讓視障者用聽覺建構圖像

Constructing Scenes by Hearing of the Visual Disabilities

Dyi-Rong Duh
Department of Computer Science and Information Engineering
National Chi Nan University
drduh@csie.ncnu.edu.tw

Chia-Wei Lu
Department of Computer Science and Information Engineering
National Chi Nan University
s1321008@ncnu.edu.tw

Kai-Hsun Wang
Department of Computer Science and Information Engineering
National Chi Nan University
s1213020@ncnu.edu.tw

摘要

視覺是人類感受外在環境最重要的方式，這方面對視覺障礙者是非常的不方便的，因為他們得要用其他的感官去感覺外在的世界，所以我們期望發展一套工具幫助盲人用聽覺去感覺物體的輪廓然後猜出是什麼東西。此外，如果他們能感覺到危險物體的接近，也可以讓他們更安全。我們基本的想法是從左到右，一行一行的將輪廓轉成不同頻率的聲音。在本論文中，我們在個人電腦上模擬和驗證此想法。我們發現實驗結果非常有趣。

關鍵詞：視覺障礙，頻率，影像輪廓，視覺輔助裝置

Abstract

Vision is the most important way for people to sense the external environment. The visual disabilities are so inconvenient that they could only use their other senses to feel out the outside world. We present a technique that helps the visual disabilities to sense curves of an object by their hearing. By hearing the sound, they could guess what the object might be. Furthermore, if they could sense the object closing to them, it might help them to keep the danger away. The basic idea of this paper is to traverse the image caught by a webcam from left to right per column at a time and produce the sound of different frequencies according to the position of the sketch. In this paper, we simulate and verify our idea on PC, and the experimental result is very interesting.

Keywords: Visual disability, frequency, image sketch, assistive vision device

1. Introduction

Eyes are vital organs of sight and the important sense of human for observing the external environment. The visual disabilities (people with seeing loss or impairments) have no vision and are so inconvenient because they could only use their other senses to feel out the outside world. Fortunately, in general, they have very sensitive hearing, smelling, and touch. They learn everything through hearing and touch. The most desirable way to provide information to pedestrians with visual disability that is to provide assistive devices with audible information. Our goal is to make an assistive vision device (a low cost embedded system) to help the visual disabilities to construct the scenes by their hearing as Figure 1 illustrated. However, in this paper, we only simulate our work on PC. In our developed system, we first

use a webcam to take a picture from outside world into PC using the program derived by Mbogho [4]. Then we extract the sketch image from the picture we taken [9] and examine the image to produce a stereo sound. The dataflow of our work is shown in Figure 2. Although at this time we implement all our work on PC to simulate and evaluate our idea, it can be transplanted into an embedded system with low cost and low energy constrains in the near future.

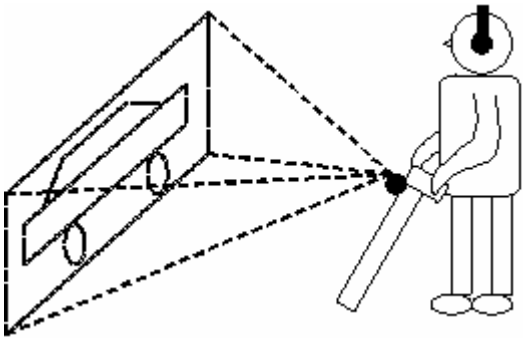

Figure 1. Our device to help the visual disabilities.

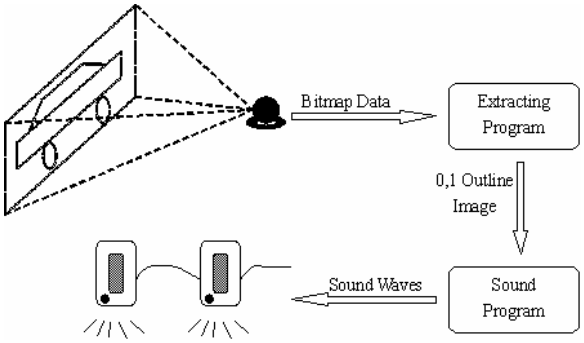

Figure 2 The dataflow of our system.

