Hook and Screen Division Based Screen Compression for the Real-time Multimedia Transformation of E-learning System

Shengjun Li, Ruimin shen
Computer Science Department, Shanghai Jiaotong University
Shengjunli,rmshen@sjtu.edu.cn

Abstract

Real-time multimedia transfer is critical for the distance E-learning application. Because the students are geographically separated, and use different kinds of PCs and variant network bandwidth, efficient and specific compression technology should exist to transfer the teaching white board (or screen image) from the main classroom to the student's PC or the distance classroom, along the limited network bandwidth. In this paper, we assume that the teacher use MS PowerPoint or MS Word in the main classroom for their teaching, and at the same time, we record the screen with our specific compression technology and transfer the screen video to the distance classroom or students' PCs, the compression uses hook technology to trace the changed area of the screen and divided the screen into 16 small rectangles, and it selectively compressed the changed area of the screen with lossless and lossy compression methods, and use a long motion compensation history buffer to further reduce the time redundancy. The compression method can get a pretty high compression ratio and very good quality of the text and simple icons that is the key issue to guarantee the quality of the teaching video; besides, the CPU utilization is also pretty low compared with other regularly video compression technology. And our technology can use unicast and multicast to transfer the compressed data and save the screen video to the local machine.

Keywords: Hook and Screen Division, Screen Compression, real-time multimedia transformation, lossless compression, lossy compression, E-Learning

1 Introduction

Currently, real-time E-Learning is a hot topic both in the research committee and industry. Real-time E-Learning should transfer the teaching video in the main classroom to the students' PCs or distance classrooms in real-time.

To transfer the teaching process vividly, the teaching whiteboard (such as PowerPoint played in the teacher's PC) should be recorded and transfer with good quality and in real-time.

At the same time, the video should be compressed with high efficiency for the following reasons. Firstly, the students are geographically distributed, so the teaching video should be transferred across the internet, for the bandwidth dynamic of the network, and the limited bandwidth of the student's PC (some students have ADSL access, but some others used ISDN or even modem), the video should be compressed efficiently so that it can be transferred relatively smoothly across the network. Secondly the compression mythology should not be very complex for we use the PC on which the teachers are display PPT for their teaching, the record program should be run on the background and it should not occupy a lot of CPU utilization so as not to influence the teaching process of the main class.

We found out that almost all the teachers use the MS PowerPoint or Word for their class, the playing of PPT is pretty different from the movies which is compressed with Mpeg-2 or Mpeg-4, for the teaching PPT is played more slowly and regularly, and the objectives in the PPT page are

mostly simple text and GUI icons, though sometime are complex pictures. And some teachers like to jump from one page of PPT to another. So In order to compress the teaching screen efficiently, special compression technique is needed.

For the forenamed reasons, we develop the hook and screen division based screen compression technology. The basic idea is that we divided the screen in to 16 small rectangles (four times four), and use hook technology to capture the change area of the screen. For simplify, we use the out-wrap rectangle of the changing area instead of the precise region. And we map the change rectangle to the 16 screen rectangles, if one rectangle is affected, it should intersect with the changed rectangle and get an intersected rectangle. Iterate the process, we can get at most 16 small changed rectangles if all the divided screen rectangles are affected. And no small changed rectangle can be got if no change on the screen at all. We record all the changed rectangles' size and location, and only compress the data of the changed rectangles and send them out as the compressed video.

To compress the data efficiently, we use both loss compression algorithm such as JPEG for complex pictures in PPT and lossless compression algorithm such as LZW for text and simple icons for they are the most important things in the teaching PPT. To further compress the video, we also use motion compensation technology with a frame buffer of longer history (store eight frames).

And our technology can also save the video data to the local computer disk so that the file can be used for non-real-time E-Learning.

The rest of this paper is organized as follows: Section 2 reviews the related works of the screen compression technology. Section 3 presents the hook and screen division based screen compression technology. Section 4 is the comparison of this technology with other methodologies. Section 5 discusses the application of the technology in the real E-Learning systems. Section 6 concludes the paper and points out the future work.

