Chapter 5
Real-time Cartoon Face Animation

5.1 Introduction

In a system for real-time generation of talking cartoon faces, four major functions are necessary. First, it is basic to capture and process image sequences in real time. Second, the work for recording and playing speeches in real time is similarly important. Third, the performance of real-time facial feature tracking is easily affected by the uncontrolled environment. So a friendly interface between the user and the system for real-time generation of talking cartoon faces must be created. Finally, a complete integration of cartoon videos and audios is needed.

According to the four functions mentioned above, a system for real-time generation of talking cartoon faces is proposed in this study and described in this chapter. The system includes four major parts: an environment regulator, a facial feature tracker, a sound recorder, and an animation generator. The environment regulator provides a friendly interface to let a user easily regulate some parameters to reduce the miss of real-time processes. The main parameters needed to regulate are the threshold values for segmentation of eye-pair regions and mouth regions and a division line. The facial feature tracker is used to track image feature points from the sequential facial images, which are captured from a camera in real time. The sound recorder is used to record a user's speeches in real time from a microphone. The animation generator is used to generate the talking cartoon faces by rendering the cartoon face images and synchronizing the video and the audio in real time. A

configuration of the system for real-time generation of talking cartoon faces is shown in Figure 5.1.

The detailed process for the friendly interface about the learning of environments is described in Section 5.2. A brief real-time process for generation of cartoon faces is described in Section 5.3. Some processes about the speeches captured from a microphone are described in Section 5.4. A synchronization of cartoon videos and speeches is described in Section 5.5.

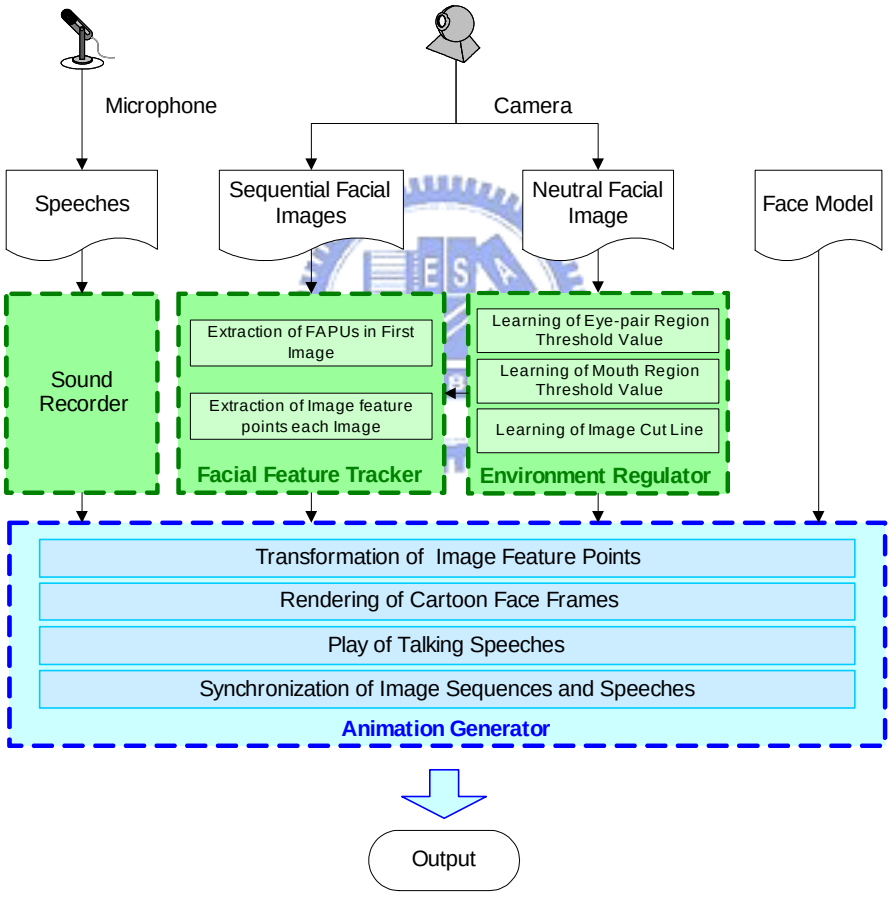


Figure 5.1 A configuration of the system for real-time generation of talking cartoon faces.