The original idea of drawing image sketch by sounds of different frequencies comes from Yeh and Lin [8]. In their work, they use Intel 8051 microcontroller and Yamaha YM2413B sound IC to implement an embedded system for scanning images by sound. They send the sketch data from the notebook to the embedded system via the parallel port and make the sound IC to produce the mono sound. They also developed some efficiently algorithms in frequency domain to improve the judgment of listener. What we do here is to make a new system simulated on PC to further examine and evaluate their idea. The experimental results of this work show that the idea is valuable and attractive.

This paper is organized as follows. In the next section, we describe the principle and process of our work. In section 3, the experimental results of our implementation are stated. Finally, we conclude our result and list some tasks for the further research in Section 4.

2. Principle and Process

We explain our work step by step as we go through the paper. First, we will tell the idea of transformation form image to sound in detail. Then we will talk a little bit about BMP format. After that, we will see how the extracting program works to extract sketchs from the image. Next, we will use DirectSound API [5] to produce stereo sound according to the sketch image.

2.1 The Idea

How exactly do we transform an sketch image to sound? At first, we preset sounds of different frequencies to each row within the image. From the bottom row to the top row are sounds of low frequency to high frequency. We then scan the image from left to right per column at a time. The wave of corresponding frequency on each row is sounded out if the scanning column intersects with the sketch at the corresponding row. For example, the sound of sketch image in Figure 3 would be like a wave of low frequency at first, the frequency then gradually increases and ends in a wave of high frequency. By this mechanism, some simple objects, such as line, triangle, and rectangular, can be easily present and recognized by one without any extra training. Notice that circle and ellipse are very difficult to recognize by anyone.

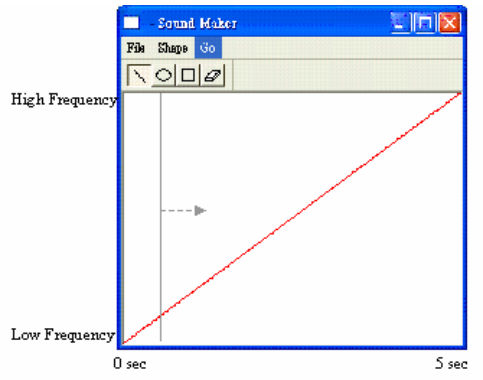

Figure 3. A line from bottom to top.

2.2 BMP Format

The BMP file [2], [3] is a bitmap image format developed by Microsoft Corporation. It is stored in a device independent bitmap (DIB) format, so that this format can display an image on any type of display device.

The BMP contains two parts; one is the header that contains the information of the bitmap, and the other is the bitmap data. In bitmap data, the scan line are stored from bottom up, so the beginning pixel of the image is in the lower-left corner of the bitmap and the last pixel is in the upper-right corner of the bitmap. In other words, an image in bmp format is stored from left to right and from bottom to top. Furthermore, if each pixel is represented by 24 bits red-green-blue (RGB) it would be changed to blue-green-red order. We use Figure 4 as an example to show the content of the bitmap file. The bitmap data of Figure 4 is illustrated in Table 1.

Figure 4. A sample BMP image.

Table 1.The BMP Data of Figure 4

Offset	BMP Data (in hexadecimal)
0x0000	42 4D 36 84 03 00 00 00
0x0008	00 00 36 00 00 00 28 00
0x0010	00 00 40 01 00 00 F0 00
0x0018	00 00 01 00 18 00 00 00
0x0020	00 00 00 84 03 00 00 00
0x0028	00 00 00 00 00 00 00 00
0x0030	00 00 00 00 00 00 FF 5A
0x0038	BB FF 59 BA FF 5D C2 FF
0x0040	5B C0 FF 61 BE FF 65 C2
0x0048	FF 63 C4 FF 58 B9 FF 5F
. . .	. . . . .

We can get some information from the Table 1 and describe them as follows:

- ◆0x0000-0x0001: 0x42 and 0x4D are the ASCII characters “B” and “M”.
- ◆0x0002-0x0005: 0x00038436 (= 230454) means that the file size is 230454 bytes.
- ◆0x000a-0x000d: 0x00000036 (= 54) means that that bitmap data start from offset “0036h”.
- ◆0x000e-0x0011: 0x00000028 (= 40) means that the header size is 40 bytes.
- ◆0x0012-0x0015: 0x00000140 (= 320) means that the image’s horizontal length is 320 pixels.
- ◆0x0016-0x0019: 0x000000F0 (= 240) means that the image’s vertical length is 240 pixels.
- ◆0x001c-0x001d: 0x0018 (= 24) means that there are 24 bits per pixel.
- ◆0x0036-: They are the bitmap data from 0x0036 to the end of the file.

