A Simple Rendering System for Web Presentation

Liang Zhao and Hideo Yamamoto
Faculty of Engineering, Utsunomiya University
Utsunomiya, Tochigi, Japan 321–8585.
Email: {zhao, yamamoto}@is.utsunomiya-u.ac.jp

Abstract—We consider the problem of rendering a PC-projector oriented presentation to web based. For the maximum compatibility, the simplest way is to convert the original presentation into a sequence of images in a standard format (e.g. PNG) then perform a “slideshow” at the web server side. We further introduce techniques used in video compression to exploit the redundancy and to adapt for network heterogeneity. A simple implementation is also shown.

I. INTRODUCTION

For years, web presentation has been extensively studied. Various systems ([1], [4], [5], [6], [9]) and many standards (markup languages) such as HTML, XML and SMIL (see http://www.w3.org/, also [2], [3]) have been proposed and some have been widely tested. From these, we can see that web presentation has been becoming common in distance learning, distance meeting and other fields.

On the other hand, PC-projector based presentation is popular in local conferences, lectures, and etc. Comparing to traditional (e.g. OHP-based) methods, it is obvious that PC-projector presentation has the advantage. It is eco-friendly. Modification and distribution can be easily achieved. Moreover, most PC-projector presentation tools (e.g., the well-known Microsoft PowerPoint) have an easy-to-use graphical user interface for designing and presenting in an intuitive way. Thus recently more and more people have become accustomed to presenting with PC and projectors. This, on the other hand, brings us a new problem.

In our study (see Section II), we have observed that, there are many cases in which a web based presentation is more suitable than a PC-projector based one, even for local presentations. However, while most presenters are good at preparing PC-projector presentations, they usually know little about how to make a good web based one (which is generally more difficult). This stops them from making, or even trying to make, a web based presentation despite the adequateness.

In this paper we propose a solution to this problem. We consider to render a PC-projector oriented presentation to fit for web presenting. The goal is, given a PC-projector oriented presentation (in any format), the proposed system can convert it into a suitable format for web presentation, such that is acceptable by a large variety of information devices including computers, PDAs and cell phones. (Notice that our aim is not a perfect web presentation but a good rendering system which should be compatible with any PC-projector presentation tool.) For this purpose, we consider that the rendering system must satisfy the following requirements.

1) Compatibility: It must allow audiences to use different kinds of devices such as computers (running different OSes), PDAs and cell phones, to attend the presentation from everywhere.
2) Seamless transition: If needed, it must provide the presenter an easy way to convert a PC-projector based presentation into an appropriate format for presenting.
3) Simplicity and Uniformity: It should have the ability to automatically show the correct slide at the distance place, i.e., during the presentation, there is no need for the presenter to instruct audiences what and how to do, yet all audiences have the same slide (may in different resolutions though).
4) Scalability: It must perform well for a large number (as well as a small number, of course) of audiences, with appropriate consideration for the network or computing capacity heterogeneity.

Our first idea is to convert the input into a sequence of images in a standard format (e.g. PNG) and do a “slideshow” at the web server side. This simple idea makes it possible to carry out the rendering task. For it to be done more efficiently, we further introduce techniques that are known in the studies of video compression.

The rest of the paper is organized as follows. In Section II we discuss situations when a web based presentation is better than a PC-project based one. In Section III we show the framework of our solution in a theoretical aspect. A simple implementation using Java applet (and some others) is shown in Section IV. Finally we conclude with remarks in Section V.

II. THE PROBLEM

Our study was first motivated by the problem of how to efficiently perform a good presentation for the next two cases.

A. Distance learning

In distance learning, it is common for a lecturer to perform a PC-projector presentation at the local place, for the meanwhile the presentation is captured by a video camera, and the captured video is streamed to distance audiences via some kind of network (e.g. the Internet). Unfortunately, due to the limit of network capacity, it is quite often that the distance audiences can receive only a low-resolution yet lossy-compressed video, from which the presentation content itself is hard to recognize.

