**MỞ ĐẦU**

Trong cuộc sống hàng ngay, ở bất kì nơi nào, chúng ta đều có thể lắng nghe được những âm thanh xung quanh mình. Đặc biệt, con người là sinh vật cao cấp trong tự nhiên. Việc liên lạc, giao tiếp, biểu lộ cảm xúc của con người đều thông qua tiếng nói. Đó là dạng âm thanh đặc biệt được con người tạo ra qua bộ máy phát âm, tuân theo các đặc trưng của các hệ thống ngôn ngữ riêng biệt.

Các kĩ thuật, công nghệ liên quan tới nhận dạng, phân tích, tổng hợp tiếng nói...hiện nay đang trở nên hết sức cần thiết. Có rất nhiều ứng dụng hiện nay dựa trên các thành tựu trong xử lí tiếng nói như: nhận dạng danh tính cá nhân, bảo mật bằng giọng nói, điều khiển tự động thông qua giọng nói... Như vậy, chúng ta có thể thấy được sự cần thiết của việc tìm hiểu, nghiên cứu các công nghệ xử lí tiếng nói. Chính vì thế, nhóm đã lựa chọn tìm hiểu, nghiên cứu lí thuyết và kiểm thử một vấn đề trong kĩ thuật xử lí tiếng nói: Xác định khoảng lặng trong tín hiệu tiếng nói. Yêu cầu đó đòi hỏi sinh viên phải tìm hiểu nghiêm túc về cấu trúc của tệp tin định dạng WAV, các trường lưu trữ thông tin, các tham số liên quan tới việc xử lí tín hiệu tiếng nói. Ứng dụng của việc xác định khoảng lặng tiếng nói được áp dụng nhiều vào vấn đề bảo mật, xử lí tách lọc âm thanh hay xác định các đặc trưng giọng nói...

Trong suốt quá trình thực hiện bài tập, nhóm đã cố gắng thực hiện tốt nhất các yêu cầu đặt ra và mở rộng phạm vi kiến thức. Tuy nhiên, kết quả thực nghiệm và bản báo cáo chắc chắn không thể tránh khỏi những thiếu xót. Nhóm xin chân thành cảm ơn sự hướng dẫn, giúp đỡ tận tình của TS. Trịnh Văn Loan để hoàn thành tốt đề tài này.

*Nhóm sinh viên*

1. Cơ sở lí thuyết
2. Cơ bản về âm thanh số

Âm thanh là sự biến thiên về áp suất không khí của quá trình di chuyển trong bộ máy phát âm con người. Âm thanh số (số hóa) được miêu tả dưới dạng đồ thị của sự thay đổi áp suất không khí trong toàn bộ quá trình đó.

Để hiểu rõ hơn về vấn đề này, ta mở trình ứng dụng Windows Sound Recorder của hệ điều hành Windows và thu âm một đoạn âm thanh ngắn, sau đó quan sát đường màu xanh. Khi đường xanh này rộng, có nghĩa là áp lực không khí lớn, đấy là lúc mà tai con người cảm nhận rằng đó là âm thanh nhiễu (tạp âm). Khi đường xanh chạy dàn đều gần mức giữa (trục nằm ngang), ta hiểu gần như không có thay đổi về giá trị áp suất không khí, lúc đó tai con người cảm nhận đó là các khoảng lặng (gần như không có âm thanh, tiếng nói). Nếu đường màu xanh dao động lên xuống càng nhanh thì âm thanh tai con người nghe được càng lớn, càng rõ.

Khi chúng ta thu âm, microphone sẽ chuyển đổi sự biến thiên áp suất không khí thành dạng biến thiên của tín hiệu điện, đó là các thành phần mà các thiết bị âm thanh điện tử tính toán và biến đổi sang dạng số - gọi là các mẫu (samples). Khi ta nghe lại một đoạn nhac, đoạn âm thanh, quá trình xử lí sẽ được thực hiện ngược lại, các thành phần điện biến thiên sẽ được gửi tới thiết bị phát (loa, tai nghe, speakers...) chứ không phải là thiết bị thu (microphone), và được biến đổi ngược lại thành dạng áp lực của không khí bởi tai loa.