2 Related works

Screen compression methods have been researched for a few years; in paper [1] the author introduced LAN-based e-learning system and the key techniques in the multimedia compression in the e-learning system. Paper [2] introduced one IP based video stream system, and the authors mainly introduced the system model and protocol model. Their protocol is RTP/RTCP protocol and RTSP and SDP, Which are all based on IP-layer. Paper [3] discussed the problem of the network video transfer and one solution based on video layer division technology. And they specifically proposed how to use the video layer to reduce the data loss of motion vector's influence on the video quality.

Furthermore, the screen compression has been used for practice for several years. The main screen compression and share technologies are divided into the four following g categories.

(1) Microsoft's NetMeeting SDK

Microsoft's NetMeeting SDK [4] is a mature technology for the screen sharing; it uses unicast to delivery the desktop to the remote customer for sharing. But it has the following limits: The screen color is too limited (it can support only 256 kinds of colors), and it cannot support multicast so the number of connections is limited by the bandwidth, and the screen video cannot be saved etc.

(2) Videoconference technology.

With videoconference technology, the screen data is collected from a video camera head. The

method is straightforward, but the quality depends on the videoconference system itself rather than the screen's PC. To get high video quality, the video collection card should have very high resolution and the screen video have the un-avoided flash phenomena.

(3) Screen capture and lossless compression

Software tool PC Anywhere [5] is a remote monitor tool; its function is like a remote terminal. The working principle of PC Anywhere is that the screen is copied and compressed as a picture and sent to the clients. PC Anywhere uses the lossless compression method for the screen data, lossless compression is good for the text or simple icons, but it is not powerful enough for the color-abundant pictures.

(4) Screen capture and lossy compression

This kind of technology is similar to the technology of screen capture and lossless Compression, except that it uses lossy compression instead of lossless compression, the classical lossy compression is JPEG [6], but JPEG is designed for the compression of colorful pictures, it is not very suit for the simple pictures such GUI and text which is the most important things in the teaching process.

Our mythology tries to combine all the advantage of the existing methodologies and overcome their deficiency.

3 Hook and division based screen compression

In this section, we introduce our technology in detail. Note that we use MS Visual C++ to develop the screen compression software on the platform of Windows 2000.

3.1 Hook and Screen Capture

Hook technology is used to capture the changed area of the screen. In our technology, system hook and API hook is installed simultaneously.

The System hook is a technology that can capture the MS window's system messages. The system hook will deal with the following messages such as window refresh, window location change, window size change, menu display, menu exit etc, every message is related to a corresponding window change and the changed area is considered as the out-wrap rectangle for simplicity, collect all the information of the rectangles and combine all of them to one region, when the window-move message comes and is captured, the information of the new window can be got.

The API hook is a trap technology that can capture the system API calls. In the API hook, we process the following API calls: TextOut, Bitblt, StretchBlt, SetDibBits, every API is related to a specific picture display area, the area is got by analyzing the called parameters. By capturing the API calls, API hook can get the management of the APIs, and work out the calls' influence to the screen picture.

With the two hooks, we only analyze the changed area of the screen and not do any other process. In the compression step, we only compress the changed areas (rectangles), if no changes happen, no extra steps are needed to compress the data. By this method, we can reduce the amount of data for video processing, and reduce the CPU utilization and the bandwidth demand for the video data transfer.

The data of the changed screen image is captured from the video card. Firstly, we use GDI function CreateDIBSection to create one bitmap with the same size as the screen, and the select the bitmap to a screen compatible device HDC. GDI function BitBlt is used to copy screen image

from the video card to the system main memory. By setting the right parameters, using BitBlt function can read any changed rectangle's data. By dividing the changed area to a few rectangles, the changed area can be read by using BitBlt multiple times (each time for one rectangle).

3.2 Compression process

In our system, the screen is divided into 16 rectangles (four rows and four columns). And we map the changed area to the 16 rectangles. Then we can get the small changed area in every rectangle. For simplicity, we use the out-wrap rectangle of the small changed area in every divided rectangle instead of the precise region. So the changed area can be divided into at most 16 small changed rectangles. And we only process the changed rectangles. The screen area move is shown in Figure-1. And the map of changed area to the divided rectangles is shown in Figure-2.