This problem may be solved in the future by introducing networks of high capacity. However, we believe that a better
solution is to transmit not the captured video but the presentation content itself, as well as the presenter's voice comment. This is because that:

- At first, the most important information sources in a distance learning are not the video but the presentation content and the presenter's voice comment.
- Secondly, comparing to video stream, a presentation that only consists of content and comment usually requires a much lower bandwidth. In fact, we have data shows that (see Section V), typically the content requires at most a few tens of kbps (compressed size, same in the following), whereas the comment requires tens of kbps. On the other hand, a moderate quality (i.e., the content can be recognized without difficulty) video would require several hundreds of kbps.
- Thirdly, decoding a compressed video stream in real time is usually quite difficult, especially for mobile devices. Hence we see that "content + comment" style is better than the traditional "captured video" style. Actually, with the help of our system, a distance audience can look at the presentation screen and hear the presenter's voice comment as if he/she was a local audience. Moreover, all this can be done at a bandwidth that is about 1/10 of a video-based system.

B. Lecturing in a large class

Similarly we meet a problem in lecturing in a large class, i.e., performing a PC-projector presentation for a large number of local audiences. In this case, a single projector may be insufficient since it could be difficult for audiences sitting in the rear to recognize the content projected on the front screen.

An obvious solution is to utilize multiple projectors and screens, connected by RGB distributors and cables. This method, however, has poor scalability. It is obvious that the above "content + comment" style presentation using a networked transmission (e.g., via wireless LAN) is a better solution. Hence again we can conclude with the necessity of rendering a PC-projector oriented presentation to a "content + comment" style networked presentation.

Also notice that, fortunately, we usually do not have to transmit the presenter's voice via the network in a room with good sound effect. In this case (which is often true), only the content needs to be transmitted. This further saves the network bandwidth (i.e., increase the scalability).

Let us finish this section with an illustration of a real application. In our university, there are a few programming related courses. Typically in such a class, there are 80 – 100 students sitting in a large classroom. Each student is sitting in front of a distinct PC (the display actually), where all PCs are connected to the department's LAN.

For presenting, a teacher usually has three choices:
1) Write on the front whiteboard.
2) Use a PC and the projector, where the (only one) screen is located in front of the classroom.
3) Prepare a text written in HTML, upload it to a web server and ask the students to operate their own PCs to find and follow what the teacher is talking about.

The first two are preferred by senior teachers, but as stated before, it is difficult for students sitting in the rear of the classroom to follow. On the other hand, the third choice is preferred by young teachers, who are usually familiar with the HTML language. This method can partly solve the problem of the first two. But however, it is still not a good solution.

One reason is that it is generally more difficult to prepare the lesson than the others (we think this is the reason why senior teachers are fond of the first two presentation methods). Furthermore, while HTML is suitable for self-study, forcing students to follow the teacher is always a hard work. That is, (unlike the others,) while talking, the teacher has no control on what the students are looking at, thus it is difficult for the teacher to let (to force) the students follow him or her.

We consider, once again, that the best solution in this case is to render a PC-projector based presentation to a web based "content" (or "content + comment" if necessary) style presentation, with the four requirements stated in Section I satisfied. Let us explain the details in the following.

III. The solution

In the previous section, we have discussed the importance of rendering a PC-projector presentation to a "content + comment" style web (network) based presentation. In this section, we propose a framework for this purpose. A simple implementation will be shown in the next section.

A. Simple solution

Recall that we have four requirements for our rendering system: "compatibility", "seamless transition", "simplicity and uniformity" and "scalability".

1) Compatibility: Particularly when we consider mobile devices, currently rendering to a WWW (HTTP) presentation is the best (if not the only) choice. Furthermore, we also want to convert the input presentation into a sequence of images in a standard format that can be widely viewed. Without this, it is almost impossible for a distance audience to have exactly the same screen as what is shown at the local place. For instance, a presentation containing text like an HTML page may suffer the problems of code-corruption; different browsers may produce different views, or even worse, text may be unreadable if the device does not support the same character set.

