_{N-1}$ = bilangan kompleks

Operasi di atas memiliki big – O sebesar $O(N^2)$. Sedangkan FFT hanya memerlukan operasi sebanyak O(N log N) untuk menghitung deret yang sama. Secara umum FFT bisa dikatakan lebih cepat dibandingkan DFT [1].

Output dari FFT ditransformasikan ke dalam rentang frekuensi. Nilai magnitude terhadap frekuensi didapatkan dari rumus [9]:

$$M(f) = \sqrt{RealOut[j]^2 + ImagOut[j]^2} \quad (2.7)$$

2.7 Pitch Class Profile dan Enhanced Pitch Class Profile

Pitch Class Profile dapat mendeteksi warna dari sebuah *chord* berdasarkan kelas pitch dari nada. PCP berawal dari penyajian frekuensi dari transformasi fourier (FFT) kemudian frekuensi dipetakan ke dalam 12 kelas pitch: E, F, G, G#, A, A#, B, C, D, D# [1].

Dari hasil FFT yang telah didapatkan dideteksi puncak-puncaknya. Parameter frekuensi dan amplitudo puncak sinyal menentukan pengkelasan dari pitch tersebut. Dengan didapatkan *range* frekuensi masing-masing class, maka puncak dapat dikelaskan menurut besar frekuensinya. Masing-masing frame sinyal akan memiliki jumlah puncak, frekuensi dan nilai PCP yang berbeda. Oleh karena itu, data *mean* PCP diambil sebagai rata-rata nilai EPCP agar memiliki nilai yang lebih akurat. Dari hasil deteksi puncak, maka dapat dicari pengelompokan *range* frekuensi nada dengan menghitung jarak awal dan akhir setiap nada [7].

Berikut penjelasan perhitungan awal dan akhir nada :

Awal nada ke –i = (frekuensi nada ke-(i-1) + frekuensi nada ke-i)/2

Akhir nada ke –i = (frekuensi nada ke-i) + frekuensi nada ke –(i +1)/2

Contoh perhitungan untuk nada A dari [7]:

Awal = (frek. Nada G2# + frek. nada A2) / 2

= (103.83 + 110.00) / 2

= 213.83 / 2

= 106.91

Akhir = (frek. nada A2 + frek. nada A2#)/2

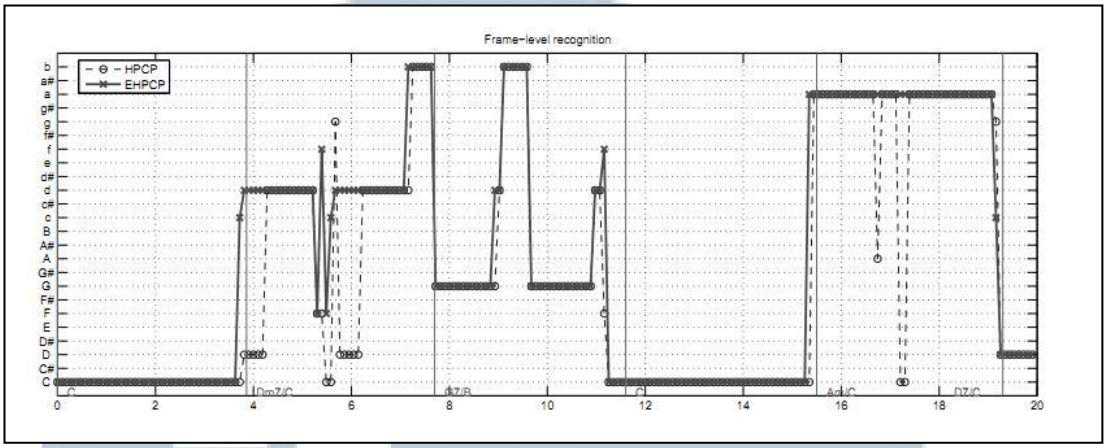
= (110.00 + 116.54) /2

= 226.54 / 2

= 113.27

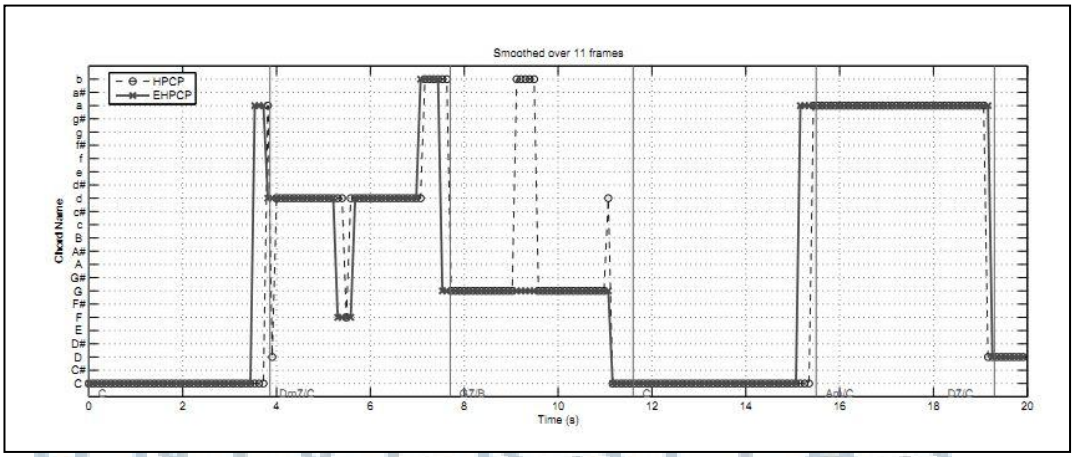
Kyogu Lee [4] membandingkan EPCP *feature vectors* dengan PCP konvensional menggunakan contoh rekaman asli. *File* audio di-*downsampled* ke 11025 Hz dan 36 *bins-per-octave constant* Q *transform* dijalankan antara *fmin* = 96 Hz dan *fmax* = 5250 Hz.