According to the Header information, we can easily know the detail of the image, for example, the “bitmap data offset” can help us to catch the actually image data, the “Width” and “Height” show us how many pixels the image has, and we can catch “Bits per pixel” to decide whether we should transform it

to 8 bits per pixel or not because we just use 8 bits per pixel and gray scale image in our simulation.

2.3 Extracting a Sketch

In order to extract the sketch of an image effectually, we first transform the original color 24 bits image to an 8 bits gray scale image; then we calculate where the color tones has a large change and record it. These two steps are stated in the following.

Step 1. (Transforming a color image to a gray scale image): We calculate the average value of each pixel's red, green, and blue values as a gray scale value. Although it is not the actually gray scale value in this way, the result is not too bad. We must consider the performance if we use a complex algorithm to calculate the gray scale value, because it must reaction in time when we implement it in an embedded system with performance constraint.

Step 2. (Extracting the sketch): There are many ways to perform edge detection. However, the most may be grouped into two categories, gradient and Laplacian. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. We apply gradient method to calculate the variation of the gray scale value [10]. If one pixel's gradient value is larger than a threshold, we would record it as the image sketch. In mathematics, a gradient value is equal to $\sqrt{f_x^2+f_y^2}$, where $f_x = f(x+1, y) - f(x, y)$, $f_y = f(x, y+1) - f(x, y)$, and $f(x, y)$ is the gray scale value of the pixel at coordinate (x, y).

We use Figure 5 as an example to test our algorithm. The result image is shown in Figure 6 and Figure 7.

Figure 5. The original image from a webcam.

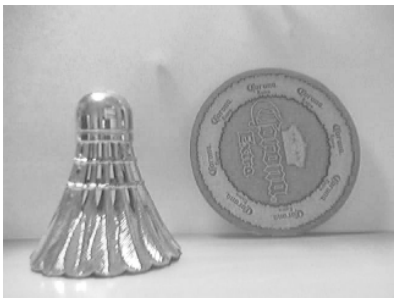

Figure 6. The result of Step 1 from Figure 5.

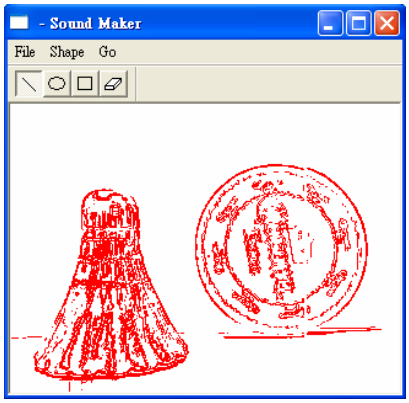

Figure 7. The result of Step 2 from Figure 6.

It is important to decide an appropriate threshold for extracting the sketch because we would get a complicated sketch when using a too small threshold and get an incomplete sketch when using a too big threshold. We use following figures as examples to illustrate our meaning. The original image is shown in Figure 8 and other three figures are the results after extracting. The threshold of Figure 9 is 50, the threshold of Figure 10 is 150, and the threshold of Figure 11 is 250.

Figure 8. The original image 1.

Figure 9. Threshold = 50.

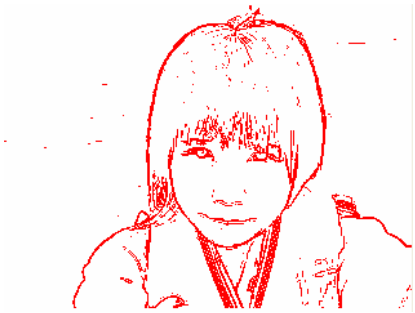

Figure 10. Threshold = 150.

Figure 11. Threshold = 250.