Fortunately we have the standard image formats. The PNG and JPEG are two examples. After converting the input into a sequence of images, we can perform a "slideshow" at the web server side, and the audience only needs to periodically fetch and show the images (and, if necessary, to fetch the presenter's voice in real time). We note that, there is a study ([8]) showing that in web based learnings, a slideshow style lecture is indeed effective. Hence we can say, from a different viewpoint, that the proposed slideshow-style system is a good solution.

2) Seamless transition: In fact, converting to images is not difficult for PC presentation tools. For example, Microsoft PowerPoint can export in GIF, PNG or JPEG format. Even if the presentation tool cannot export images, we can capture each slide by print-screen and save it in a standard format.
Thus converting a PC-projector presentation to images in a standard format is easy.

3) Simplicity and Uniformity: Notice that we need to synchronize the displayed image with the presenter’s voice. We consider that the simplest way is to use a unique URL for presentation. For example, suppose that we are using “http://192.168.0.1/slide.png”. At the server (IP=192.168.0.1) side, the presenter changes file “slide.png” to whatever he or she wants to show, whereas an audience needs only to access the above URL (to get the image) periodically.

In our implementation, we distribute a Java applet which runs at the audience side and automatically fetches and shows the specified image in a specified period. In this way, there is no need for a distance audience to care about how to synchronize the slides with the presenter’s voice (as the shown slide is changed at the presenter’s will). On the other hand, the presenter only needs to care the presentation at local place.

4) Scalability: Since the proposed system uses a much lower bandwidth, it performs much better than a video-based system. In a local place where only the content is needed to transmit, it can perform even better. Let us illustrate with our previously stated application (see Section II).

Let \( r \) be the maximum bandwidth required by the transmission. And let \( C \) be the minimum network capacity. Then \( C/r \) is the maximum number of audiences who can access the presentation server concurrently. In our application, typical values are \( r = 50 \text{ kbps} \) (see Remark 2 in Section V), \( C = 100000 \text{ kbps} \), thus \( C/r = 2000 \). We see there is no problem for a class of at most 100 students. Of course, it is rare for all the audiences to access the server at the same time.

Our simple solution is illustrated in Fig. 1.

![Diagram](image-url)

**Fig. 1.** An illustration of the simple solution.

### B. Advanced solution

While most presentations do not require a high bandwidth, complicated ones may require, e.g., a presentation containing a lot of animations, photos or even videos. For them, we have to render in a more efficient (hard) way. We consider the next techniques that are common in video compression.

1) **Entropy coding:** This includes Huffman coding and others. In our simple solution, however, since standard formats like PNG and JPEG are used, this feature is automatically implemented. Hence we will not discuss it in the following.

2) **Spatial redundancy exploitation:** Techniques like DCT or others can be used to exploit the spatial redundancy in one image. By the same reason as above, this feature is usually automatically implemented in our system. Thus it is not a topic in the following too.

3) **Motion compensation:** This is especially useful when the temporal redundancy between successive images is high. (Of course, it also increases the difficulty.) In our simple rendering system, however, we do not intend to handle video-embedded presentations, since currently all (general-purpose) video codecs are very complicated. Instead, we believe that a simple and presentation-oriented codec is needed (we are working at it). Our simple implementation can take advantage of the difference between two successive images, which we think can give a fairly good efficiency in an easy manner.

Finally we note that, while a regular video has a frame rate of \( 15 – 30 \text{ fps} \), a presentation (with no video embedded) usually has a much lower rate from as fast as several slides per minute to as slow as several minutes per slide. This difference may produce different treatment in considering motion compensation.

4) **Object coding, layered coding, etc:** For a simple animation embedded slideshows, object coding could be useful. But it increases the implementation difficulty. We refer the reader to an MPEG-4 reference for object coding.