Tốc độ mà thiết bị xử lí âm thanh lấy mẫu tín hiệu điện được gọi là tần số lấy mẫu (sample rate) và được thể hiện theo đơn vị kilohertz (kHZ):

1kHZ = 1000 samples/sec

**Chú ý:** các file âm thanh số không hề chứa trực tiếp giá trị tần số âm thanh, trường độ, cường độ...Chúng ta cần phải áp dụng các thuật toán xử lí trên các mẫu (samples) để xác định giá trị chính xác của các mẫu tín hiệu âm thanh này.

1. Khái niệm tệp tin Wave

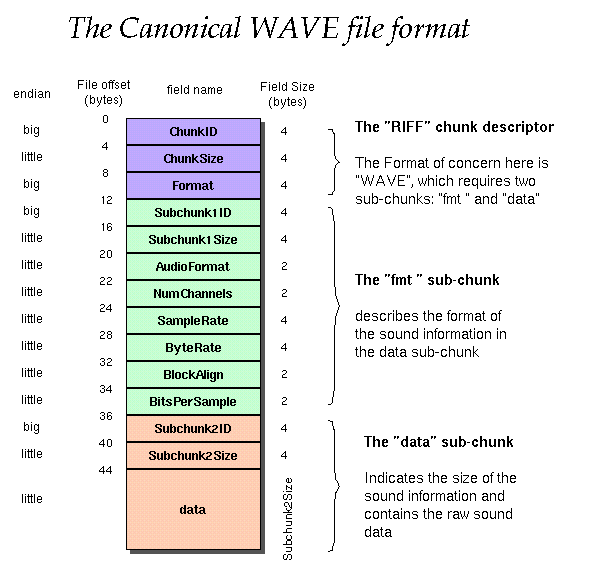
Trước hết, ta cần biết một vài thông tin về chuẩn RIFF (Resource Interchange File Format) – một dạng tập tin tài nguyên có thể chuyển đổi. RIFF là một định dạng tập tin cho phép lưu trữ nhiều loại dữ liệu, các loại dữ liệu đa phương tiên cơ bản như âm thanh (audio) hay hình ảnh, phim (video)... Đặc điểm của những tập tin thuộc chuẩn RIFF đó là nó sẽ nhóm nội dung của tập tin thành các khối riêng biệt và mỗi một khối sẽ gồm một header (dùng để qui định kiểu và kích thước của khối ) và các byte dữ liệu. Định dạng này dựa trên 2 thành phần chính là chunks và sub-chunks. Mỗi chunk sẽ có kiểu riêng biệt và được thể hiện bằng các tag có độ dài 4 kí tự. Chunk được mở đầu cho mỗi tập tin, bao gồm kích thước của chink và sau đó là nội dung của chunk. Một số định dạng file nằm trong chuẩn RIFF là: Audio/visual interleaved data (.AVI), Waveform data (.WAV), Bitmapped data (.RDI), MIDI information (.RMI), Color palette (.PAL), Multimedia movie (.RMN), Animated cursor (.ANI), A bundle of other RIFF files (.BND).

Tập tin Wave là một dạng tập tin dùng để lưu trữ dữ liệu âm thanh số (dạng sóng) và nó là một trong những định dạng phổ biến nhất của hệ điều hành Windows, thuộc chuẩn RIFF. Thông thường thì tập tin Wave có 2 dạng là dạng nén và dạng không nén (dạng chuẩn).

Trong công nghệ thu âm, định dạng Wave thường được dùng để thu âm thanh dạng thô, sau đó sẽ được xử lí và chuyển đổi sang các định dạng khác.

1. Cấu trúc tập tin Wave

Cấu trúc file wave gồm 3 khối: khối mô tả dạng RIFF, khối thuộc tính “fmt“ và khối dữ liệu “data” trong đó khối thuộc tính “fmt” và khối dữ liệu “data” là 2 khối con của khối mô tả dạng RIFF.



Hình 1 – Sơ đồ mô tả cấu trúc các trường thông tin của Wave

1. Khối mô tả dạng RIFF

Khối này xác định dạng RIFF và có kích thước là 12 byte gồm các trường:

* ChunkID:

Kích thước: 4 byte  
Chức năng: chứa chuỗi “RIFF” dưới dạng mã ASCII

* ChunkSize:   
  Kích thước: 4 byte  
  Chức năng: cho biết tổng kích thước của các trường sau nó.  
  (ChunkSize=4+(8+Subchunk1Size)+(8+Subchunk2Size))
* Format:  
  Kích thước: 4 byte  
  Chức năng: chứa chuỗi “WAVE”.