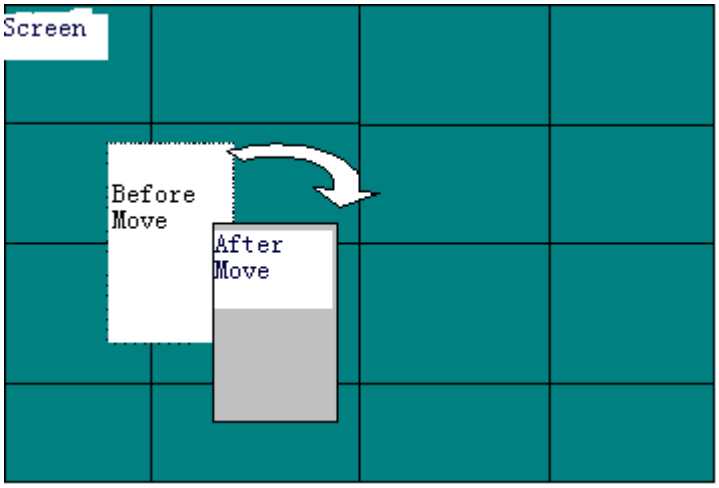

Figure-1 The screen change area move

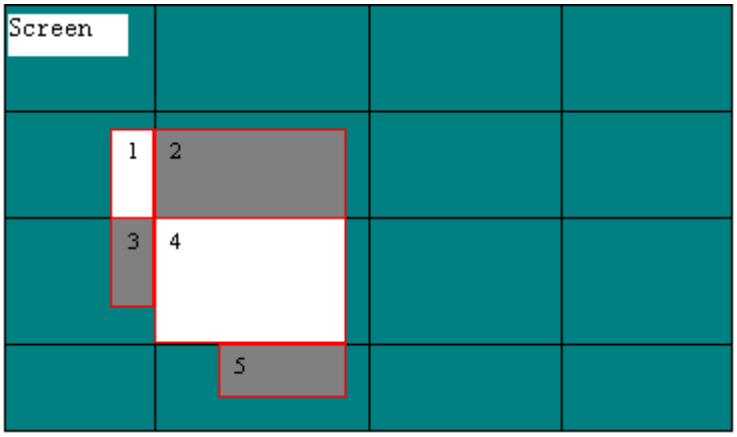

Figure-2 map the changed area to the divided rectangles. The change area is mapped to five divided rectangles, and five small changed rectangles are got, numbered from 1 to 5.

If one divided rectangle has one small changed rectangle, we will encode the small changed rectangles, also record the location and the size of the small changed rectangle.

To encode the data of one small changed rectangle, we use too kinds of compression techniques, lossy compression method and lossless compression method. Lossy compression method such as Jpeg is used for the complex pictures, and the lossless compression method LZW [7] is used for the simple text and icons. The selection of the compression method depends on a color change threshold, we use a simple and straight forward method to distinguish text and icon from complex

pictures, if the color change in a small macro block is beyond the threshold (in our system we use 24 as the threshold), the data is considered as the data of the pictures, otherwise it is considered as the data of simple text and icons. By using both the lossy and lossless compression methods, we can get very good quality of the video especially of the text and icons, at the same time we can get very good compression ratio.

Furthermore, we use a pretty simple motion compensation strategy to reduce the time redundancy between the frames (or corresponding rectangles), we use a history frame buffer for the previous frames, at most the buffer can store eight most recent previous frames, if the small change rectangle is the same as some rectangle of previous frames, we can only save the index of the rectangle and need not record the data of it, we can further compress the data in this way. After compression, we record the location and the size of the small changed rectangle and the length of compressed data. For the divided rectangle that has no small changed rectangles, we need not record its image data at all.

3.3 Video data transfer and local data record

To transfer the reordered data, we use unicast and multicast methods.

Under unicast, the screen-recording program listens to one port (such as port 900), and the remote client using TCP protocol to connect with the recording program. The recording program transfers the compressed data to the client after connection. The application of unicast is limited by the bandwidth and the performance of the recording machine, so that multicast should be used for the large-scale E-Learning systems.