Another important topic is how to adapt for heterogeneity, e.g., generally a mobile device can have only a low-resolution display and low computation power. Even for computers that have high resolution displays and high computation powers, there can exist the network heterogeneity.

An easy solution (which is used in our implementation) is to prepare multiple sources in different resolutions for the presentation, and let the audience choose the one that fits his/her equipment. For efficiency, **layered coding** should be a better solution. Of course, there are other techniques such as **multicast** that can be combined with layered coding (see e.g., [10] for more details).

Again, we note that these techniques must pay attention to this kind of application, since presentation has its own characters that may be ignored by a general-purpose video encoder. For example, the frame rates are different. Moreover, the desired picture qualities are different. Speaking precisely, we know that usually a presentation uses a lot of text messages, which could become unreadable if high frequency component of the image is heavily discarded. On the other hand, however, discarding high frequency component is a basic technique used in lossy image/video compression. This could bring us some trouble in applications, see [7] for an example.

Notice that the discussed advanced solution may (greatly) increase the implementation complexity. Currently our implementation (see the next section) does not support object coding, layered coding nor multicast.

### IV. A SIMPLE IMPLEMENTATION

So far, we have proposed a rendering framework which basically does a “slideshow” at the web server side, whereas each audience needs to fetch the images individually and periodically. In this section, we show a simple implementation.
At first, we need a web server (or any other that can provide clients images). We have chosen the Apache web server, which runs on a Linux PC (some other servers have been tested). “Slideshow” is executed at the server side, which actually consists of maintaining a symbolic link to the appropriate slide file according to the presenter’s command. We provide the presenter an intuitive utility for this purpose.

Next let us consider from the viewpoint of audiences. For simplicity, software installation at the audience side should be avoided. For this, currently we have two choices, either to use a WWW browser or to use the Java (applet) technology.

The idea of using a WWW browser has good compatibility, since recently most new information devices are equipped with WWW browsers. On the other hand, it also has many problems: firstly, showing in full-screen requires user’s operation. Secondly, it is not easy to control the browser to periodically access the same URL (cache-enabled browsers would not access the same URL in a short period. Unfortunately, cache is usually enabled by default.). Finally, there is no way for an ordinary browser to know how to deal with motion compensated images.

We choose Java applet in our implementation (of course, it is also possible to install a client software at the audience side). The only problem is that it requires a Java embedded device. But this is not a big problem since it is the recent trend. See the next web sites for more informations about the Java technology and its applications, especially to mobile devices.

- Java web site (http://java.sun.com/)
- Sun J2ME (http://java.sun.com/j2me/)
- Sun J2ME Devices (http://wireless.java.sun.com/device/)
- www.javamobiles.com (http://www.javamobiles.com/)

Now let us describe the usage of our implementation. Suppose that the presentation is performed at the next URL.


We first prepare an HTML page, slide.html, which is embedded with our Java applet. Then we ask the audiences to access “http://192.168.0.1/slide.html” (using a Java enabled WWW browser or other applet-aware viewers). The applet is then automatically downloaded and is executed at the audience side. It does nothing but to periodically fetch “http://192.168.0.1/slide.png” and show the image in full-screen mode. Both the URL and the refresh period can be specified in slide.html (which are passed to the applet). For PCs, instead of the HTTP protocol, the FTP protocol also works fine, i.e., supported by the Java virtual machine (VM). Standard image formats (GIF, PNG, JPEG) are supported.

Experiments have been done for the next combinations. We note that the (100% pure) Java applet should run under any Java embedded devices if an appropriate Java VM is available.

- **WWW server:** Apache/Linux, thttpd/Linux, vsftpd/Linux.
- **Client:**
  - Microsoft Internet Explorer (with the Sun’s Java 2 plugin) running on Windows PCs;
  - Mozilla, Konqueror (with the Sun’s Java 2 plugin) running on Linux PCs;
  - appletviewer (in Java 2 SDK) running on PCs;
  - evm (a J2ME appletviewer) running on a Sharp PDA Zaurus SL-C760.