1. Khối thuộc tính “fmt”

Khối này xác định các thuộc tính của dữ liệu âm thanh và có kích thước là 24 byte gồm các trường:

* Subchunk1ID:   
  Kích thước: 4 byte  
  Chức năng: chứa chuỗi “fmt ”.
* Subchunk1Size:  
  Kích thước: 4 byte  
  Chức năng:cho biết tổng kích thước của các trường thuộc khối thuộc tính đứng phía sau trường này.  
  (đối với tập tin wave không nén thì Subchunk1Size bằng 16)
* AudioFormat:  
  Kích thước: 2 byte  
  Chức năng: cho ta biết dạng nén của dữ liệu trong tập tin wave.

|  |  |  |
| --- | --- | --- |
| **STT** | **Giá trị** | **Mô tả** |
| 1 | 0 | Không xác định |
| 2 | 1 | Không nén (PCM-Pulse Code Modulation) |
| 3 | 80 | MPEG |
| 4 | 49 | GSM 6.10 |
| 5 | 17 | IMA ADPCM |

Bảng thông tin về định dạng âm thanh

* NumChannels:  
  Kích thước: 2 byte  
  Chức năng: cho biết số kênh của tập tin wave.  
  (Mono=1,Stereo=2,v.v)
* SampleRate:  
  Kích thước: 4 byte  
  Chức năng: cho biết số mẫu trên 1 s và đây chính là tần số lấy mẫu của tập tin wave.
* ByteRate:  
  Kích thước: 4 byte  
  Chức năng:cho biết số byte trên 1 s ứng với tần số lấy mẫu trên.  
  (ByteRate=SampleRate\*NumChannels\*(BitsperSample/8))
* BlockAlign: == NumChannels \* BitsPerSample/8

Kích thước:2 byte  
Chức năng: cho biết số byte của 1 mẫu gồm tất cả các kênh.

* BitsPerSample: cho biết số bit trên 1 mẫu chỉ tính cho 1 kênh.  
  ( 8 bit=8,16 bit=16,v.v)

1. Khối dữ liệu “data”

* Subchunk2ID:  
  Kích thước: 4 byte  
  Chức năng: chứa chuỗi định danh “dataId”.
* Subchunk2Size:  
  Kích thước:4 byte  
  Chức năng: cho biết kích thước của dữ liệu âm thanh thô trong trường data.  
  (Subchunk2Size=NumSamples\*NumChannels\*BitsperSample/8)
* Data: trường data chứa các byte dữ liệu âm thanh và các byte dữ liệu này sẽ nhóm thành các mẫu,mỗi mẫu có 1 hoặc 2 kênh.Tùy vào số kênh của một mẫu(xác định bởi trường NumChannels) và số bit trên 1 kênh (xác định bởi trường BitsPerSample) mà ta sẽ xác định được số byte của 1 mẫu.

Bảng thông tin về dữ liệu

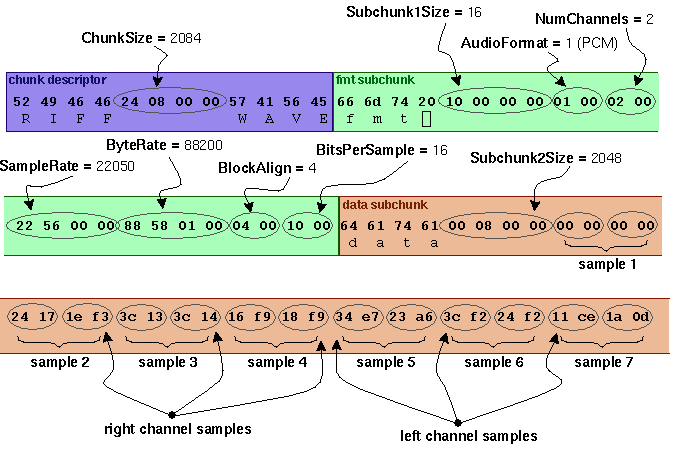
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Số kênh | Số bit/kênh | Kích thước 1 mẫu | Max | Min |
| 1 | 1(Mono) | 8 | 1 byte | 255 | 0 |
| 2 | 1(Mono) | 16 | 2 byte | 32767 | -32768 |
| 3 | 2 (Stereo) | 8 | 2 byte | 255 | 0 |
| 4 | 2 (Stereo) | 16 | 4 byte | 32767 | -32768 |

Một ví dụ về cấu trúc thông tin của một tập tin Wave 72 bytes được biểu diễn như sau:

52 49 46 46 24 08 00 00 57 41 56 45 66 6d 74 20 10 00 00 00 01 00 02 00

22 56 00 00 88 58 01 00 04 00 10 00 64 61 74 61 00 08 00 00 00 00 00 00

24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d

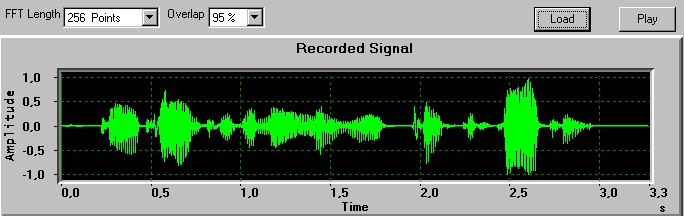


Hình 2 – Mô tả cấu trúc các trường dữ liệu Wave

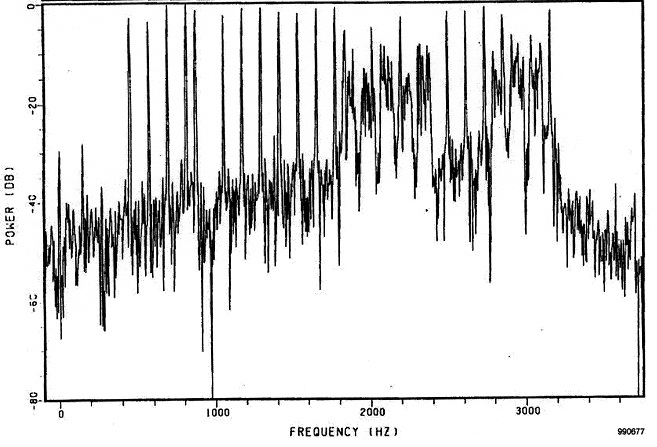
1. Một số đặc điểm

* Với định dạng thường mặc định thì thứ tự tổ chức dữ liệu trong Wabe được tổ chức theo kiểu đầu nhỏ (little-endian). Các tập tin sử dụng kiểu tổ chức bytes đầu to (big-endian) trong chuẩn RIFX thay cho RIFF.
* Các mẫu dữ liệu phải kết thức bằng một byte chẵn.
* Các mẫu 8-bit được lữu trữ với dạng bytes không dấu, nằm trong khoảng từ 0 – 255. Các mẫu 16-bit được lưu ở dạng các số nguyên có dấu, nằm trong khoảng từ -32768 – 32767.
* Có thể có thêm các subchunk mở rộng trong dòng dữ liệu Wave. Khi đó, mỗi subchunk sẽ bao gồm SubchunkID được biểu diễn bởi char[4] (kí tự có độ dài 4), SubChunkSize với kiểu số long có dấu dùng để lưu dữ liệu.

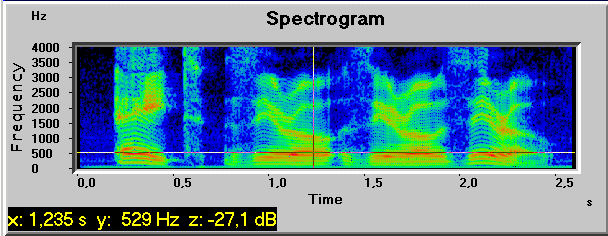
1. Một số dạng biểu diễn tín hiệu tiếng nói
2. Biểu diễn theo miền thời gian



1. Biểu diễn theo miền tần số



1. Biểu diễn dạng Spectrogram



1. Lập trình kiểm thử

Toàn bộ kết quả của chương trình kiểm thử được thực hiện với tập tin đầu vào chuẩn “A96.wav”

1. Đọc và hiển thị tín hiệu tiếng nói từ file .wav

Các hàm thực hiện việc đọc tập tin Wave nằm trong file “WaveFile.cs” của chương trình với namespace AudioUtils

(tổng quan toàn bộ về file -> wavefile.cs, mở + đọc ntn)

1. Đọc tập tin Wave từ ổ đĩa

* Sử dụng hàm Read() trong class WaveFile để đọc file theo đường dẫn do người dùng nhập vào:

waveFile = new AudioUtils.WaveFile(openFileDialog1.FileName);

waveFile.Read();

* Hàm khởi tạo một đối tượng WaveFile:

public WaveFile(String inFilepath)