5.2 Learning of Environment

Before starting the real-time facial feature tracking from sequential facial images, some parameters must be confirmed first. In the method for off-line facial feature tracking, the neutral facial image is the first frame in the image sequence. The FAPUs extracted from the neutral facial image are used for tracking in the image sequences. But in the method for real-time facial feature tracking, the neutral facial image of the image sequence is unknown until tracking is started. Therefore, a learning of the environment is necessary to obtain information about users in the environment. Then using the information, the FAPUs can be extracted from the neutral facial image in the image sequence when starting tracking to reduce the errors.

There are three parameters that must be confirmed before tracking. In Section 5.2.1, a process for learning the threshold value for segmentation of eye-pair regions is described. In Section 5.2.2, a process for learning a division line for mouth tracking is described. In Section 5.2.3, a process for learning the threshold value for segmentation of mouth regions is described.

5.2.1 Learning of Eye-pair Region Threshold Value

An optimal threshold value for segmentation of eye-pair regions is speculated in this section. The detailed process is described in the following algorithm.

Algorithm 5.1. *Learning process of eye-pair region threshold value.*

Input: several neutral facial images N_1, N_2, …, N_s.

Output: a threshold value t_1.

Steps:

1. For each neutral facial image N_i, compute the threshold value e_i by Algorithm 3.1, where i = 1, 2, …, s.
2. Choose n threshold values that have good results by visual inspection, where $n \leq s$.
3. Get the value t_1 by averaging the n threshold values obtained in Step 2.

5.2.2 Learning of Division Line

The division line mentioned here is the line Div_{NL} described in Algorithm 3.2. This horizontal division line is used to cut a binary facial image. The binary facial image can be separated into two binary images B_1 and B_2. The binary image B_1 is obtained by thresholding the pixels of the image above the division line with the eye-pair region threshold value. The binary image B_2 is obtained by thresholding the pixels of the image under the division line with the mouth region threshold value. An illustration of B_1 and B_2 is shown in Figure 5.2.

Figure 5.2 An illustration of B_1 and B_2.

Using the division line can reduce the computation time of image processing and the errors of segmentation of mouth regions. In this Section, a coefficient value u used

to speculate the division line in image sequences is computed. The detailed process is described in the following algorithm:

Algorithm 5.2. *Learning process of image division line.*

Input: several neutral facial images N_1, N_2, …, N_s.

Output: a coefficient value u.

Steps:

1. For each neutral facial image N_i, perform Steps 2 through 4.
2. Compute the initial division line Div_{NL} in the following way:

   $$y_{Div} = y_{mideye} + d,$$

   where y_{Div} is the y position of the division line Div_{NL}, y_{mideye} denotes the y-position of the middle of eye pairs and d denotes the distance between the eye pairs.
3. If the position of the initial division line in the neutral image is not good as judged by visual inspection, regulate the division line for a better result.
4. Compute a coefficient value u_i in the following way:

   $$u_i = \frac{y_{Div}' - y_{mideye}}{d},$$

   where the y_{Div}' is the y position of the regulated division line Div'_{NL}.
5. Get the final coefficient value u by averaging the values u_1, u_2, …, u_s.

5.2.3 Learning of Mouth Region Threshold Value

According to the division line described above, an optimal threshold value for segmentation of mouth regions is speculated in the following algorithm:

Algorithm 5.3. *Learning of mouth region threshold value.*

Input: several neutral facial images N_1, N_2, …, N_s, and a coefficient value u.

Output: a threshold value t_2.

Steps:

1. For each neutral facial image N_i, perform Steps 2 and 3.
2. Compute the division line Div_{NL} in the following way:

   $y_{Div} = y_{mideye} + u \times d$,

   where y_{Div} is the y position of Div_{NL}, y_{mideye} denotes the y-position of the middle of eye pairs, and the d denotes the distance between the eye pairs.
3. Compute the threshold value m_i by Algorithm 3.2, where i = 1, 2, …, s.
4. Choose n threshold values that have good results by visual inspection, where $n \le s$.
5. Get the value t_2 by averaging the n threshold values obtained in Step 4.