Under multicast, the recording program only need to send the compressed data to a multicast server (such as 218.7.7.7), and the clients who join in the multicast group can get data from the server. Multicast can support much more clients, and it will not influence the workload of the recording computer. But the multicast has been limited by the network topology and the package loss problem, so we use both unicast and multicast strategies.

To save the data to the local computer, we use two kinds of file formats to save the video data *.AVI file and our self-defined file *.SR file. For AVI file, we use the API function AVIFileOpen to create AVI file. And use AVIFileCreateStream to create screen stream, and set up the compression flag (AVIStreamSetFormat) that is defined by ourselves.

The *.SR file uses the format which is similar to AVI file but it is much simpler, we throw away the unnecessary file trunk from the AVI files, so the *.SR file is much simpler and a little smaller compared with the corresponding AVI file.

3.4 Decompression process

The decompression process is a contrary process of compression.

Firstly, the remote client receives IP data package, every data package has the data of a frame of the screen video, one frame consists of a few small rectangles, and we decompress them according to predefined order. Firstly, we receive one video frame data package, and read the location and size of the rectangle in the screen, and then decompress the data, move the data to the corresponding area of the screen, then decompress the second rectangle data, till all of the rectangles (at most 16) are decompressed.

To display the screen image to the preferred window, we use GDI style to display the image; firstly we can the device context of the window (GetDC), and display the screen data to the device (SetDIBBitsToDevice). And then release the device (ReleaseDC).

4 Comparisons and Evaluation

To evaluate the performance of our algorithm, we compared it with other methods. Firstly, we compared it with the other screen compression algorithms. We compare our compression tool SjtuRecorder with other three tools CamTasia [8], Media Encoder [9], and TeachingRecorder.

The test machine is one PC with Pentium III CPU and 256 M memory. And we displayed one teaching PPT and record it with the different tools, respectively. We compare the CPU utilization, the colorDepth and the average bandwidth of the tools. The test result is shown as the following table—table-1.

Encode Tool	SjtuRecorder	CamTasia	MediaEncoder	TeachingRecorder
AudioEncoder	8kbpsDivx	8kbpsDivx	8kbpsWmv	8kbps
CPU utilization	8%	30%	80%	10%
ColorDepth(bits)	16	16	16	16
Average Screen Data (Byte per Second)	1.6KBps	2KBps	3.1KBps	10.3KBps

Table-1

From the table, we can find out that our tool SjtuRecorder can get the lowest CPU utilization and the lowest average screen data (Byte per Second).

The reason that CamTasia's performance is poorer than our SjtuRecorder is because that CamTasia use Tscc that is one lossless algorithm as the compression algorithm, and it can only reduce the redundancy between the former and the current frames. It can lead to huge data increase if it meets complex pictures and in the PPT page jump. But in our SjtuRecorder, we use both lossy and lossless compression methods so the data will not increase too bursty even meet complex pictures, and at the same time, our history buffer has stored the data of the most previous eight frames rather than only one, so it can reduce more time redundancy.

The following curve is the comparison between SjtuRecorder and CamTasia.