We need an uncomplicated but complete sketch to display the sound of image, but it is difficult to decide an appropriate threshold for every landscape picture. For example, the testing in Figure 8, if the threshold is 50 (as shown in Figure 9), it will extract a complicated sketch; if the threshold is 150 (Figure 10), it will extract a good sketch. In the following, the image shown in Figure 12 is tested. If the threshold is 50 the result is uncomplicated as shown in Figure 13. The result is incomplete as shown in Figure 14**,** if the threshold is 150.

As described above, we know that if the boundary chromatic aberration of the image's is not very explicit then the threshold should be low for extracting the complete sketch. Otherwise, the threshold should be high for extracting the uncomplicated sketch. This is why the threshold is difficult to decide.

Figure 12. The original image 2.

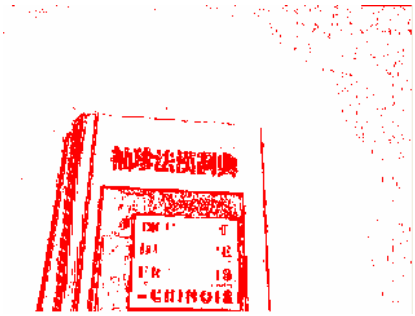

Figure 13. Threshold = 50.

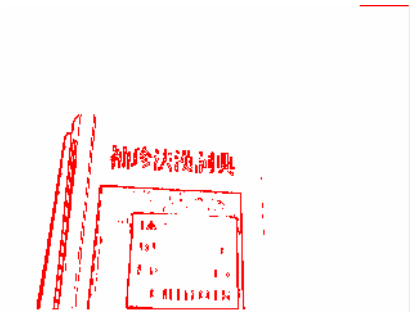

Figure 14. Threshold = 150.

In our solution, we use standard deviation to represent the similarity degree of the image's hues. We have two steps that are stated in the following.

Step 1. (Finding an appropriate threshold): We calculate the standard deviation of the pixels value in an image. Let n be the number of pixels, x_i be the value of a pixel i, $1 \le i \le n$, and $\overline{X}$ be the mean of pixels value of the testing image. The standard deviation S is stated in Equation (1).

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{X})^2}$$

$$= \sqrt{\frac{1}{n-1}((\sum_{i=1}^{n}x_i^2) - n\,\overline{X}^2)} \qquad (1)$$

Based on Equation (1), we could determine an appropriate threshold by analyzing the standard deviation. According to our experiments shown in Figure 15, we find that the standard deviation value is direct proportion to the threshold. So in our assumption, we recommend a small threshold for a small standard deviation value and vice versa, but there are still some exceptions.

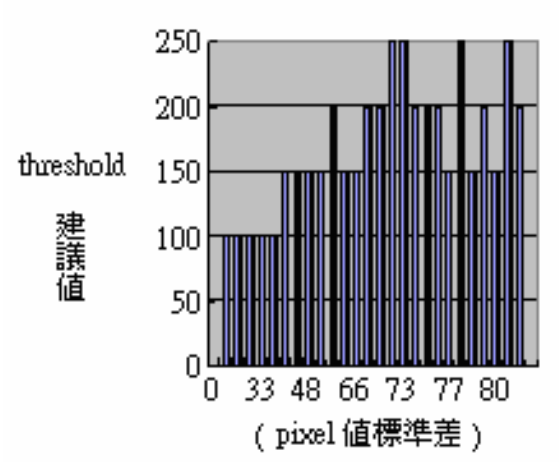

Figure 15. Our experiments data.

Step 2. (Simplifying the image): If one pixel is surrounded by up, down, left, and right four pixels, we name that pixel the inner pixel and mark the inner pixel CAN'T-DELETED. Then for each inner pixel, we delete surrounding four pixels if they are not marked CAN'T-DELETED. As shown in Figure 16, after simplification process we proposed, red pixels are those being preserved and black pixels are those being deleted. Thus, as we can see, only the framework of the picture is preserved. Repeat this process until no more pixels are deleted during a process.

Figure 17 is a sample image for testing our simplification method. The resulting image is shown in Figure 18. The result seems good for our simulation because the framework of picture is preserved. In words, the resulting image has less unnecessary pixels around the sketch. Consequently, it would be more easily to listen to these pixels by sound.