5.3 Real-time Cartoon Face Generation Process

When starting to track facial features in real time, the first image captured from a camera is taken as the neutral facial image. If the FAPUs are extracted from the neutral facial image correctly, the facial feature tracker keeps capturing the image sequence from a camera and tracking the image feature points. The detailed process of extraction of facial features was described before in Chapter 3. Then, the animation generator transforms the image feature points into the face model control points and

renders the cartoon face images in real time. The detailed process of such a transformation was described in Chapter 2.

Sometimes an exception happens in the process of real-time facial feature tracking. Usually the facial feature tracker ignores the exception by using the facial feature points in the previous frame to recover the facial feature points in the frame that has an exception. If the exception is very grave, the facial feature tracker will stop the process and notice users that an exception happened in the process. The entire process for real-time generation of cartoon faces is described as follows and a flowchart of the process is shown in Figure 5.3.

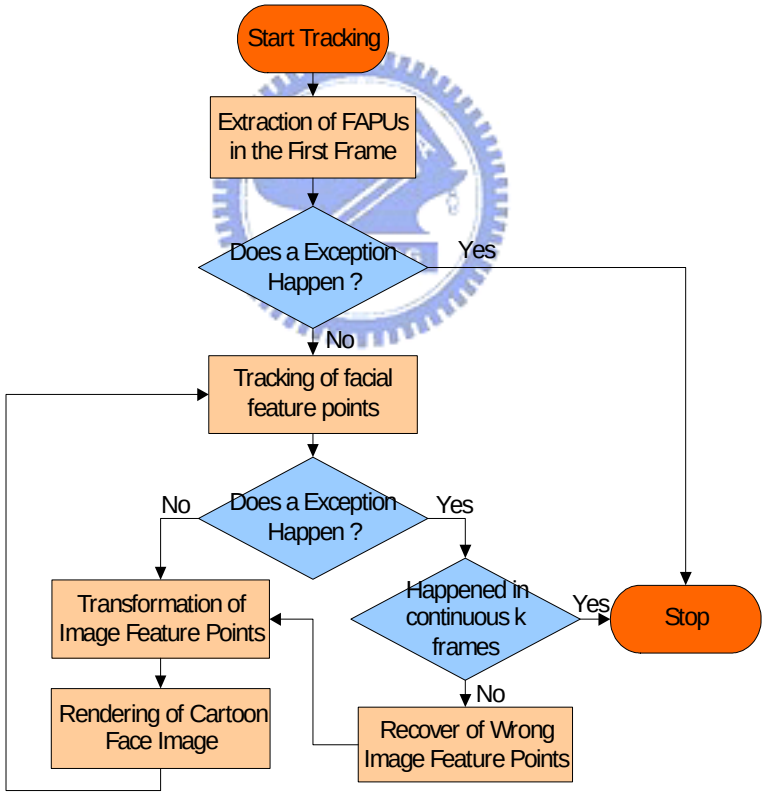


Figure 5.3 A flowchart of the process for real-time generation of cartoon faces.

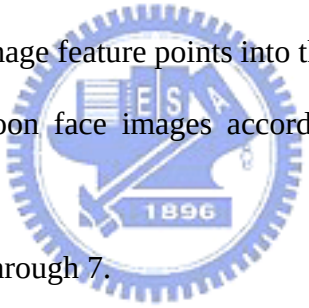
Algorithm 5.4. *Process for real-time generation of cartoon faces.*

Input: a sequence of facial images $I = \{I_1, I_2, \ldots, I_n\}$, and a value k.

Output: a sequence of cartoon images $C = \{C_1, C_2, \ldots, C_n\}$.

Steps:

1. Extract the FAPUs from the first frame of image I_1.
2. Stop the process if the extraction of the FAPUs has an error.
3. Track the facial feature points from the image sequence.
4. Stop the process if wrong eye-pair regions or wrong mouth regions are extracted and the number of times of the wrong extractions is greater than k.
5. Recover the wrong image feature points by the image feature points in the previous frame if the number of times of the wrong extraction is smaller than k.
6. Transform the image feature points into the face model control points.
7. Render the cartoon face images according to the face model control points.
8. Repeat Steps 3 through 7.



5.4 Speech Recording and Play

In this Section, some processes about processing speeches are described. An overview of the WAV audio file format is described in Section 5.4.1. Processes about how to record and play speeches are described in Section 5.4.2.