{

m\_Filepath = inFilepath;

m\_FileInfo = new FileInfo(inFilepath);

m\_FileStream = m\_FileInfo.OpenRead();

m\_Riff = new Riff();

m\_Fmt = new Fmt();

m\_Data = new Data();

}

* Triển khai hàm Read():

public void Read( )

{

m\_Riff.ReadRiff( m\_FileStream );

m\_Fmt.ReadFmt( m\_FileStream );

m\_Data.ReadData( m\_FileStream );

}

1. Đọc các khối dữ liệu bên trong tập tin Wave

Như đã trình bày ở phần lí thuyết, cấu trúc tập tin Wave bao gồm ba khối chính. Chương trình thực hiện đọc dữ liệu tương ứng bai khối đó:

* Khối mô tả RIFF:

Dữ liệu của khối “RIFF” sẽ được xử lí trong lớp Riff

Các thuộc tính:

private byte[] m\_RiffID;

private uint m\_RiffSize;

private byte[] m\_RiffFormat;

Các phương thức:

public void ReadRiff( FileStream inFS )

{

inFS.Read( m\_RiffID, 0, 4 );

Debug.Assert( m\_RiffID[0] == 82, "Riff ID Not Valid" );

BinaryReader binRead = new BinaryReader( inFS );

m\_RiffSize = binRead.ReadUInt32( );

inFS.Read( m\_RiffFormat, 0, 4 );

}

* Khối thuộc tính “fmt”

Dữ liệu của khối “fmt” sẽ được xử lí trong lớp Fmt

Các thuộc tính:

private byte[] m\_FmtID;

private uint m\_FmtSize;

private ushort m\_FmtTag;

private ushort m\_Channels;

private uint m\_SamplesPerSec;

private uint m\_AverageBytesPerSec;

private ushort m\_BlockAlign;

private ushort m\_BitsPerSample;

Các phương thức:

public void ReadFmt( FileStream inFS )

{

inFS.Read( m\_FmtID, 0, 4 );

Debug.Assert( m\_FmtID[0] == 102, "Format ID Not Valid" );

BinaryReader binRead = new BinaryReader( inFS );

m\_FmtSize = binRead.ReadUInt32( );

m\_FmtTag = binRead.ReadUInt16( );

m\_Channels = binRead.ReadUInt16( );

m\_SamplesPerSec = binRead.ReadUInt32( );

m\_AverageBytesPerSec = binRead.ReadUInt32( );

m\_BlockAlign = binRead.ReadUInt16( );

m\_BitsPerSample = binRead.ReadUInt16( );

// This accounts for the variable format header size

// 12 bytes of Riff Header, 4 bytes for FormatId, 4 bytes for FormatSize & the Actual size of the Format Header

inFS.Seek( m\_FmtSize + 20, System.IO.SeekOrigin.Begin );

}

* Khối dữ liệu “data”

Dữ liệu của khối “data” sẽ được xử lí trong lớp Data

Các thuộc tính:

private byte[] m\_DataID;

private uint m\_DataSize;

private Int16[] m\_Data;

private int m\_NumSamples;

Các phương thức:

public void ReadData(FileStream inFS)

{

inFS.Read(m\_DataID, 0, 4);

Debug.Assert(m\_DataID[0] == 100, "Data ID Not Valid");

BinaryReader binRead = new BinaryReader(inFS);

m\_DataSize = binRead.ReadUInt32();

m\_Data = new Int16[m\_DataSize];

inFS.Seek(44, System.IO.SeekOrigin.Begin);

m\_NumSamples = (int)(m\_DataSize / 2);

for (int i = 0; i < m\_NumSamples; i++)

{

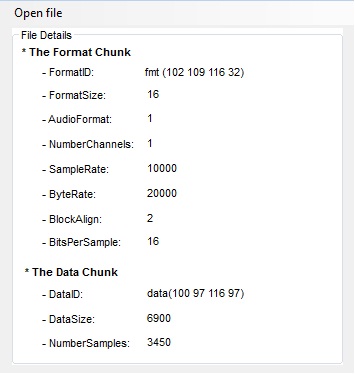
m\_Data[i] = binRead.ReadInt16();