See Fig. 2 for a screen shot (the original resolution is 1024x768 pixel) when the applet is running on a PC.

![Fig. 2. An illustration of the presentation screen on a PC.](image)

Fig. 2 shows the screen (original resolution is 640x480 pixel) on a Zaurus SL-C760. We note that colors in both images are reversed for printing purpose.

![Fig. 3. Presentation screen on a Zaurus SL-C760.](image)

We note that, as we use images (also due to the essential of the Java language specification), there is no significant difference between different browsers and applet viewers. Further details of the implementation are omitted due to space.

### V. Conclusion and Remarks

#### A. Conclusion

In this paper, we discussed the problem of rendering a PC-projector oriented presentation to a web based one. It is important in distance learning and other fields. We have proposed a simple solution, i.e., given an input, to convert
it into a sequence of images in a standard format. When the system is running, the presenter controls what image is currently shown at the web server side, while audiences can fetch the image periodically.

The proposed system requires a much lower transmission bandwidth than a video-based one. Actually, in our experience, it usually requires at most a few tens of kbps (see Remark 2). For complicated presentations that are embedded with a lot of animations or even videos, some advanced techniques used in video compression, e.g., motion compensation, can be applied with care about the characteristic of presentation.

We have also shown a simple implementation. It uses the Java applet technology, which is applicable to various computers and mobile devices. Using standard protocols, it should work with any HTTP or FTP servers. It also has the ability to handle difference images to take advantage of the temporal redundancy between successive images. As a future work, we plan to implement a more efficient solution that can treat complicated motion compensated images to further reduce the bandwidth.

B. Remark 1: a short study on networked projectors

This remark only applies to local presentations. In this case, now we have another solution — buy a newest high-class projectors, since recently many of them (e.g., Epson EMP-9300NL, Sony VPL-PX40) can show (though with limitation) slides via the TCP/IP network. These projectors actually can be viewed as a combination of a regular projector and a mini-computer, where a customized OS (Microsoft Windows variants as far as we know) is running between the projector hardware and the presenter PC. This increases the usability of the projector. But unfortunately (despite the price), they are designed to handle Microsoft PowerPoint files only. A question thus asks, how can they handle files produced by other presentation tools? Or even a future version of PowerPoint? Obviously an “answer” to the traditional RGB cables is not preferred. For the future, we suggest to embedded a Java VM (with applet support) in the projector and apply the proposed approach.

C. Remark 2: comparisons of different data format

Finally let us give a rough estimation on the data amount. We randomly picked up 10 Microsoft PowerPoint files from the Internet or from our presentations. The next table shows a comparison of the data sizes (in kilobytes) for three kinds of presentation methods. “PowerPoint” is the original file size. “HTML” denotes the total size of the HTML export of PowerPoint 2000. “Image” is the total size of the image export of PowerPoint 2000 (in GIF, PNG or JPEG format, depending on which has the smallest size).

We note that this is not an accurate evaluation, since the HTML export and image export results varies by the choice of resolution and compression parameters. The next table uses a resolution of 1024x768 (pixel), and uses the default parameters of PowerPoint 2000. We note that, the HTML export of PowerPoint cannot be viewed by many WWW browsers (thus has a poor compatibility), whereas our image-based solution can be widely viewed. On the other hand, however, in the image export, some slides containing animations get a poor looking (they collapse. This seems to be a bug of PowerPoint).

To our experience, if there are not so much animations, averagely it takes a presenter at least 20 seconds to explain a slide. Then the table gives an estimation on the bandwidth of the image-based solution to less than 50 kbps.

ACKNOWLEDGMENT

This research is partially supported by the International Communications Foundation (http://www.icf.or.jp).

REFERENCES