Question:

For music robots, the reception of sound information is very important. But how do we retrieve the sound information?

Solution:

The answer is audio Capture Assistant – “ OpenAL”.

I have studied the use of openAL a long time, and finally succeeded in using a microphone to capture sound data. I am very touched (weeping).  
  
In contrast, openCV relatively simple, and openAL are not easy to get started. Thereason is that the program too much on the Internet, information is very fragmentary, I spent some time to understand them and organize. This time I retrieve voiceinformation and stored data in the txt file "to do a brief introduction of the code.

To normally use openAL, the following lead:  
1. Download the OpenAL SDK and install it:  
• official website download page:http://connect.creativelabs.com/openal/Downloads/Forms/AllItems.aspx  
• If your operating system is windows XP, please select OpenAL11CoreSDK  
  
2. VC + + link  
• Visual Studio -> Tool -> Options -> Projects and Solutions -> VC + + Directories -> Include Files Add the C: \ Program Files \ takes advantage of OpenAL, 1.1 the SDK \ include (depending on installation location)  
• Visual Studio -> Tool -> Options -> the Projects and Solutions -> VC + + the Directories -> Library Files to join the C: \ Program Files \ takes advantage of OpenAL, 1.1 SDK \ libs \ Win32 (depending on installation location and system)  
3. Create a project to link  
• Right-click on the project -> Configuration and the Properties -> Linker -> the Input -> Additional dependencies -> join "openal32.lib.  
After carrying out these steps, you can enjoy openAL.

|  |
| --- |
| /\*\*\*\*程式碼開始\*\*\*/ #include <cv.h>  #include <highgui.h>  #include <al.h>  #include <alc.h>  #include <iostream>  #include <stdio.h>  #include <windows.h>  #include <conio.h>  #define SRATE 44100 #define BUFFERSIZE 4410  using namespace std;  ALshort buffer[BUFFERSIZE]; ALint sample;   int main(int argc, char \*argv[]) { alGetError(); FILE \*file; file = fopen("test.txt","w");  ALCdevice \*device = alcCaptureOpenDevice(NULL, SRATE, AL\_FORMAT\_MONO16, BUFFERSIZE);  if (alGetError() != AL\_NO\_ERROR) { cout << "Error!" << endl; return 0; }  alcCaptureStart(device); const ALchar \*actualDeviceName; actualDeviceName = alcGetString(device, ALC\_DEVICE\_SPECIFIER);  cout << "actualDeviceName: " << actualDeviceName << endl; cout << "device: " << device << endl; system("PAUSE");  int line = 1;  while (!\_kbhit()) {  Sleep(10); alcGetIntegerv(device, ALC\_CAPTURE\_SAMPLES, 1, &sample);  if(sample >= 4410) { alcCaptureSamples(device, (ALCvoid \*)buffer, sample); cout << "sample = " << sample << endl; cout << "buffer[0] = " << (int)buffer[0] << endl; cout << "buffer[1] = " << (int)buffer[1] << endl; cout << "buffer[2] = " << (int)buffer[2] << endl; cout << "buffer[3] = " << (int)buffer[3] << endl;  fprintf(file,"==============LINE %d==================\n",line); for(int i = 0; i < BUFFERSIZE; i++) fprintf(file,"[%d] %d\n",i,(int)buffer[i]);  line++;  }  }  alcCaptureStop(device); alcCaptureCloseDevice(device);  return 0; } |

Next, I explain the important part.

 I have not yet a thorough understanding of openAL, there may be some places to understand not correct. If the article is wrong, please correct me.

**#define SRATE 44100**  
**#define BUFFERSIZE 4410**  
sample  rate 44100Hz is standard CD quality sound  
Why BUFFERSIZE = 4410?

Because I hope microphones store 4410 samples (about 0.1 seconds) into the buffer, so my buffer size is set to 4410 (the samples)  
In addition, each sample size is 2bytes, and therefore declare the buffer type for short.  
  