5.4.1 Overview of WAVE Audio File Format

The WAVE audio file format is developed by Microsoft. It is based on the RIFF document format and is identified by a file name extension of WAV. Because the

Microsoft Company uses WAVE files in its operating system Windows, the WAVE files get popular and become the standard PC audio format. And the WAVE files support many audio compression algorithms.

In this section, an uncompressed PCM wave format is described. The WAVE PCM file starts with an “RIFF” chunk identified by “WAVE.” And the “fmt” chunk and the “data” chunk are the subchunks of the “RIFF” chunk. The “fmt” chunk contains the wave audio format in the file. The “data” chunk contains the uncompressed raw audio data. An illustration of WAVE PCM files is shown in Figure 5.4.

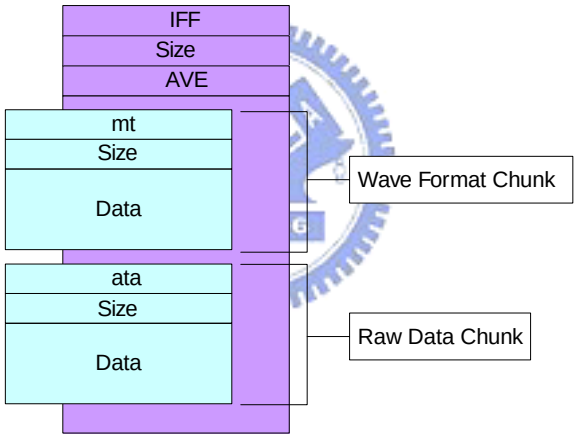


Figure 5.4 An illustration of WAVE PCM files

The WAVE audio format includes six elements: wFormatTag, nChannels, nSamplesPerSec, nAvgBytesPerSec, nBlockAlign, and wBitsPerSample. The wFormatTag denotes the waveform-audio format type. The nChannels denotes the number of channels in the waveform-audio data. The value is 1 for mono and 2 for stereo. The nSamplesPerSec denotes the sample rate, the number of samples per second. In WAVE PCM files, the value can be 8.0 kHz, 11.025 kHz, 22.05 kHz, and

44.1 kHz. The nAvgBytesPerSec denotes the required average data-transfer rate. The nBlockAlign denotes the block alignment, the minimal data size in bytes. The wBitsPerSample denotes the bits per samples. In WAVE PCM files, this value should be equal to 8 or 16.

5.4.2 Process of Recording and Play

In order to capture a user's speeches, a Waveform Audio SDK provided by Microsoft is used in this study. The speeches are recorded with a simple waveform-audio format specification: mono, 8.0 kHz, and 8 bits per sample. Although the quality of digital sounds is not the best, it cannot be discriminated by the human's ear.

An audio buffer is used when recording the speeches. If the audio buffer is full with the speeches captured by a microphone, the output audio device plays the speeches stored in the buffer immediately and releases the data in the buffer. By repeating the work for recording and playing the speeches in the audio buffer, a system for real-time talking speeches can be established.

If a user wants to stop the system for real-time talking speeches, the system will keep recording speeches until one audio buffer is full, and then the recording is stopped. And the output audio device will play the speeches in the full audio buffer and stop. A flowchart of the system for real-time talking speeches is shown in Figure 5.5.

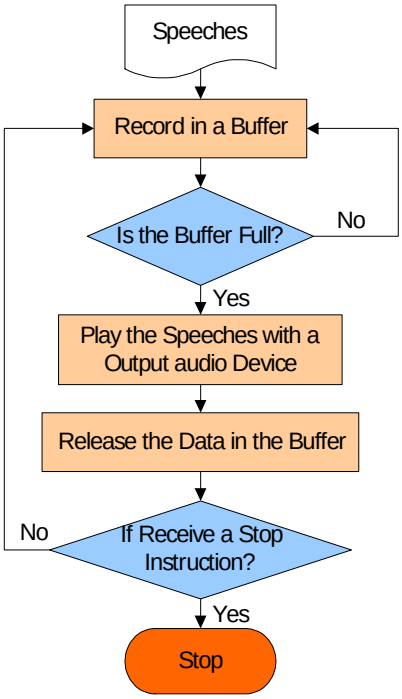
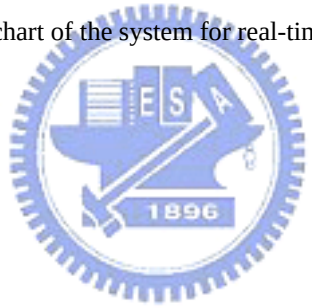


Figure 5.5 A flowchart of the system for real-time talking speeches.