Berdasarkan Gambar 2.5, yang menunjukkan *frame-level recognition* dari 20 detik pertama lagu *Bach's Prelude in C Major* oleh Glenn Gould, jelas bahwa EPCP membuat error yang lebih sedikit daripada PCP.



Gambar 2.5 Hasil *frame-level recognition* dari sepenggal lagu *Bach's Prelude in C Major*[4]

Dan pada Gambar 2.6, yang merupakan hasil dari recognition dari smoothed PCP/EPCP *vectors* pada 11 *frames*, menunjukkan *smoothing process* mengurangi *errors* karena perubahan mendadak pada sinyal yang dapat mengaburkan komponen harmonik.



Gambar 2.6 Hasil recognition 11 *frames* yang di-*smoothing* dari sepenggal lagu *Bach's Prelude in C Major*[4]

2.8 Matching

Menurut [7], proses *matching* merupakan proses pemadanan data antara *database* dengan data masukan, agar dapat menghasilkan keluaran yang sesuai. Metode *matching* yang digunakan adalah *Sum Square Error*. Metode ini mencari *error* minimum dari perbandingan data *test* dan *train*. *Error* terkecil akan dianggap sebagai hasil.

Rumus *matching* menggunakan metode *Sum Square Error*:

$$SSE(X) = \sum_{k=1}^{m} (X_k - c(X))^2 \quad \text{.....(2.8)}$$

Keterangan:

SSE(X) = jumlah error input ke –x

X_k = input PCP nada ke – k

c(X) = nilai PCP pada database ke –X

2.9 Related works

Xinling Zhang dan David Gerhard [6] telah memberikan kontribusi untuk pendeteksian chord menggunakan *Pitch Class Profile*. Hasil dari sistemnya menghasilkan akurasi sebesar 75% untuk *chord major* dan *dominant seventh*. Berikut adalah hasil akurasi untuk chord *major* dan *dominant seventh* dari sistem dalam bentuk *confusion matrix* [6]

Tabel 2.2 *confusion matrix* milik [6] menunjukkan hasil akurasi *chord major* dan *dominant seventh* sebesar 75% [6]

Chord	Rate	C	C7	D	D7	E	E7	F	F7	G	G7	A	A7
C	40/40	40											
C7	35/40		35		2	2						1	
D	40/40			40									
D7	13/40			1	13		1	3	20				2
E	40/40					40							
E7	37/40						37				3		
F	38/40					1	1	38					
F7	5/40					3	16	16	5				
G	40/40									40			
G7	17/40				2					21	17		
A	30/40			1	8							30	1
A7	25/40	5	2								8		25

Berikut perhitungan pada confusion matrix sehingga mendapatkan akurasi 75%:

- *Chord* C memiliki rate 40/40, maka akurasinya adalah 100%.
- *Chord* C7 memiliki rate 35/40, maka akurasinya adalah 87,5%
- *Chord* D memiliki rate 40/40, maka akurasinya adalah 100%
- *Chord* D7 memiliki rate 13/40, maka akurasinya adalah 32,5%
- *Chord* E memiliki rate 40/40, maka akurasinya adalah 100%
- *Chord* E7 memiliki rate 37/40, maka akurasinya adalah 92,5%
- *Chord* F memiliki rate 38/40, maka akurasinya adalah 95%
- *Chord* F7 memiliki rate 5/40, maka akurasinya adalah 12,5%
- *Chord* G memiliki rate 40/40, maka akurasinya adalah 100%
- *Chord* G7 memiliki rate 17/40, maka akurasinya adalah 42,5%
- *Chord* A memiliki rate 30/40, maka akurasinya adalah 75%
- *Chord* A7 memiliki rate 25/40, maka akurasinya adalah 62,5%

Dari kumpulan akurasi masing – masing *chord* di atas, dapat dilakukan perhitungan persentasi keseluruhan:

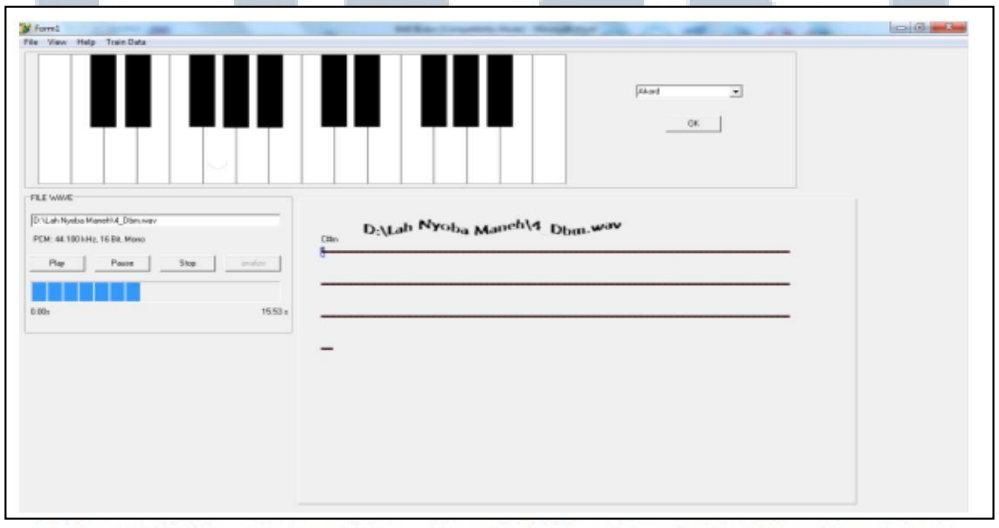
$$\text{Persentasi akurasi (\%)} = \frac{(\text{total akurasi (\%)})}{\text{banyak } chord}$$

$$= \frac{C + C7 + D + D7 + E + E7 + F + F7 + G + G7 + A + A7}{12}$$

$$= \frac{100 + 87{,}5 + 100 + 32{,}5 + 100 + 92{,}5 + 95 + 12{,}5 + 100 + 42{,}5 + 75 + 62{,}5}{12}$$

$$= \frac{900}{12} = 75$$

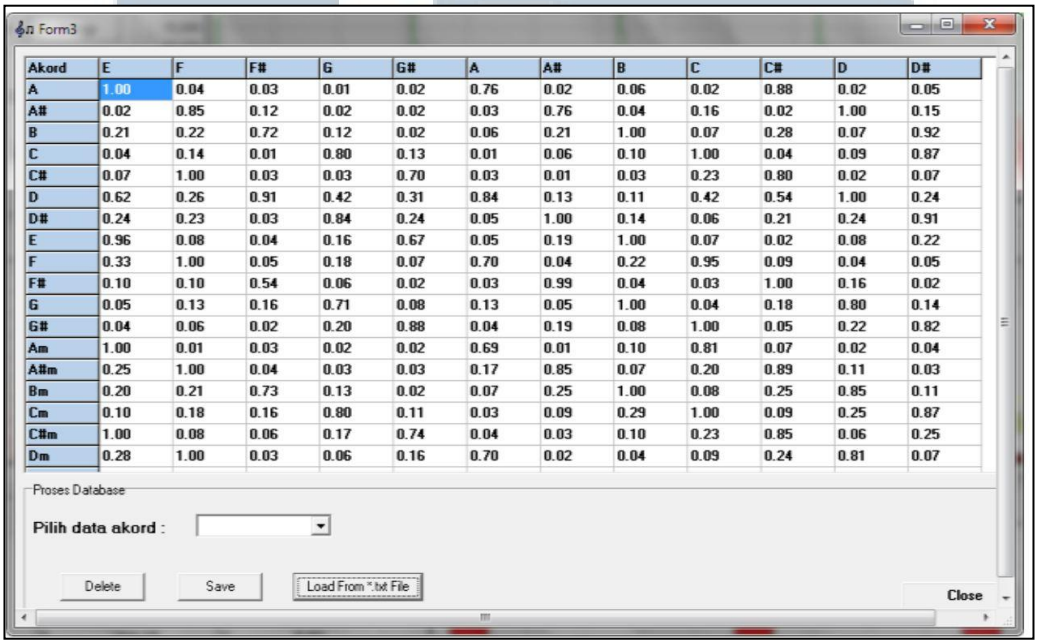
Metode Enhanced Pitch Class Profile telah pula digunakan oleh Nabila Azzahra Syahbani [7] untuk membuat sistem transkrip akord. *Chord* yang digunakan adalah *major* dan *minor*. Gambar 2.7 menujukkan *screenshot* hasil aplikasinya.



Gambar 2.7 sebuah *screenshot* aplikasi transkrip akord milik [7]

Milla Fitriani Kushidayati [11] membuat database transkrip akord yang juga menggunakan *Enhanced Pitch Class Profile. Chord* yang digunakan adalah *major* dan *minor*.

Berikut adalah *screenshot* hasil aplikasinya.



Gambar 2.8 sebuah *screenshot* aplikasi database transkrip akord milik [11]