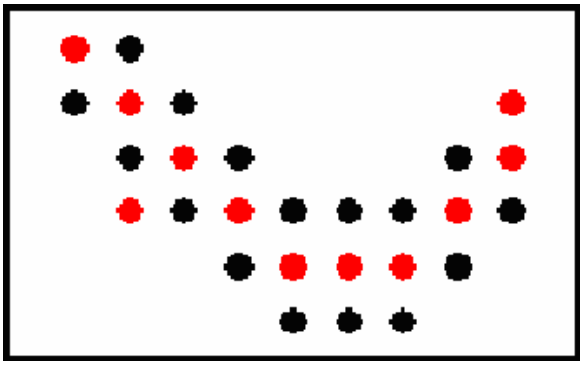

Figure 16. Our simplified method.

Figure 17. Image for testing simplified method.

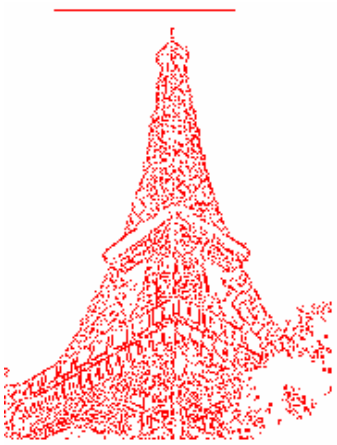

Figure 18. Simplified result of Figure 17.

2.4 Making Sound

We choose DirectSound API which is a part of DirectX to make the computer sound [5]. In DirectSound, every sound must first be written into buffer to be sounded out. There are two kinds of buffers primary and secondary. User could only have one primary buffer and the sound inside the primary buffer is the sound coming out from speaker at last. Normally user doesn't access the primary buffer directly. Instead, we modify secondary buffers. It is very easy for DirectSound to combine various different sounds. User only needs to write different sounds into different secondary buffers and play them together at the same time. DirectSound then combines all of them into the primary buffer and plays them out. Therefore, after extracting the sketch image from the photo taken from the webcam, we create a secondary buffer for each row in the image. We then write waves of different frequencies into each secondary buffer; starting from the bottom row to the top row are waves of low frequencies to waves of high frequencies. When the image is being scanned from left to right per column at a time, the corresponding wave is sounded out if the scanning column intersects with the sketch at the corresponding row.

In DirectSound, there are two ways to write wave data into the sound buffer: static and streaming [6]. Static method writes whole data into buffer at once. Streaming method writes data gradually while buffer is playing. For simplicity, we use the static method in our work, but this results in discontinue sound and costs a lot of memory to retain wave data.

2.5 Sounding with Stereo

To improve the quality of sound and make the visual disabilities aware of sound more easily, we have implemented sound to have stereo. More specifically, we implemented sound to have 44.1kHz sample rate, 16-bit bit-depth and 2 channels. We also set the position of sound source according to its position in the image. DirectSound supports 3-D sound implementation. In DirectSound, 3-D space is represented by Cartesian coordinates which are values on three axes: the *X*-axis, the *Y*-axis and the *Z*-axis as shown in Figure 19. Values along with the *X*-axis increase from left to right, along with the *Y*-axis from bottom to top, and along with the *Z*-axis from near to far [5]. The default position of the listener is on the right of the coordinate (0, 0, 0) facing forward to the positive *Z*-axis as Figure 19 illustrated.

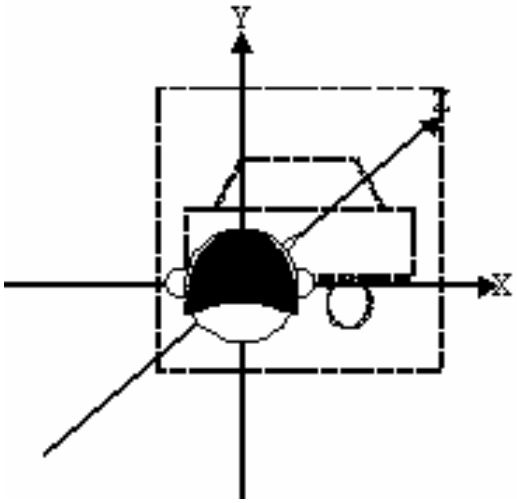

Figure 19. The coordinate system in DirectSound.