5.5 Synchronization of Cartoon Videos and Speeches

In most researches of talking virtual faces, the syllables of speeches are analyzed for synchronizing moving lips and speeches. However, the methods for analysis of syllables are different for different languages. The applications for the research of talking virtual faces are limited by the analysis of syllables in different languages.

In this study, the syllables of speeches captured from a microphone are not analyzed. The lip movements of talking cartoon faces are synthesized according to the sequential facial images. It is possible then to generate talking cartoon faces in real

time with any speeches.

On the other hand, automatic synchronization of cartoon videos and speeches is a difficult problem because no information about speeches can be used. In order to solve the problem, a friendly interface is proposed to help users to synchronize the cartoon videos and speeches. The basic idea is to fix the timing of speeches and regulate the delay of images to synchronize the cartoon videos and speeches. In this interface, a delay value is used to control the delay of images. The delay value denotes the delay of frames in the system for real-time generation of talking cartoon faces. A user can change the delay value to regulate the delay of images to synchronize the cartoon videos and speeches in real time. The detailed process for synchronization of videos and speeches is described as follows. And a flowchart of the process is shown in Figure 5.6.

Algorithm 5.5. *Real-time synchronization of videos and speeches.*

Input: a delay value *d*.

Output: a synchronized talking cartoon face.

Steps:

1. Store a set of face model control points in the present frame into a queue *Q*.
2. If the size of *Q* equals *d*, pop up a set of face model control points from *Q* and render the talking cartoon face image according to the face model control points.
3. Clear the data in *Q* if *d* is changed in the process of real time talking cartoon face generation.
4. Repeat Steps 1 through 3.

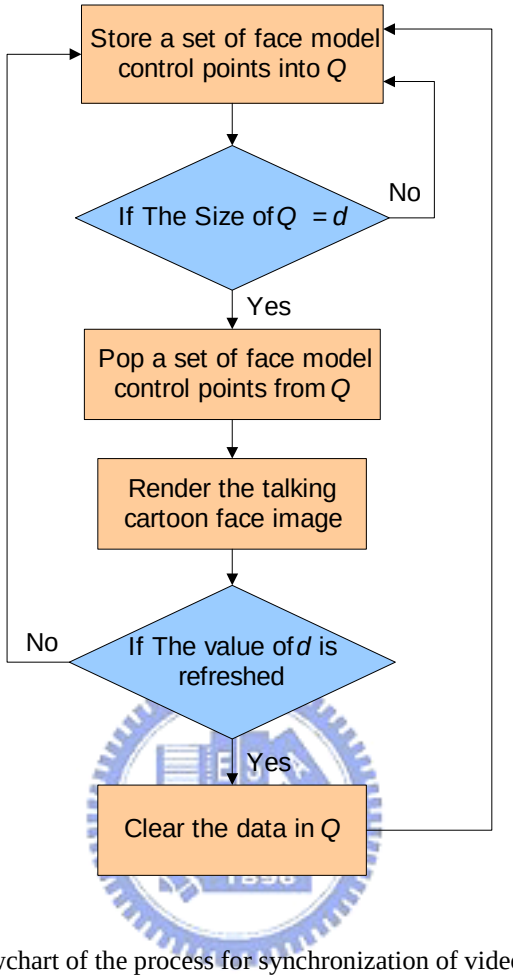


Figure 5.6 A flowchart of the process for synchronization of videos and audios.

5.6 Experimental Results

In this section, some experimental results of real-time cartoon face animation are shown. Figure 5.7 shows the program interface of real-time talking cartoon face generation used in this study. The red circle indicates the interface for synchronization of images and speeches. A process of environment learning is conducted before starting the real-time facial feature tracking, as shown in Figure 5.8. Then a face model is chosen as the avatar, as shown in Figure 5.9.

Figure 5.7 The program interface of real-time talking cartoon face generation.