**ALCdevice \*device = alcCaptureOpenDevice(NULL, SRATE, AL\_FORMAT\_MONO16, BUFFERSIZE);**  
ALCdevice \* alcCaptureOpenDevice(const ALCchar \*devicename, ALCuint frequency, ALCenum format, ALCsizei buffersize);  
devicename: If you do not know the device name, set device name NULL ,and then the program will help you find he read the audio device.  
frequency: the sample rate.  
format: AL\_FORMAT\_MONO8/16，A represents a single microphone, a sample 8/16bits said.  
In addition there are AL\_FORMAT\_STEREO8/16, on behalf of the microphone array(for sound source location), a sample of 8/16bits said.  
buffersize: the buffer size

**alGetError()**  
This function returns the current error state and then clears the error state.  
  
**alcCaptureStart(device);**  
This function begins a capture operation  
  
**actualDeviceName = alcGetString(device, ALC\_DEVICE\_SPECIFIER);**  
 const ALCchar \* alcGetString(ALCdevice \*device, ALenum param);  
\*device: Memory address of the device  
param: Depends on what kind of information you want.  getting the address of the device \* device name. Input ALC\_DEVICE\_SPECIFIER is

**alcGetIntegerv(device, ALC\_CAPTURE\_SAMPLES, 1, &sample);**  
void alcGetIntegerv(ALCdevice \*device, ALCenum param, ALCsizei size, ALCint \*data );  
\*device: Memory address of the device  
param: Depends on what kind of information you want.  Getting

Input ALC\_CAPTURE\_SAMPLES is investigating \* the device has been the number of the sample.  
size: Get the number of byte buffer available. You can enter 1 or more. Enter 0, the program that the buffer is full, do not archive action.  
\* data: the data obtained is stored at this address. (The data on behalf of the microphone to collect the number of the samples)  
  
**alcCaptureSamples(device, (ALCvoid \*)buffer, sample);**  
void alcCaptureSamples(ALCdevice \*device, ALCvoid \*buffer, ALCsizei samples);  
\*device: Memory address of the device  
\*buffer: Buffer  
samples: Sampling number, obtain by alcGetIntegerv (...).

**alcCaptureStop(device);**  
**alcCaptureCloseDevice(device);**This feature is needless to say, write and it's on.  
The program illustrated here. I hope will be helpful to people new to openAL.  
Let's look at the microphone equipment; we use a total of two.  
The first one is the device  on  laboratory laptop, the middle of black small hole is the radio hole.

[](http://4.bp.blogspot.com/-fgYMcGAH8DQ/TqrlHgBN99I/AAAAAAAAABs/lZUHbZEOg7A/s1600/CIMG3075.JPG)  
The second is provided by Zhang Fengming headset.  
[](http://3.bp.blogspot.com/-HFUfjQQxAmQ/TqrlwEWaeZI/AAAAAAAAAB4/TUSm1Xx9kz4/s1600/CIMG3077.JPG)

Zhang Fengming headset connected to the microphone ASUS, Zhang Fengming headset is set to priority access to the device.  
[](http://3.bp.blogspot.com/-9b3g4ri7q0k/TqrmPwtfMnI/AAAAAAAAACE/6h33zUkH_Jw/s1600/CIMG3078.JPG)

Next is the test video to the microphone.  
Video link: http://musirobobo.blogspot.com/2011/10/openal.html  
  
The program functions: If the signals received by microphone is very small, shows a very SAD onion; if the signal is great, the singing onion.  
  
If I did not do anything when the surrounding environment is quiet, the screen displays a SAD Onion.  
When I clap when the microphone senses sound the moment, the screen display Singing Onion. Finally, the microphone and yelled, Singing Onion has continued for quite a while.  
  
Microphone data can be obtained.  
In order to complete the music robot, the next work is to analyze the sound data.

Question:

After obtaining sound information, how to identify the music?

Solution:

The main features of the music: pitch, timbre, the volume.  
Pitch refers to the frequency of the sound; sound means the sound characteristics (appearance); volume is the loudness of sound.  
I use pitch and volume to distinguish between the different music.  
  
Our method to analysis music is the familiar Fourier transform.

In [mathematics](http://en.wikipedia.org/wiki/Mathematics), a Fourier series decomposes [periodic functions](http://en.wikipedia.org/wiki/Periodic_function) or periodic signals into the sum of a (possibly infinite) set of simple oscillating functions, namely sines and cosines (or [complex exponentials](http://en.wikipedia.org/wiki/Complex_exponential)).