By using the function *SetPosition* of

IDirect3DBuffer interface in DirectSound, we set the sound position according to its vector in 3-D space. We make *z* component to have a fixed positive value and change *x*, *y* components according to its corresponding dot pixel in the image. Therefore, when the scanning column is on the left side, we could hear louder sound coming out from the left speaker and vice versa. Moreover, we could feel that the sound is slightly higher if the sketch is on the upper region.

3. Experimental Results

The implemented system includes a webcam and a notebook with Pentium M CPU running MS Windows XP SP2. The system we made is shown in Figure 20.

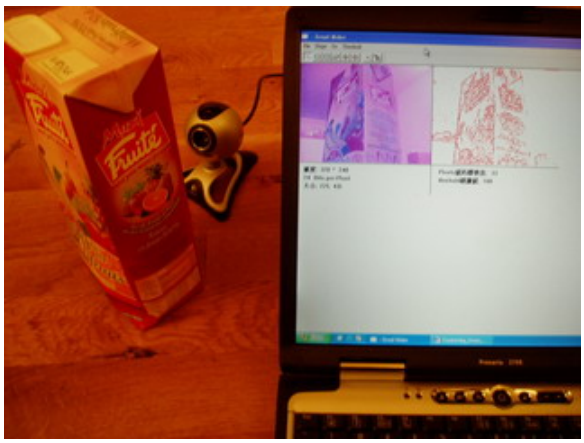

Figure 20. The implemented system.

We use a webcam to take some pictures and send them to the notebook via a USB port. Then we can hear the sounds corresponding to these pictures by our simulation. We have taken eight pictures for testing our work. The tested pictures are Figure 22, Figure 23**,** Figure 25**,** Figure 27**,** Figure 29**,** Figure 31**,** Figure 33, and Figure 35 and their sketch images are Figure 22, Figure 24, Figure 26, Figure 28, Figure 30, Figure 32, Figure 34, and Figure 36, respectively. We describe the sound of each picture following the sketches of the picture. The experimental results are described as follows.

Figure 21. Tested picture 1.

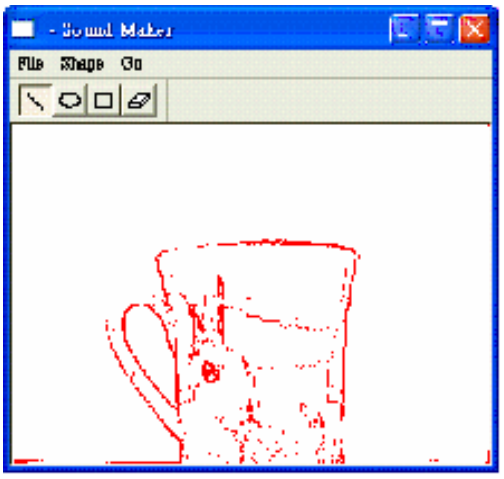

Figure 22. Image sketch of Figure 22.

Result of Figure 21:

We first hear a low frequency sound for a short time, then the frequency suddenly jump to high and have other lower frequencies noise, but we could hear the higher frequency clearly. We could recognize that there is an oblong shape object in the middle.

Figure 23. Tested picture 2.

Figure 24. Image sketch of Figure 23.

Result of Figure 23:

We hear two consecutive sounds. The former one's frequency is increased from low to high and decreased to low again. The later one is like the first one but its frequency's gap is smaller and the frequency is changed more slowly than the first sound.

Figure 25. Tested picture 3.

Figure 26. Image sketch of Figure 30.

Result of Figure 30:

We first hear that one frequency is increased from low to high slowly and then some high frequencies is decreased from high to low with a short time break. After that some frequencies is changed from low to high with some frequencies as noise for a long time.

Figure 27. Tested picture 4.