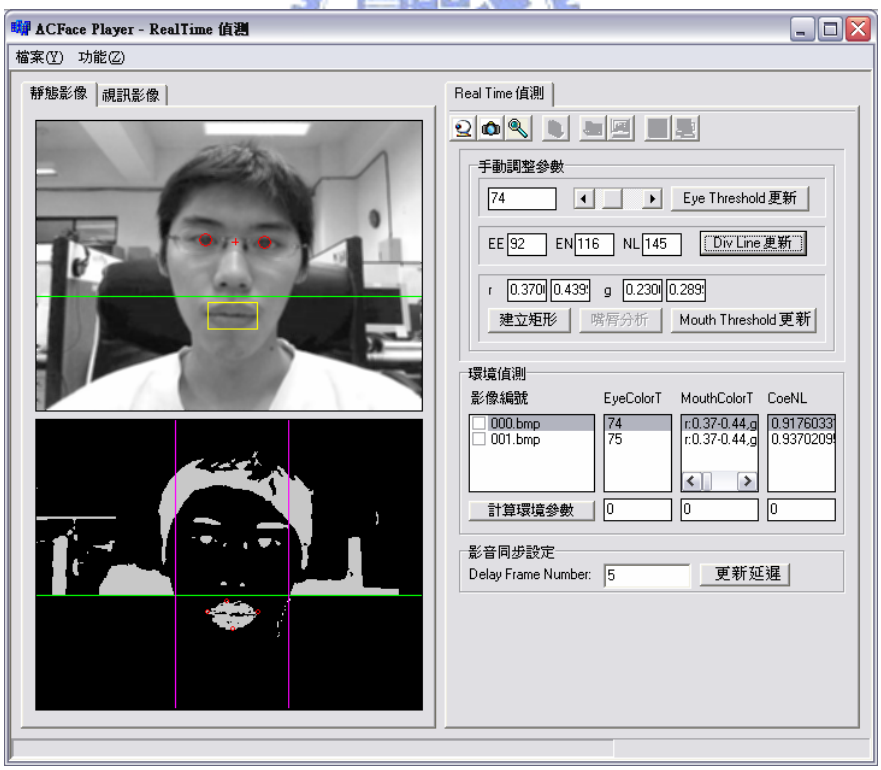


Figure 5.8 The process of learning of environments before starting real-time facial feature tracking.

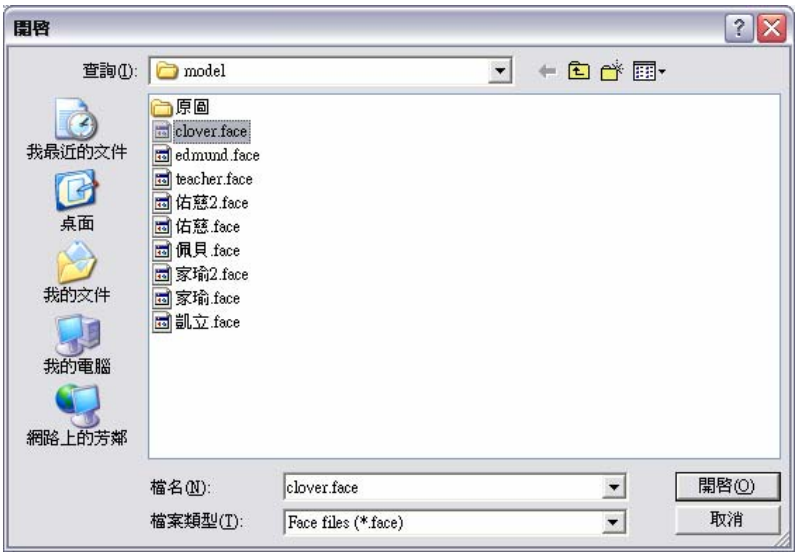


Figure 5.9 The interface for the process of choosing a face model.

Two resulting talking cartoon faces are shown. An experimental result of real-time frontal talking cartoon faces is shown in Figure 5.10. An experimental result of real-time oblique talking cartoon faces is shown in Figure 5.11.