}

}

1. Hiển thị thông tin các khối dữ liệu trên giao diện

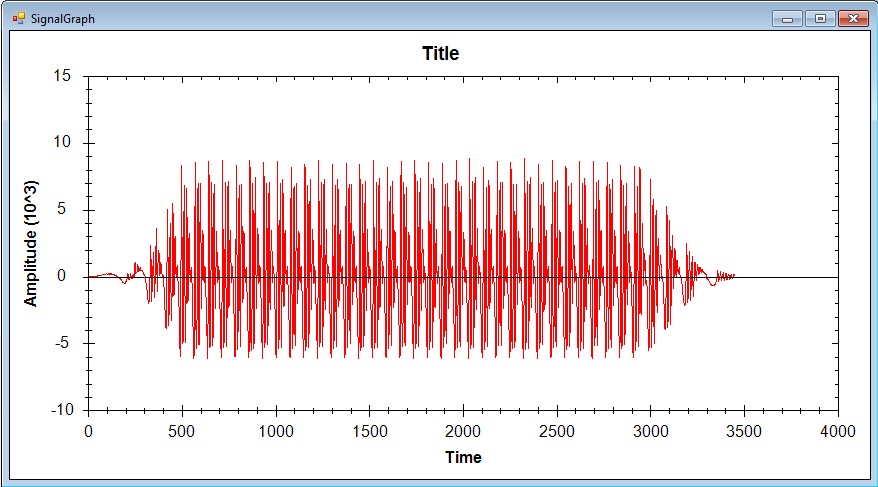
Sau khi người dùng nhập đúng đường dẫn tập tin Wave chuẩn thì tất cả các thông tin sẽ được đọc theo các phương thức như trên và được hiển thị ra màn hình chính:



Hình 3 – Giao diện chính hiển thị thông tin tập tin Wave

1. Hiển thị dạng tín hiệu tiếng nói trong miền thời gian:

Sau khi thực hiện quá trình đọc, ấn nút Show Signal Graph để xem hiển thị tín hiệu tiếng nói trong miền thời gian:



Hình 4 – Biểu diễn tín hiệu tiếng nói trong miền thời gian tập tin A96

1. Xác định khoảng lặng của tín hiệu tiếng nói theo thời gian
2. Ý tưởng thuật toán

* Bước 1: xác định chuỗi tín hiệu tiếng nói đầu vào x(n) trong miền thời gian
* Bước 2: nhân tín hiệu tiếng nói với hàm cửa sổ Hamming. Độ rộng cửa sổ Hamming khoảng 20-30 ms. Khi trượt liên tiếp các cửa sổ theo miền thời gian, 2 cửa sổ liên tiếp sẽ có phần trùm lên nhau khoảng 0< n < M/2 với M là số mẫu tín hiệu nằm trong một cửa sổ Hamming.

**Mục đích:** sử dụng cửa sổ Hamming thay cho cửa sổ hình chữ nhật nhằm làm giảm sai số, tránh đột biến ở 2 đầu cửa sổ.

* Bước 3: sử dụng công thức tính năng lượng tương ứng từng tín hiệu
* Bước 4: chọn 1 giá trị làm ngưỡng để xác định khoảng lặng:

+ Nếu các đoạn giá trị *W(n)* nào nhỏ hơn thì sẽ là các khoảng lặng.

+ Nếu các đoạn giá trị *W(n)* nào lớn hơn thì sẽ là các khoảng tín hiệu tiếng nói.

1. Quá trình thực hiện

Đường biểu diễn năng lượng

Hamming Windows

x(n)

* Hàm cửa sổ Hamming:

Tính chuỗi tín hiệu:

public float[] CalculateData()

{

float[] result = new float[Wide];

for (uint i = 0; i < Wide; i++)

{

result[i] = CaculateN(i);

}

return result;

}

Tính từng tín hiệu:

public float CaculateN(uint n)

{

return (float)(0.54 - 0.46 \* Math.Cos(2 \* Math.PI \* ((float)n / (Wide - 1))));

}

* Hàm tính năng lượng tín hiệu:

public void Process()

{

uint jumpStep = Parameters.HAMMING\_WINDOW\_WIDE - Parameters.COVERED\_WIDE;

ulong signalLength = (ulong)waveFile.WaveData.NumSamples;

float[] hammingArray = hammingWindow.CalculateData();

for (ulong i = 0; i < signalLength; i += jumpStep)

{

if (i > signalLength)

{

}

else

{

short[] tempSource = new short[hammingWindow.Wide];

Array.Copy(this.waveFile.WaveData.ConvertToArray(), (long)i, tempSource, 0, (long)hammingWindow.Wide);

double[] frame = Utility.MultipleSignal(tempSource, hammingArray);

if (enegryArray == null) enegryArray = new LinkedList<double>();

enegryArray.AddLast(Utility.CalculateEnegry(frame));