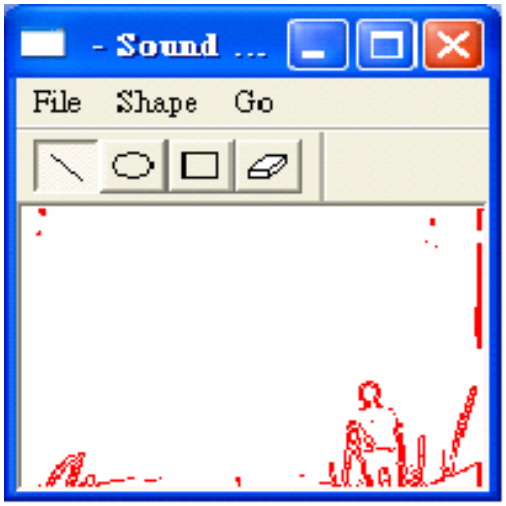

Figure 28. Image sketch of Figure 34.

Result of Figure 34:

We hear some low and short frequencies in the beginning, and then we hear some frequencies changed from low to high and high to low rapidly. Although we could not judge there is a person standing on the right of the picture, we could make sure there is something in the right of the picture.

Figure 29. Tested picture 5.

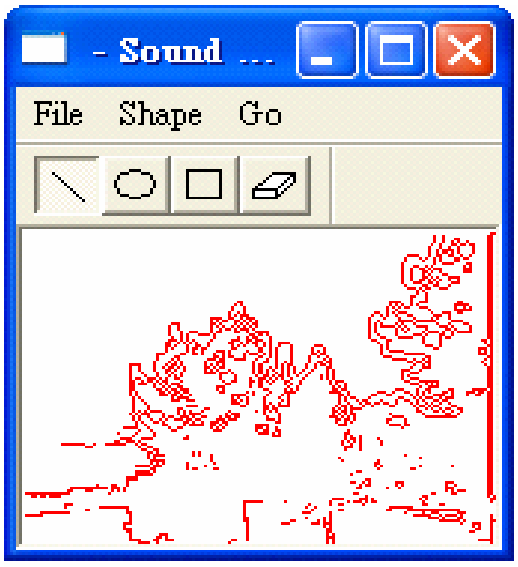

Figure 30. Image sketch of Figure 29.

Result of Figure 29:

We hear some frequencies with lot of noise from low to high, and the frequencies changed irregularly. It is too complex and difficult to judge what it is.

Figure 31. Tested picture 6.

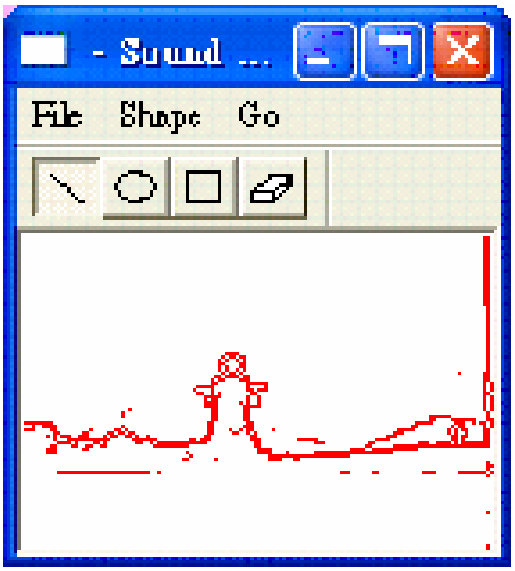

Figure 32. Image sketch of Figure 31.

Figure 33. Tested picture 7.

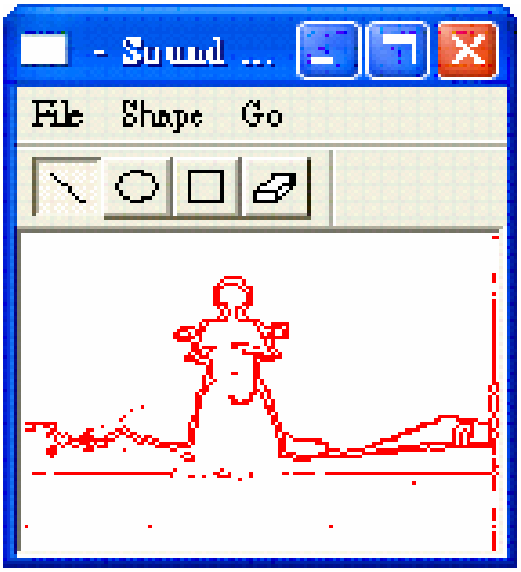

Figure 34. Image sketch of Figure 33.

Figure 35. Tested picture 8.