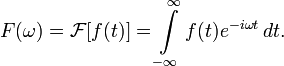
[http://1.bp.blogspot.com/-QO2HTZnNZCI/TuF5WYXkdHI/AAAAAAAAACY/HowlhlNyL2I/s320/63300210376f639deb5ca350fc7fc804.png](http://1.bp.blogspot.com/-QO2HTZnNZCI/TuF5WYXkdHI/AAAAAAAAACY/HowlhlNyL2I/s1600/63300210376f639deb5ca350fc7fc804.png)

In our case, f (x) is the music input, a0, a1, a2, ...; b1, b2, ... is the weights of the various frequency waveform weight  
The music is composed by many different frequencies of sound  
If you know the weight of the various frequency waveform, you can find the main frequency of the music.

To continuous information

[http://2.bp.blogspot.com/-USnMvnTwlr0/TuF-cUItpgI/AAAAAAAAACw/O9crlGuNhM4/s320/CIFT.png](http://2.bp.blogspot.com/-USnMvnTwlr0/TuF-cUItpgI/AAAAAAAAACw/O9crlGuNhM4/s1600/CIFT.png)

f(t) is the music; F(w) is the signal of the various frequency waveform

We know that the combination of general signal can be decomposed into many frequency signal,  
So how to find out the weight of the various frequency signals. To achieve this goal, we rely on the Fourier Transform.  
Continuous signal the FT :  
[](http://3.bp.blogspot.com/-Y-o_fBnmTBM/TuGGDqthZlI/AAAAAAAAAC8/FBzerETUUDg/s1600/CFT.png)

 Discrete signal the FT :  
[http://1.bp.blogspot.com/-CAO6Hm9pL68/TuGGenTz8qI/AAAAAAAAADI/LT1UECBIytI/s320/DIFT.png](http://1.bp.blogspot.com/-CAO6Hm9pL68/TuGGenTz8qI/AAAAAAAAADI/LT1UECBIytI/s1600/DIFT.png)

Computer music is a discrete signal, so we use the discrete Fourier transform,  
However, the DFT (Discrete Fourier the transform) computation process is too cumbersome,  
Because the computation time is too long, so the Fast Fourier Transform was invented, the algorithm can only be used in the discrete data (digital data).  
  
How fast is the FFT ?  
N sets of data, the processing time ratio of N: log N  
To a 1024 set of data, you can save about 100 times.  
Our music is a 4410 set of data, save more time.

I used FFT library is FFTW  
(A calculation of the DFT (Discrete Fourier Transform) C library, is recognized worldwide as the execution speed of the fastest Fourier transform software. )  
The software introduction and instruction please refer to the following URL:  
http://blog.chinson.idv.tw/2006/05/fftw-fft-c.html  
  
The demo video link - music fft

<http://musirobobo.blogspot.com/2011/11/fast-fourier-transform.html>

<Description>  
Horizontal axis: frequency (the more right position serve as the greater frequency)  
Vertical axis: amplitude (the higher the pillars, on behalf of the frequency with the more ingredients in logarithm)  
  
<Video>  
At the outset, the surrounding environment is quiet, each frequency component is about the same (noise)  
Later, I played two songs, you can see the status of the analysis of the frequency of the music change.  
  
  
The following attached DEMO movie software code (useful to some openCV)