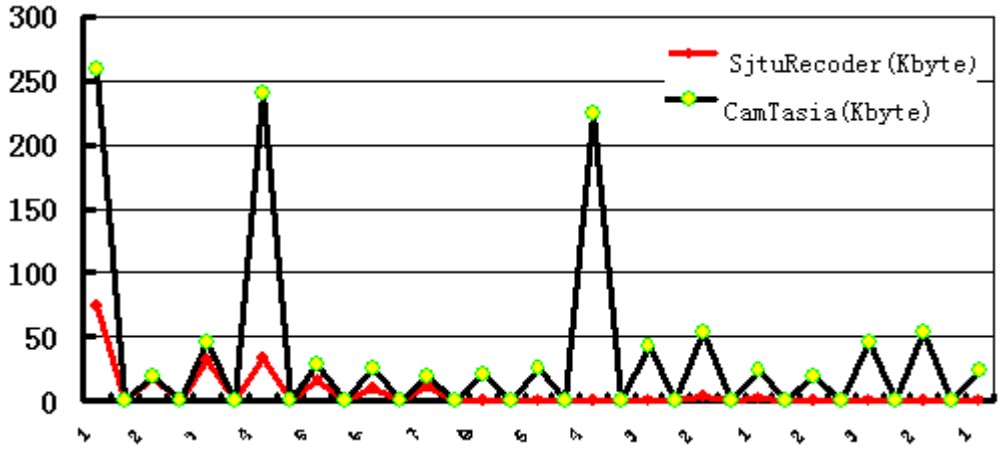

Figure-3 the X-axis is the sequential numbers of PPT pages, and the Y-axis is the size of the screen data after the compression

This figure shows one interesting thing, if we play the PPT from page 1 to 7, and then play from 6-1,we find out that in the second half of the playing time, the SjtuRecorder's data created is pretty little, this is because the long motion compensation buffer (recall we store most recent eight frames in the buffer), but the CamTasia's data needed is still high which is because it only reduces

the time redundancy between the current and former frames, totally only two frames. Also, we find when the CamTasia meet complex pictures, the data will increase suddenly but the increase in the data of SjtuRecorder is not that obvious, this is because Camtasia uses lossless compression for all the screen but SjtuRecorder uses lossy compression for the complex pictures.

To explain why we use 16 as the screen division granularity, we want to say that it is got from the tradeoff between process complexity and the screen division precision. If the granularity is too fine, the process will be too complex and the CPU utilization will be higher, on the contrary, if the granularity is too gross, the data will be bigger and we will lose the usage of screen division.

The following table is one comparison between three screen division granularities.

Division Granularity	2 *2	4*4	8*8
CPU Utilization	6%	8%	11%
Average Screen Data (Byte per second)	2.1KBps	1.6KBps	1.4KBps

Table-2

We found out that finer granularity than 4*4 can make the average screen data a little smaller, but the CPU utilization is more than 10%, and grosser granularity than 4*4 can get lower CPU utilization (six percent), but the average screen Data is higher than 2 KB per second, so we choose 4*4 for the tradeoff.

5 The application of the technology

Our screen compression tool SjtuRecorder have been used in the Distance Education College of Shanghai Jiaotong University, and currently there are more than 12,000 students are using the tool for real-time classes. More than eighty percent of them feel satisfied with the software, for they can get good video quality even their home's network bandwidth is pretty limited. Also, we would like to point out that because our compression technology uses lossless compression for the text and simple GUI image such icons, the projected video from the remote classrooms' projector still have very good quality.

6 Conclusion and future work

In this paper, we introduce one new technology to record the screen video for distance E-Learning, the technique divided the screen into 16 small rectangles, and used hook technology to trace the changed area of the screen and map the changed region into the 16 small rectangles, for the small changed rectangles, we use both lossy method (for complex pictures) and lossless compression method (for simple text and icons), and at the same time we use long history buffer for motion compensation (buffer eight frames).This method can get very low CPU utilization and high compression ratio compared with other screen compression and sharing tools.

At the same time, our compression tool can use both unicast and multicast to transfer the compressed data, and can record the video to the local machines.

The limitation of our algorithm is that it cannot compress the movies that are played on the screen; also the compression of natural scenery image is poorer than MPEG.

In the future, we would extend the compression tool with Layered Scalable Coding to fit the network bandwidth variation.

References

1. Yong Liu,Xiao,Luo,Critical Issues in the Screen Compression and Transfer Demonstration Ssytems",Journal of SouthWest Technology University,Vol18,No.3,2003.9
2. Renxiang Yan,Yuan Gao, "Design and Implementation of IP-Based Video Streaming System",Computer Engineering ,Vol.27 No.5,2001.5
3. Jie Li et.al, "Layered Coding for Video Over Network", Computer Engineering and Application, 1998.8
4. URL, http://www.microsoft.com/windows/netmeeting/
5. URL, http://www.symantic.com/pcanywhere/
6. URL, http://www.jpeg.org/
7. URL, http://datacompression.info/Compression.shtml/
8. URL,http://www.techsmith.com/products/studio/default.asp
9. URL,http://www.microsoft.com/windows/windowsmedia