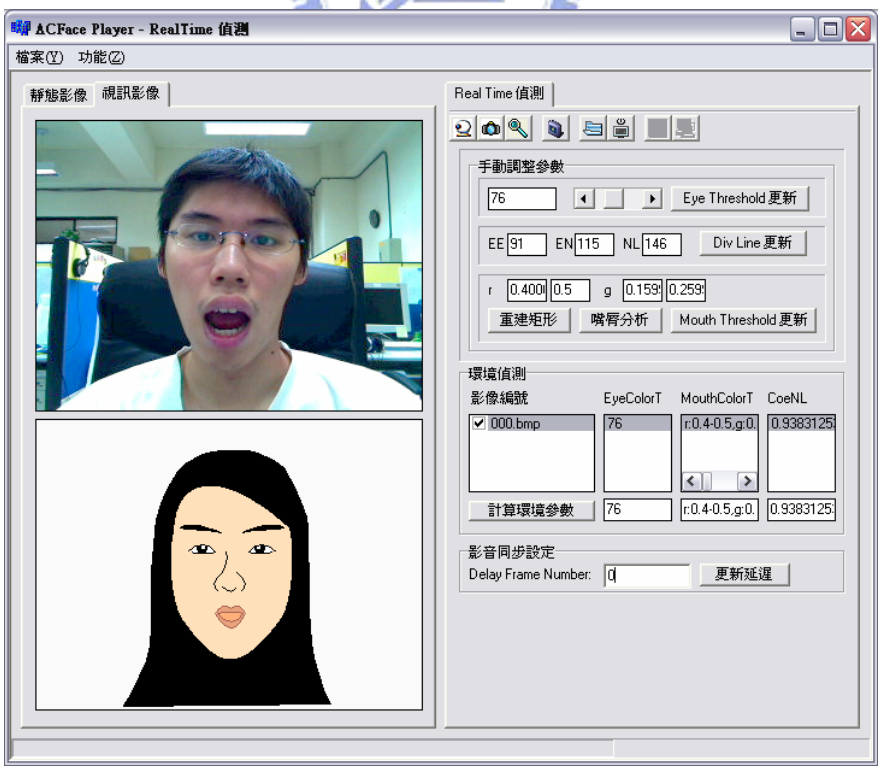


Figure 5.10 An experimental result of real-time frontal talking cartoon faces.

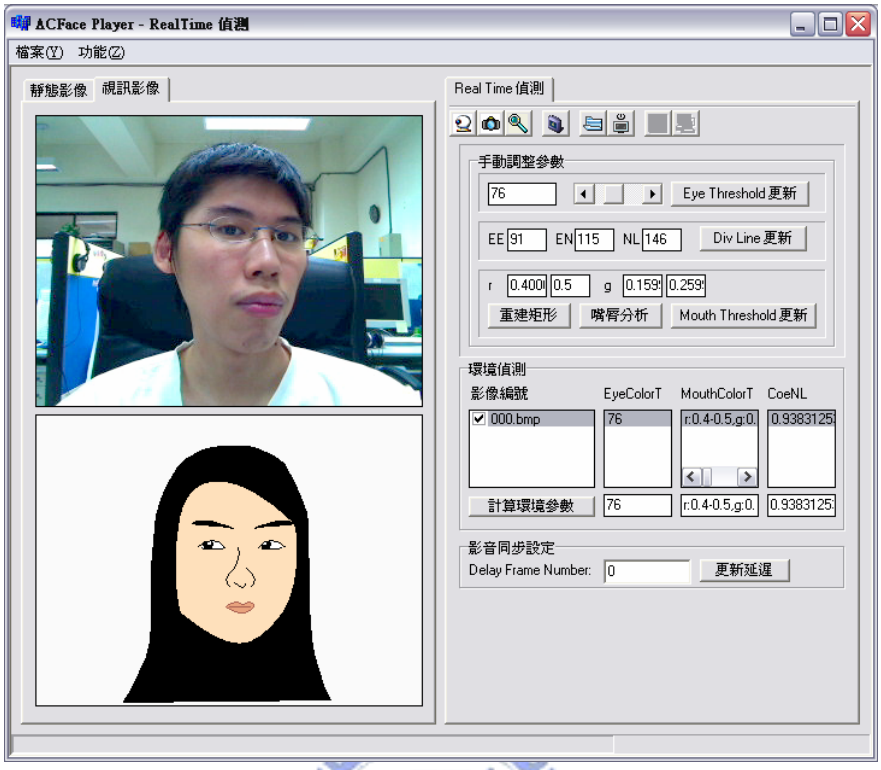


Figure 5.11 An experimental result of real-time oblique talking cartoon faces.