|  |
| --- |
| /\*\*\*\*程式碼開始\*\*\*/ #include < cv.h > #include < highgui.h > #include < al.h > #include < alc.h > #include < iostream > #include < stdio.h > #include < windows.h > #include < conio.h > #include < fftw3.h > #include < math.h >  #define BUFFERSIZE 8820 #define Nr 4410 // # real number #define Nc floor( (double)Nr/2.0 )+1 // # fourier number  using namespace std;  const int SRATE = 44100; // sampling rate. means 44100 samples per second (CD quality) const int SSIZE = 4410; // the size of the ring buffer, Sampling-Rate \* Seconds \* Resolution \* Trackse  ALshort buffer[BUFFERSIZE]; // ALbyte: signed 8-bit 2's-complement integer ALint sample; // ALint: signed 32-bit 2's-complement integer  int main(int argc, char \*argv[]) { int i,j,k,l;  double \*FT\_in; fftw\_complex \*FT\_out; //直角坐標 double \*FT\_Amp; //極座標大小 fftw\_plan FT\_plan;  FT\_in = (double\*) fftw\_malloc(sizeof(double) \* Nr); FT\_out = (fftw\_complex\*) fftw\_malloc(sizeof(fftw\_complex) \* Nc); FT\_Amp = (double\*) fftw\_malloc(sizeof(double) \* Nc); FT\_plan = fftw\_plan\_dft\_r2c\_1d(Nr, FT\_in, FT\_out, FFTW\_ESTIMATE);   IplImage \*image1; image1 = cvCreateImage(cvSize(1103,500),IPL\_DEPTH\_8U,3); // FFT Transform  alGetError(); // This function returns the current error state and then clears the error state  ALCdevice \*device = alcCaptureOpenDevice(NULL, SRATE, AL\_FORMAT\_MONO16, SSIZE);  if (alGetError() != AL\_NO\_ERROR) { cout << "Error!" << endl; return 0; } alcCaptureStart(device);  const ALchar \*actualDeviceName; actualDeviceName = alcGetString(device, ALC\_DEVICE\_SPECIFIER);  cout << "actualDeviceName: " << actualDeviceName << endl; cout << "device adress: " << device << endl; system("PAUSE");  while(1) { Sleep(1); alcGetIntegerv(device, ALC\_CAPTURE\_SAMPLES, 1, &sample);  if(sample >= 4410) { alcCaptureSamples(device, (ALCvoid \*)buffer, sample);  for(i = 0; i < 4410; i++) FT\_in[i] = buffer[i]; fftw\_execute(FT\_plan); for(i = 0; i < Nc; i++) { FT\_Amp[i] = sqrt( pow(FT\_out[i][0],2) + pow(FT\_out[i][1],2) ); // amplitude of polar coordinate }  cvSetZero(image1); for(int j = 0; j < image1->widthStep; j+=3) { k = j/3; l = (float) log10(FT\_Amp[k]/10.0+1)\*80.0;  for(int i = image1->height -1 - l; i < image1->height; i++) {  image1->imageData[i\*image1->widthStep + j ] = 255; image1->imageData[i\*image1->widthStep + j +1] = 255; image1->imageData[i\*image1->widthStep + j +2] = 255; } }  cvNamedWindow("Freq",1); cvShowImage("Freq",image1);  if(cvWaitKey (1) == 'q') break; }  }  alcCaptureStop(device); alcCaptureCloseDevice(device);  cvDestroyWindow("Freq"); cvReleaseImage(&image1);  return 0; } |

Question:  
Sound data received by the microphone, and then we use the FFT to turn it into a spectrum data.  
We can observe from the film that there are a lot of noises in the converted spectrum.  
What causes this result? What are the solutions?

Solution:  
The microphone receives sound information, in addition to music, but also receives the noise in the environment.  
Therefore, I intend to eliminate the noise impact. So, I hope to read audio data directly from wav music files.  
The specific method is that, while playing a wav music while using the spectrum analyzer to analyze the data displayed.  
However, I can not use openAL broadcast wav and at the same time capture the wav internal information. So I have to use noisy sound to do analysis.  
After all, the music robot is to receive external sound to interact with the outside world, and therefore focus on how to do music analysis under noise interference action.

Question:  
How to make the robot to interact with the music?  
Solution:  
The original idea was to find the rhythm of the music, and then let the robot make excited to moderate actions with the rhythm of the music speed to.

In music rhythm section, I compare the current spectrum with the spectrum of the first few seconds in order to find the same tone interval, and then find out the rhythm of the music. However, I found that the noise of the microphone is large. The same musical passages maybe have very different spectrums. So we give up to find the rhythm of the music; instead, we find the pitch and volume of the sound as the music feature.

Find the volume of the sound is very simple, but there are a lot of noise in the real environment, the voice of people can easily cause the volume error of judgment, and to allow the volume to surge because the microphone moving.Therefore, a quiet environment is very important.  
The next problem is how to find the pitch of the sound?  
The concept is picking out the most important frequencies in the spectrum analysis. But a lot of noise in the environment, our microphone noise is very sensitive.  
In addition, the music contains many musical instruments and a lot of treble and bass.  
Spectrum after many signal processing was able to accurately determine the frequency of the sound.  
Therefore, we selected music source is the whistle.