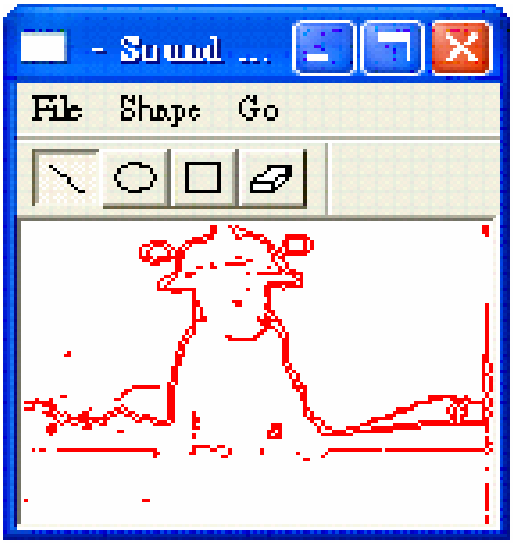

Figure 36. Image sketch of Figure 35.

Results of Figure 31, Figure 33, and Figure 35:

We continually took three pictures as shown in Figure 31, Figure 33, and Figure 35, and we could hear the frequencies changed similarly from the three pictures by our simulation. The frequencies produced by the three pictures are all from low to high and then from high to low clearly, but the average frequency of Figure 32 is the lowest one; the average frequency of Figure 36 is the highest one. According to the sounds produced by our system in accordance with the sketches of these three pictures, we could judge that there is an object closing to us.

4. Conclusion and Future Work

According to the result of our implementation, we could judge the simple sketch graph and we could sense one object getting closer to us by taking some continuing pictures, but it is difficult to judge the complex sketch graph. Although our algorithm for extracting sketch simplifies the complex sketch and makes our judgment less noise, one still could not judge objects in very complex pictures.

For the future work, we list four tasks to improve our current system. First, we could add an ultrasonic device to detect the distance of the center point of the image. Then the volume could be adjust according to the distance. For example, if the distance of the object is short, we make sound louder to warn the user; then the user would be watchful. Second, we could improve our sketch extracting program to extract the biggest or only the nearest object. Therefore, the user can only concentrate on the most dangerous object. Thirdly, the quality of sound is still not good. We could use streaming method to write wave data in the future to make the sound better and make the usage of memory smaller. Fourthly, we could study some researches regarding the hearing of human and improve the proposed image extracting algorithm.

References

[1] 井上誠喜，八木伸行，林正樹，中須英輔，三谷公二，奥井誠人，利用微分擷取影像的輪廓，C 語言數位影像處理，全華圖書, 2005.

[2] M. Luse, *Bitmapped Graphics Programming in C++*, CA, Addison Wesley, 1993.

[3] M. Luse, "The BMP File Format," *Dr. Dobb's Journal*, Vol. 9, No. 10, pp. 18-22, 1994.

[4] A. Mbogho, "Video for Windows Single-Frame Capture Class Without MFC," February 1, 2005, http://www.codeguru.com/Cpp/G-M/multimedia/video/article.php/c4723.

[5] Microsoft Corporation, DirectSound, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/directx/htm/directsound.asp.

[6] M. J. Norton, “Creating DirectSound Objects, Creating DirectSound Buffers and Writing DirectSound Buffers,” in *Spells of Fury: Building Windows 95 Games Using DirectX 2*, Waite Group, 1996.

[7] X.-B. Peng, L.-P. Chen, F.-L. Zhou, and J. Zhou, “Singularity Analysis of Geometric Constraint Systems”, *Journal of Computer Science and Technology*, Vol. 17, No. 3, pp. 314-323, 2002.

[8] H.-C. Yeh and T.-C. Lin, “An Embedded System for The visual disabilities Seeing Scenes by Ears,” Project Report, Department of Computer Science and Information Engineering, National Chi Nan University, 2004.

[9] D. Vernon, *Machine Vision*, NJ, Prentice-Hall, 1991.

[10] L. Zimet, M. Shahram, and P. Milanfar, “An Adaptive Framework for Image and Video Sensing, Department of Electrical Engineering, University of California, available at http://www.cse.ucsc.edu/~milanfar/EI5678-15Final.pdf.