}

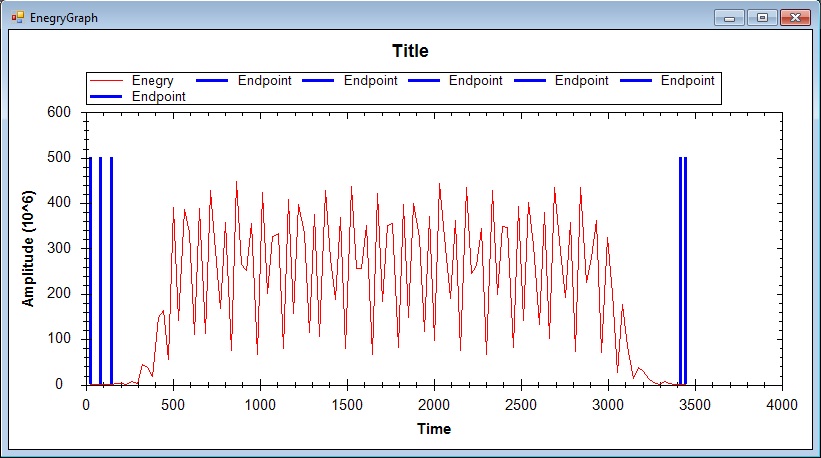
}

}

1. Biểu diễn đường năng lượng tín hiệu tiếng nói và xác định khoảng lặng

Như trên hình, chúng ta thấy được đường biểu diễn năng lượng tín hiệu tiếng nói màu đỏ.

Các cột màu xanh đánh dấu các khoảng lặng: đó là các khoảng có giá trị năng lượng tín hiệu tiếng nói nhỏ hơn ngưỡng (tạm chọn 600000)



Hình 5 – Biểu diễn năng lượng tín hiệu tiếng nói

và xác định các khoảng lặng

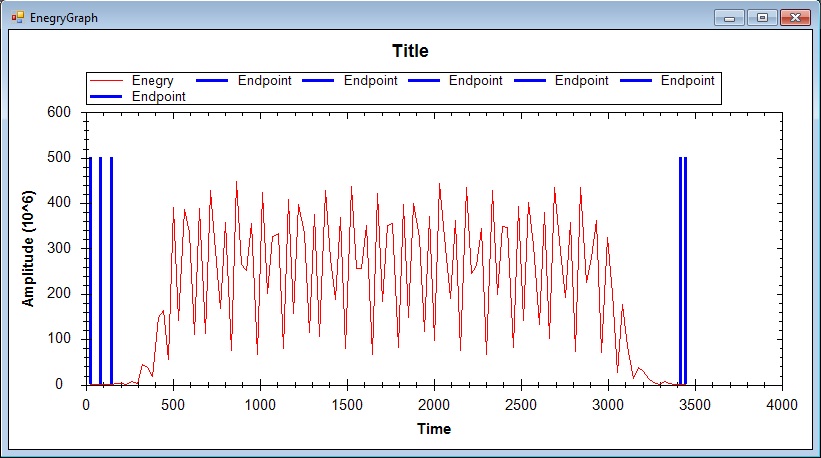
1. Xác định thời điểm xuất hiện tín hiệu tiếng nói và đánh dấu trên trục thời gian cùng với tín hiệu tiếng nói

* Nếu coi nhiễu là tín hiệu :

Trên hình biểu diễn tín hiệu tiếng nói trong miền thời gian, ta thấy có những tín hiệu với biên độ rất nhỏ ở 2 đầu. Đó là nhiễu. Nếu ta coi nhiễu đó là tín hiệu thì điểm bắt đầu của tín hiệu tiếng nói sẽ là điểm đầu tiên xuất hiện các giá trị đó.

* Nếu không coi nhiễn là tín hiệu :

Ngược lại, nếu ta không coi nhiễu là tín hiệu thì ta sẽ đưa ra một giá trị nào đó làm ngưỡng để xác định điểm đầu tiên có giá trị năng lượng lớn hơn ngưỡng. Từ đó, ta xác định được điểm bắt đầu của tín hiệu tiếng nói.



Hình 6 – Biểu diễn năng lượng tín hiệu tiếng nói và các điểm có giá trị năng lượng nhỏ hơn mức đường đánh dấu bằng đường màu xanh